Movie Recommender System
Final report

Polina Lesak, BS21-DC-01(exchange student),
p.lesak@innopolis.university, 2023

Introduction

A recommendation system is an artificial intelligence or AI algorithm, usually associated with machine learning, that uses Big Data to suggest or recommend additional products to consumers. These can be based on various criteria, including past purchases, search history, demographic information, and other factors. Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own.

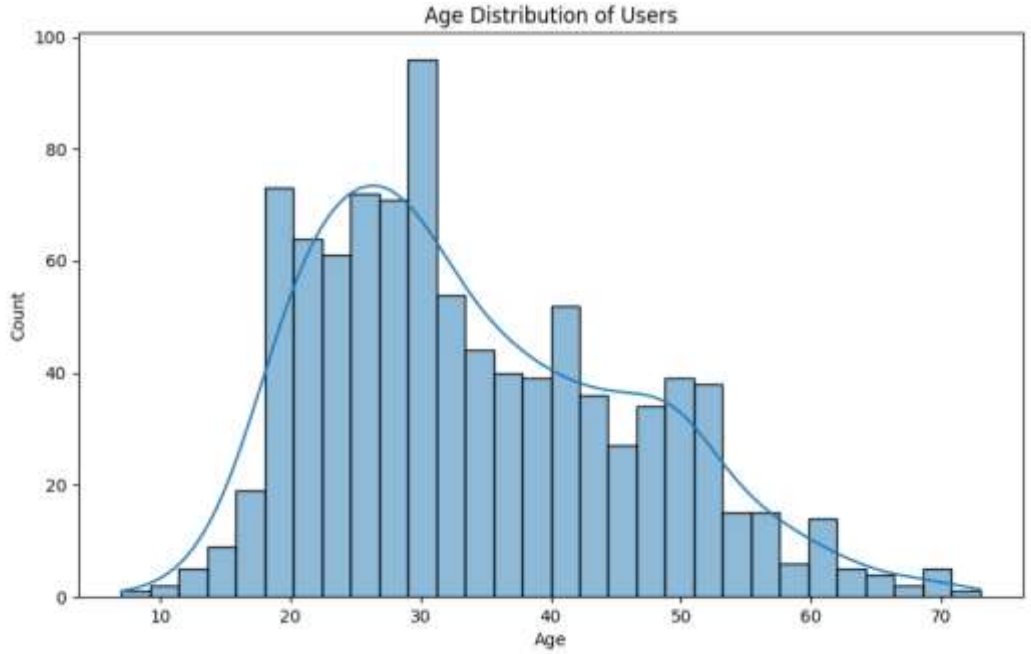
The task is to take into account the demographic data of the user when selecting movie recommendations. I decided to implement this in the following way:

1. Find "similar" users using their demographic data (profession, gender, age)
2. Predict ratings for those films that "similar" users rated above 3 and already from these predictions give recommendations for viewing.

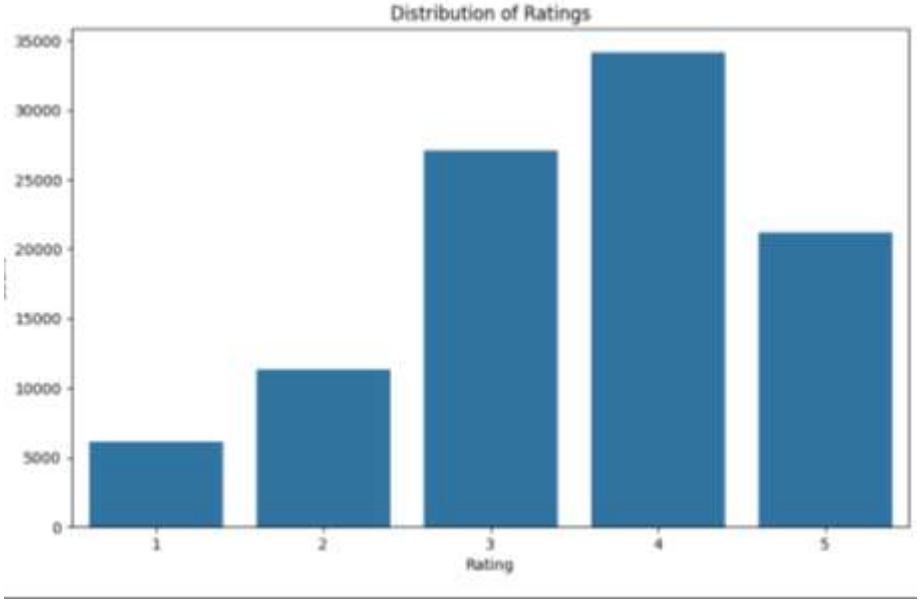
To find "similar" users, I decided to use the KNN model, and to get recommendations, the NKF model.

