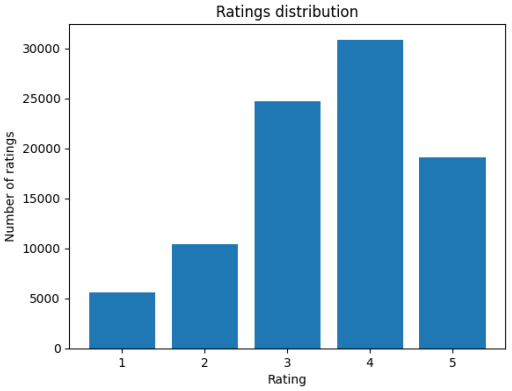
Final report

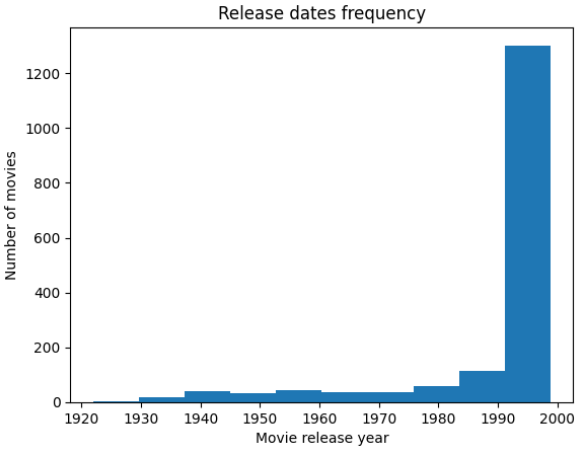
Data analysis

Before implementing a task, I analyzed the data. Full analysis is shown in the notebook 1. The dataset is MovieLens 100K.

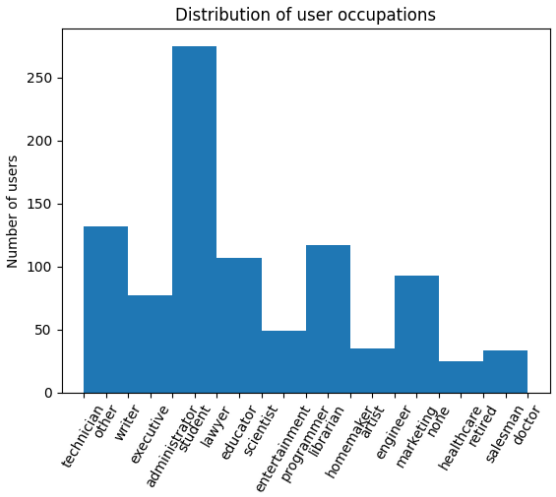
I used *ua.base* and *ua.test* as train and test datasets respectively. Train set contains at least 1 rating from each user, but does not have any rating for 2 movies that occur in the test set. The ratings are not balanced - for some movies there are more than *400* ratings, while for some others around *0*. The same imbalance seen in the test set. Similarly, the user ratings distribution is imbalanced. Also, ratings distribution is not equal - most of the ratings are 4. Similar distribution is seen in the test data.



Also, I analyzed information about movies. Its video_release_date is also None, so do not give value and may be dropped. The unusual datapoint is "unknown" film. I supposed that in real use this datapoint may represent films that are not in the database (for example, the new one). It should be noted that most of the films are in the range of *1990-2000* years, while the fist film is of *1922* year. That means that the database is old, and movies distribution with respect to dates is imbalanced too.



Users' information was analyzed too. It should be noted that the number of men is bigger than women, the age distribution is around the shifted normal, the most frequent occupation is “student”, “other” and “educator”.



Data preprocessing

To preprocess movies I vectorized title (without release year) with pre-trained sentence transformer, dropped video release date and links columns, reformat release date to date type and normalized it.
As user information preprocessing, I normalize age, encode gender and occupation by one hot encoding.
Also, I added average rating from all users to each movie (to the movie info), and mean movie rates given by a particular users (to the user info)

As a final dataset, I merged rates data with user info and item info. Overall, I obtained *426* features (without ground-truth rating) per entry.

Model Implementation

Main idea of my system is to pass all movies that the user did not watch before to a model that will predict rating this user can give to a film, sort movies by predicted ratings and show *k* best. By intuition, rating is how a person likes an item, so it is a continuous value. That means the task is to solve a regression problem.

As a model that will predict ratings I created a feed-forward neural network with *426* input neurons and *1* output. It contains *3* linear layers with *LeakyReLU* and *ReLU* activation functions. Before making a prediction, I applied sigmoid to put the output in a range *[0, 1]*. Also, during training, with *0.5* probability a dropout was used.

