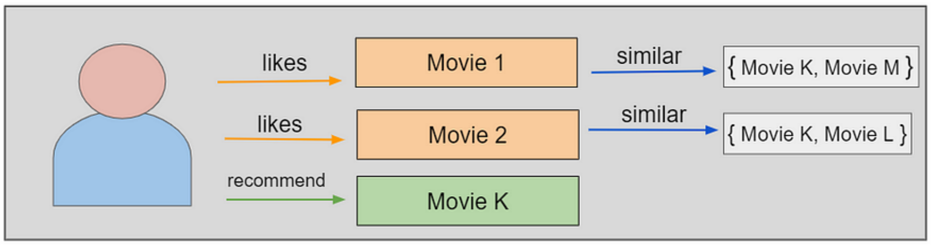
Final Report

Introduction

A group of methods and algorithms known as recommender systems are capable of making "relevant" suggestions to users. They use a variety of methods, including matrix factorization, to forecast future behavior based on historical data [3].
Two types of machine learning algorithms are frequently observed in recommender systems: collaborative filtering systems and content-based systems. Both of these approaches are combined in modern recommender systems.

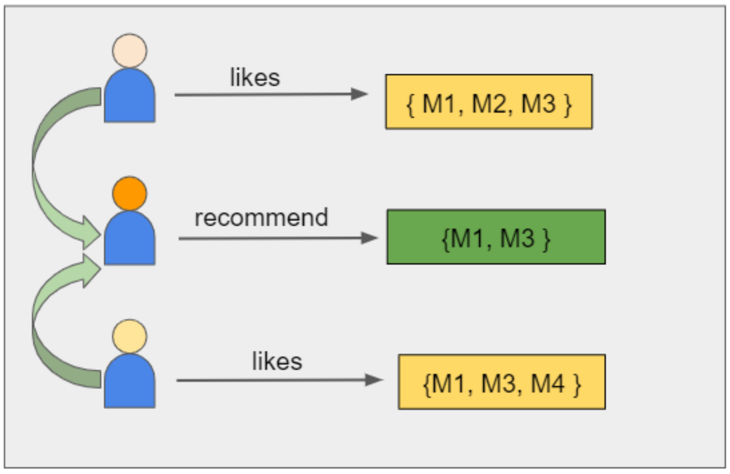
1. Content-Based Movie Recommendation Systems

The shared characteristics of movie attributes is the foundation of content-based techniques. If a user views one movie this system suggests more similar movies. Hence, movie features are the input used to create a content-based recommender system.



2. Collaborative Filtering Movie Recommendation Systems

The system uses previous interactions between users and movies for collaborative filtering. Hence, previous information on user interactions with the movies they view becomes the input for a collaborative filtering system.



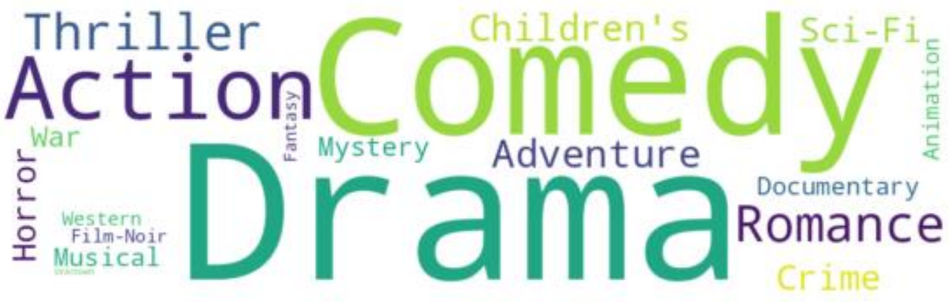
The user and movie interactions matrix is a matrix in which the user data is arranged as rows and the movie data as columns.