Data analysis

To analyze the data, I have plotted some indicators. The first graph is the distribution of users by age:

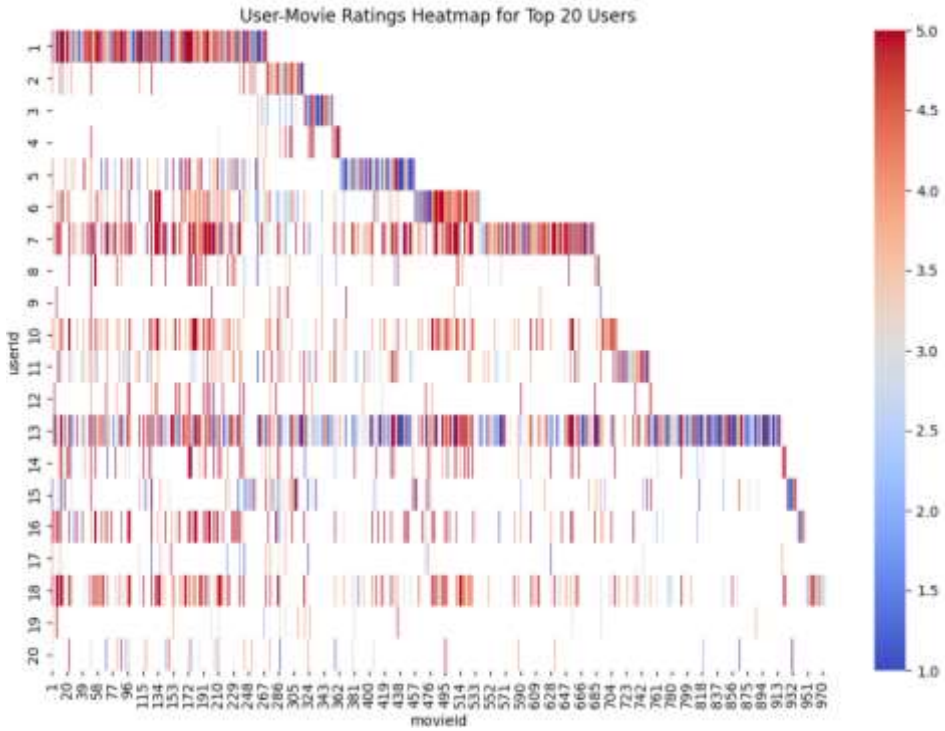


Based on the graph data, we can say that users are evenly distributed by age.