Model Advantages and Disadvantages

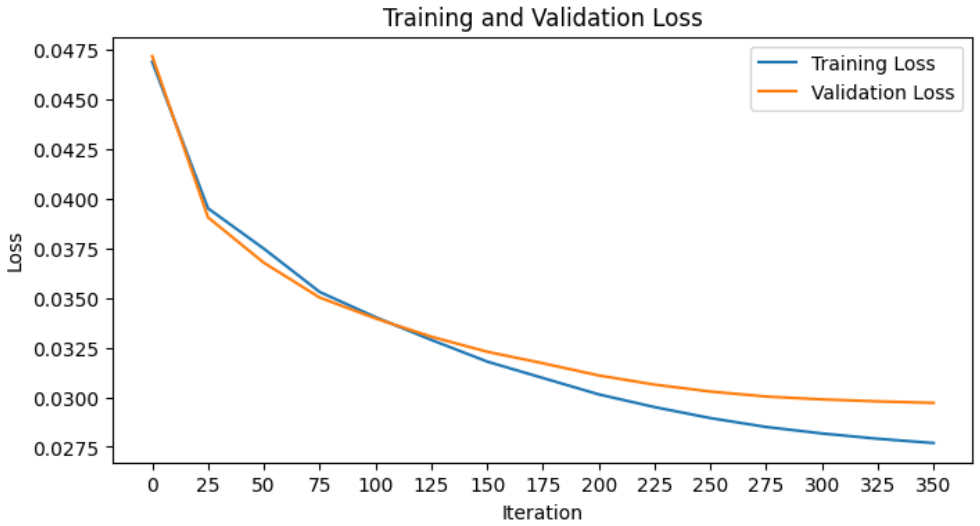
First of all, an implemented model is a lightweight neural network. It does not need a lot of resources: it may be trained in a short time even with cpu. Also, the model needs little time to make a prediction, as it needs only user and film features from the preprocessed dataset.
Moreover, the output of the model is a continuous value from *0* to *1*. It may be considered as a discrete rating, given by the user. However, using it as the continuous number gives an opportunity to sort movies by their ratings more accurately. Also, the architecture of the model as regression is more promising compared to classification, because rating *1* is better than *0*, but classification loses this meaning.

On the other hand, predictions of the implemented model are difficult to interpret. We could not say precisely what influenced the model's decision and which features had greater importance.
Also, the performance of the model depends on the quality of the training data. However, the used dataset is imbalanced both in terms of ratings and the number of evaluations left on the movie or from the user. For example, predicting always a rating *4* will provide not bad scores.
Moreover, to give recommendations to a user the model needs to evaluate all unseen films from the dataset and sort by predicted ratings. If the number of movies in the media library tends to infinity, making a recommendation can take a significant time.

Additionally, one should note that the neural network may find important patterns in user and film features, as well as in their combination. Also, the input features contain average ratings given by the user and mean ratings of a film, which can improve predictions. However, the implemented model does not have user preferences explicitly as input, only known metadata, and learn them during training. Unfortunately, there could be a situation where 2 users with the same features like completely different movies. This may confuse the model, so in the future it is worth adding features responsible for user preferences.

Training process

As a loss function, a HuberLoss was applied. It combines advantages of both Mean Absolute Error (MAE) Loss and Mean Squared Error (MSE) Loss: it is less sensitive to outliers and smooth near 0. The training was performed on 350 epochs with a shuffled train dataset. Ground-truth ratings were normalized to a range [0, 1]. Also, each 25 training epoch I showed a loss of validation set and its binary accuracy (4-5 became positive, 1-3 became negative). The dynamics of loss decline is shown on the image:



Evaluation

To evaluate the model, I showed MAE, MSE and Root Mean Square Error (RMSE) for both normalized and initial ratings. As stated in found articles, MAE and RMSE are popular metrics for evaluation of recommender systems. This metrics for initial (unnormalized) ratings are:

```
MAE: 0.689
RMSE: 0.817
```

That is good for ratings with range [1, 5], so the predictions usually are close enough to the real values.
Also, I created a function that will print top *k* movies that were unseen by the user before (both in train and test sets). Example of 10 predicted movies for user with id 21:

```
Recommended movies for user 21 are
Wrong Trousers, The (1993)
Santa with Muscles (1996)
Schindler's List (1993)
Citizen Kane (1941)
They Made Me a Criminal (1939)
Casablanca (1942)
Usual Suspects, The (1995)
Rear Window (1954)
Third Man, The (1949)
One Flew Over the Cuckoo's Nest (1975)
```

Results

The implemented movie recommendation system is based on a feed-forward neural network. It predicts ratings that a user may give to a movie. The model was trained on MovieLens dataset using HuberLoss. The evaluation was made with MAE and RMSE, the obtained results are good for a simple model.

The system can be further developed in the future. For example, using more advanced model architecture, bigger and more balanced dataset, extending user features by user preferences in movies may improve the performance of the system.