Data analysis

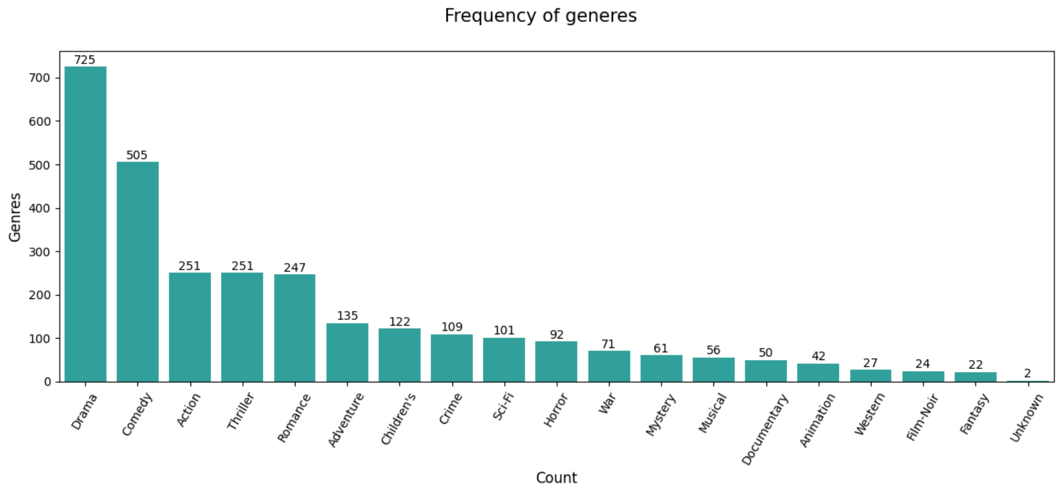
In this assignment I used MovieLens 100K dataset consisting user ratings to movies [1].
Initial analysis of the 3 dataset files ("u.item", "u.data" and "u.user") showed that:

- Number of ratings - 100000
- Number of unique movies - 1682
- Number of unique users - 943
- Average ratings per user - 106.04
- Average ratings per movie - 59.45

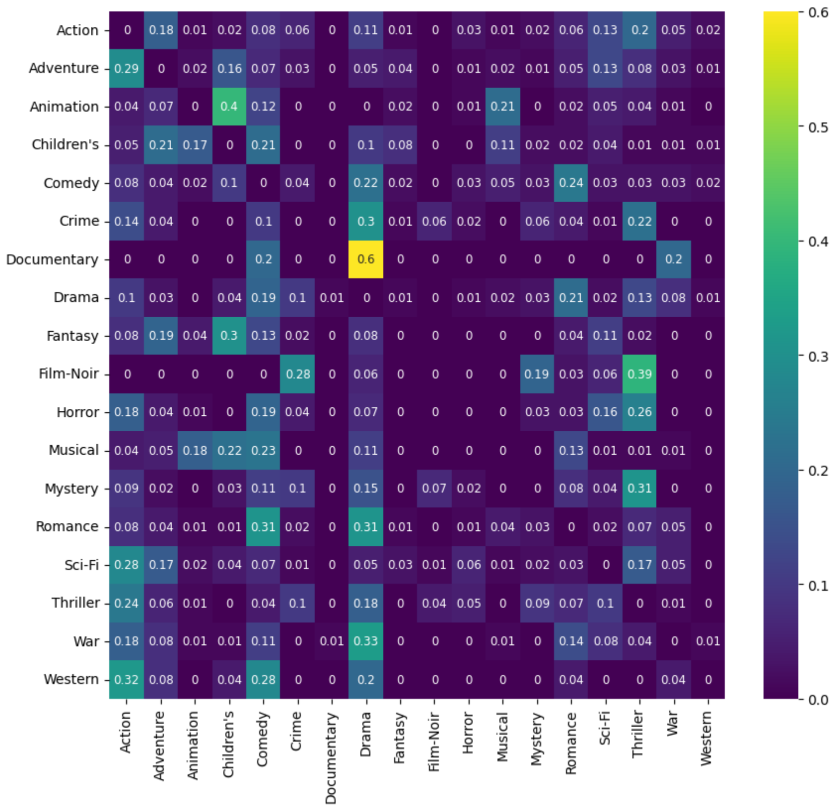
Basically, this means that an arbitrary user watched only a small amount of movies.
By visualizing a movie genre cloud in which the size of a genre is proportional to its frequency, I found that the dataset is not balanced with respect to movie genres.



My findings were confirmed when I plotted the genre frequencies with the number of occurrences, for clarity.