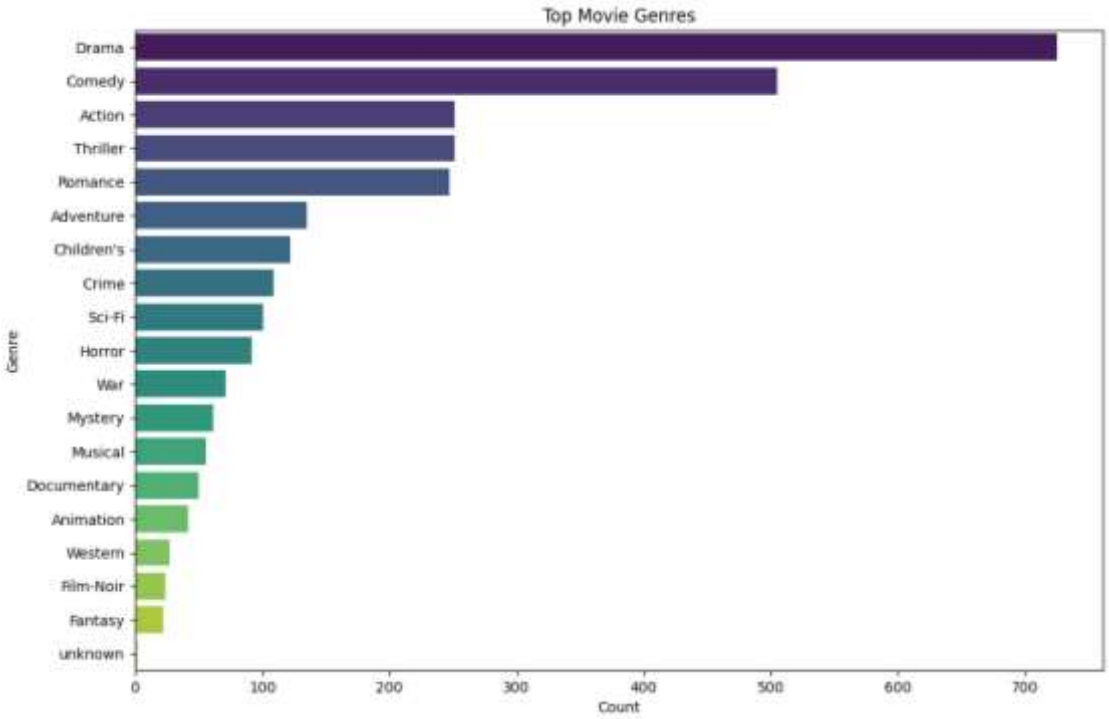


The second graph is the distribution of ratings, based on its data, we can say that on average users rate movies by 3-4, which is quite logical.

On the third graph, you can look at how the user evaluates films on average, there are users who give many films only bad ratings, and some, on the contrary, only good ones.



And on the last 4 charts, you can watch the most popular movie genres. It turned out that this is a Drama and a Comedy, and we can also say that there are very few films with an indeterminate genre.

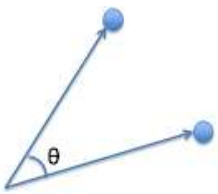


After analyzing the data, I realized that the data is really high-quality, each user rated 20+ movies, there are no strong outliers in the data, which facilitates the data processing process for learning. To use user demographic data in a machine learning model, we convert categorical features into numeric ones using LabelEncoder.

Model Implementation

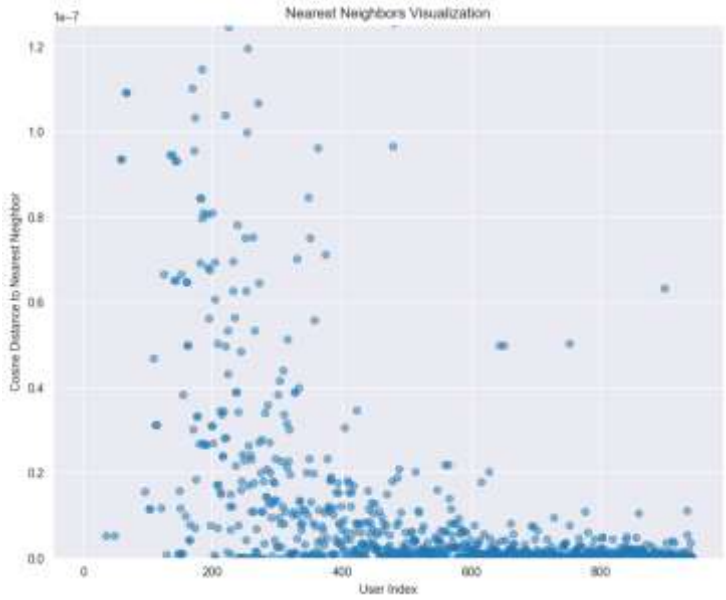
1. The KNN model was used to find similar users.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



The graph consists of points where each point represents a user (User Index), and the y coordinate of this point corresponds to the cosine distance to its nearest neighbor.

The point is near zero on the y axis, which indicates close neighbors for most users in the context of the selected cosine metric.



To test how the model works, let's look at the real data. I found top-10 users "similar" to 778.

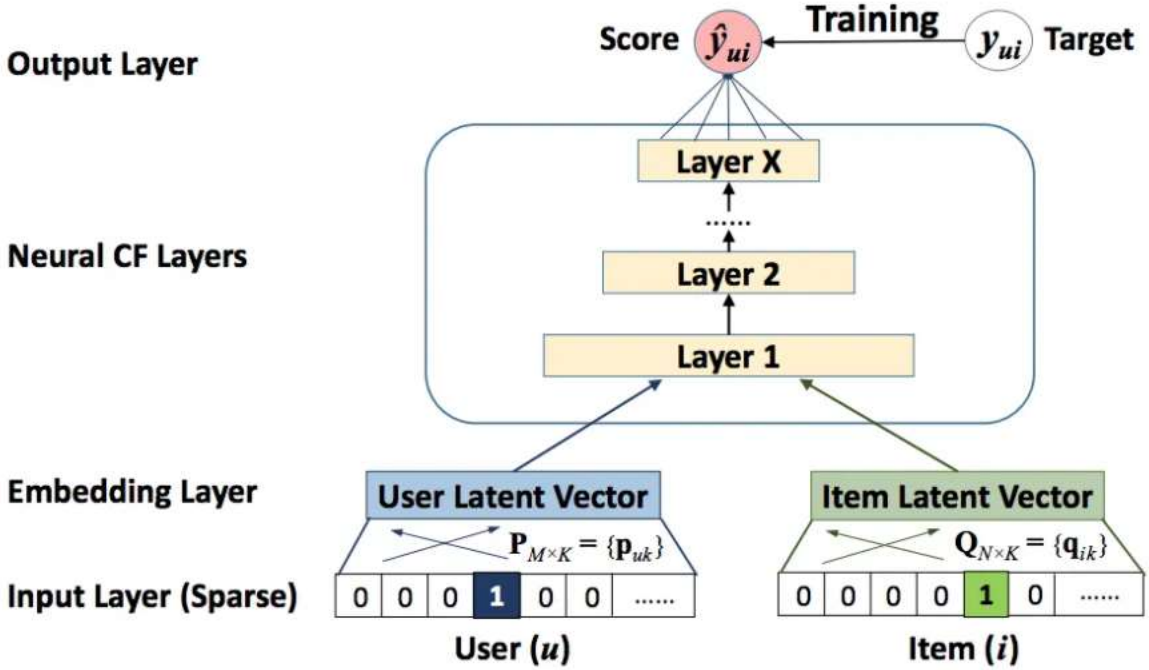
user_id	age	sex	occupation
727	25	M	student
742	35	M	student
757	26	M	student
758	27	M	student
765	31	M	student
770	28	M	student
771	26	M	student
774	30	M	student
778	34	M	student
779	31	M	student
847	29	M	student

Our user 778 is a 34 year-old male student "Similar" users are 25-35 year-old male students The demographic data is really similar, which means that the algorithm works correctly.

2. Neural Collaborative Filtering Model

Predict ratings for those films that "similar" users rated above 3 and already from these predictions give recommendations for viewing. Neural Collaborative Filtering is well suited for such purposes, with its help we will give recommendations to the user based on the films that he and other users have watched.

Neural Collaborative Filtering replaces the user-item inner product with a neural architecture.



Here's a brief overview of the architecture:

Input Layer:

- Two separate input layers for user and movie, each of size 1, representing user and movie indices.

Embedding Layers:

- Embedding layers for both user and movie, which transform the input indices into dense vectors of a specified size (embedding size).
- The embeddings capture latent factors for users and movies.

MLP Component:

- The embedded user and movie vectors are fed into separate Multi-Layer Perceptrons (MLPs).
- The output of each MLP is flattened to create vectors of fixed size.
- These vectors are concatenated to form a single vector.

Matrix Factorization Component:

- Another set of embedding layers for user and movie indices are used to create latent factor vectors.
- The dot product of the user and movie latent factor vectors is taken.

Concatenation and Fusion:

- The output vectors from the MLP and the dot product from the matrix factorization are concatenated.
- This combination aims to capture both the non-linear interactions and the latent factors.

Fully Connected Layers:

- The concatenated vector is passed through fully connected layers with batch normalization and dropout for regularization.
- This helps the model learn complex representations and prevent overfitting.

Output Layer:

- The final output layer with a single neuron and linear activation function is used to predict the rating or preference value.

Loss and Optimization:

- The model is trained using the mean squared error loss, commonly used for regression tasks.
- Adam optimizer with a specified learning rate is employed to minimize the loss during training.

In addition, when further using the model in a function that receives the top 10 recommendations, I used only those movies that were viewed by "similar" users for prediction.

Model Advantages and Disadvantages

Pros of Neural Collaborative Filtering Model:

1. The model captures non-linear interactions between users and items. Unlike traditional collaborative filtering methods that rely on linear relationships, neural collaborative filtering can learn complex patterns and relationships in the data.

2. The model uses embedding layers to create latent factor representations for users and items. This allows the model to learn and represent the underlying characteristics or features of users and items, capturing more nuanced information.

3. Neural collaborative filtering can generalize well to unseen users and items. The learned embeddings enable the model to make predictions even for users or items with sparse interaction histories, making it suitable for scenarios with evolving datasets.

Cons of Neural Collaborative Filtering Model:

1. Neural collaborative filtering models may require a relatively large amount of data to effectively learn meaningful representations. In scenarios with sparse data or cold start problems (new users or items with limited interaction history), the model may struggle to provide accurate recommendations.

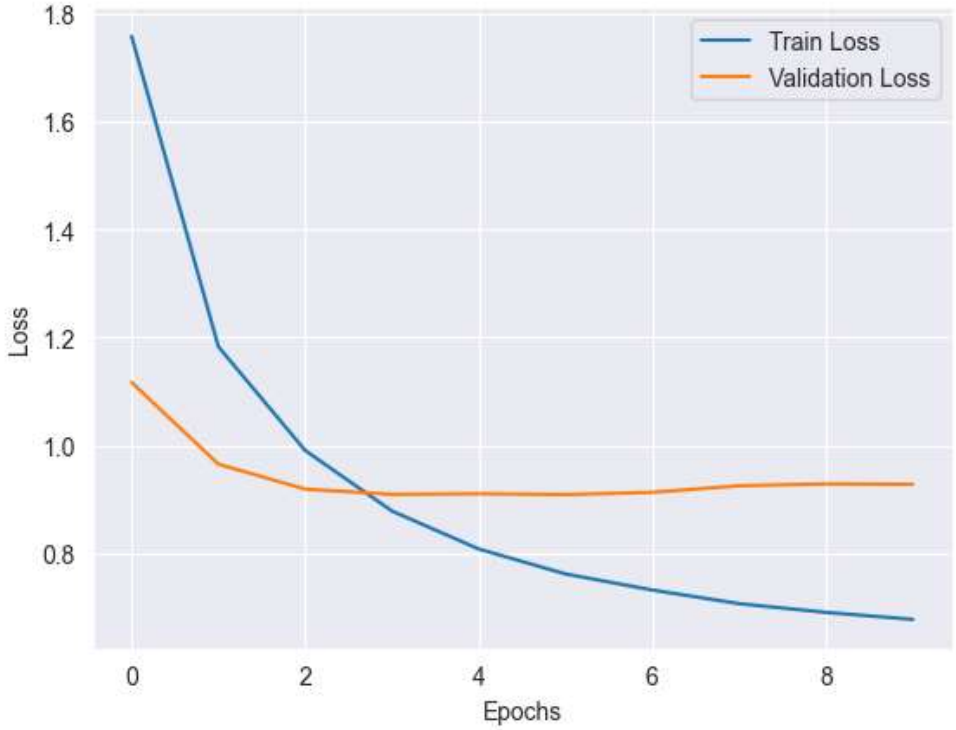
2. The highly non-linear nature of neural collaborative filtering models can make them less interpretable compared to simpler models. Understanding the reasons behind a specific recommendation might be challenging, limiting the model's transparency.

Training Process

At the beginning of the learning process, the collaborative filtering model is initialized with certain configurations. The size of the embedding layer is selected, equal to 5. This layer is crucial for collecting hidden factors representing users and

movies. Dropout helps prevent overfitting by randomly dropping out some compounds during each epoch.

The model is trained using a training set consisting of user indexes, movie indexes, and corresponding ratings. The learning process takes place over 10 epochs. The optimization algorithm updates the model parameters to minimize the loss of the standard error between the predicted and actual estimates. A validation set containing a separate set of user-movie pairs with scores is used to monitor the performance of the model on invisible data during training. The ModelCheckpoint callback is used to save the model with the least loss during validation, ensuring that the model generalizes well to new data.



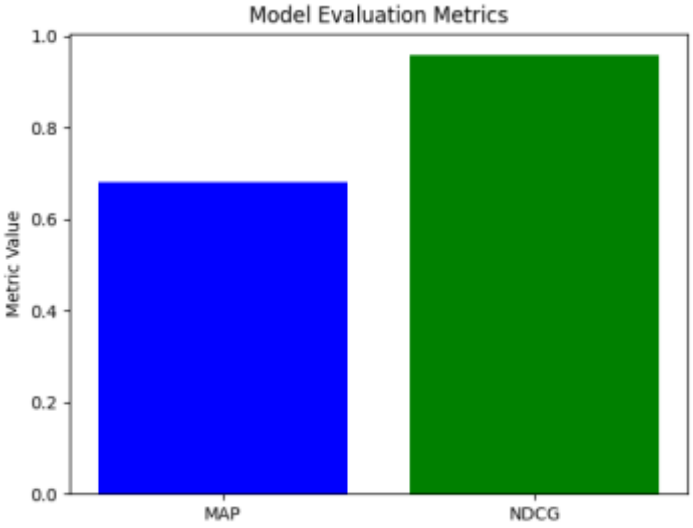
On the graph, you can see that the losses on validation data are the lowest in the 2nd epoch. The model with weights has been preserved since the 2nd epoch and it is used in the future

Evaluation

Evaluating will be carried out on a test dataset previously unknown to our model with best weights. MAP and NDCG are commonly used in recommender systems because they consider the ranking aspect and the relevance of items

1. Mean Average Precision (MAP): MAP evaluates the precision of the recommendations by considering the ranking of relevant items. It provides an average precision value, considering different users and their preferences.

2. Normalized Discounted Cumulative Gain (NDCG): NDCG is especially useful when the goal is to prioritize relevant items at the top of the recommendation list.



MAP = 0.68 is a relatively good score, suggesting that the model is effective in ranking relevant items higher.

NDCG = 0.957 is quite high, indicating that the model is providing good rankings for relevant items.

Results

To get movie recommendations run this script.

```
python src/get_recommendations.py <USER_ID> <NUM_OF_USERS_TO_FIND>
<NUM_FILMS_TO_PREDICT>
```

Example:

```
python src/get_recommendations.py 775 10 5
```

{"similar_users": [784, 788, 745, 772, 769, 855, 808, 829, 852, 766]}

Title	Prediction	MovieId	Genres	UserId	Rating
Star Wars (1977)	4.334508	50	Action, Adventure, Romance, Sci-Fi, War	852	5
Empire Strikes Back, The (1980)	4.328842	172	Action, Adventure, Drama, Romance, Sci-Fi, War	766	3
Princess Bride, The (1987)	4.2975836	173	Action, Adventure, Comedy, Romance	766	4
Schindler's List (1993)	4.272797	318	Drama, War	766	5
Some Folks Call It a Sling Blade (1993)	4.223508	963	Drama, Thriller	788	4

In the result table, we see the movie recommended for viewing by the user, the name of the movie, its genre and id, as well as the rating given by a "similar" user to this movie