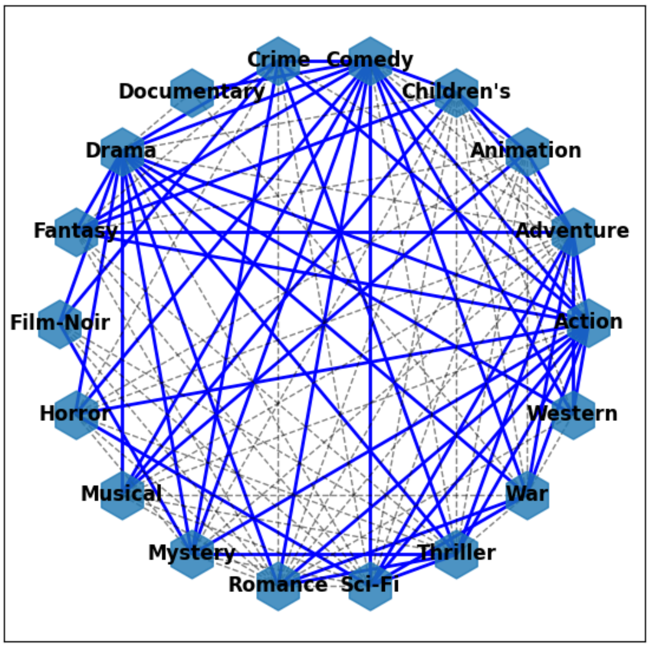
I decided to see if different genres could often appear together. To do this, I constructed a co-occurrence matrix between genres.



	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	0	0.18	0.01	0.02	0.08	0.06	0	0.11	0.01	0	0.03	0.01	0.02	0.06	0.13	0.2	0.05	0.02
Adventure	0.29	0	0.02	0.16	0.07	0.03	0	0.05	0.04	0	0.01	0.02	0.01	0.05	0.13	0.08	0.03	0.01
Animation	0.04	0.07	0	0.4	0.12	0	0	0	0.02	0	0.01	0.21	0	0.02	0.05	0.04	0.01	0
Children's	0.05	0.21	0.17	0	0.21	0	0	0.1	0.08	0	0	0.11	0.02	0.02	0.04	0.01	0.01	0.01
Comedy	0.08	0.04	0.02	0.1	0	0.04	0	0.22	0.02	0	0.03	0.05	0.03	0.24	0.03	0.03	0.03	0.02
Crime	0.14	0.04	0	0	0.1	0	0	0.3	0.01	0.06	0.02	0	0.06	0.04	0.01	0.22	0	0
Documentary	0	0	0	0	0.2	0	0	0.6	0	0	0	0	0	0	0	0	0.2	0
Drama	0.1	0.03	0	0.04	0.19	0.1	0.01	0	0.01	0	0.01	0.02	0.03	0.21	0.02	0.13	0.08	0.01
Fantasy	0.08	0.19	0.04	0.3	0.13	0.02	0	0.08	0	0	0	0	0	0.04	0.11	0.02	0	0
Film-Noir	0	0	0	0	0	0.28	0	0.06	0	0	0	0	0.19	0.03	0.06	0.39	0	0
Horror	0.18	0.04	0.01	0	0.19	0.04	0	0.07	0	0	0	0	0.03	0.03	0.16	0.26	0	0
Musical	0.04	0.05	0.18	0.22	0.23	0	0	0.11	0	0	0	0	0	0.13	0.01	0.01	0.01	0
Mystery	0.09	0.02	0	0.03	0.11	0.1	0	0.15	0	0.07	0.02	0	0	0.08	0.04	0.31	0	0
Romance	0.08	0.04	0.01	0.01	0.31	0.02	0	0.31	0.01	0	0.01	0.04	0.03	0	0.02	0.07	0.05	0
Sci-Fi	0.28	0.17	0.02	0.04	0.07	0.01	0	0.05	0.03	0.01	0.06	0.01	0.02	0.03	0	0.17	0.05	0
Thriller	0.24	0.06	0.01	0	0.04	0.1	0	0.18	0	0.04	0.05	0	0.09	0.07	0.1	0	0.01	0
War	0.18	0.08	0.01	0.01	0.11	0	0.01	0.33	0	0	0	0.01	0	0.14	0.08	0.04	0	0.01
Western	0.32	0.08	0	0.04	0.28	0	0	0.2	0	0	0	0	0	0.04	0	0	0.04	0

It turned out that most genres have a rather low correlation. Only Documentary & Drama (0.6), Animation & Children's (0.4) and Film-Noir & Thriller (0.39) showed up.

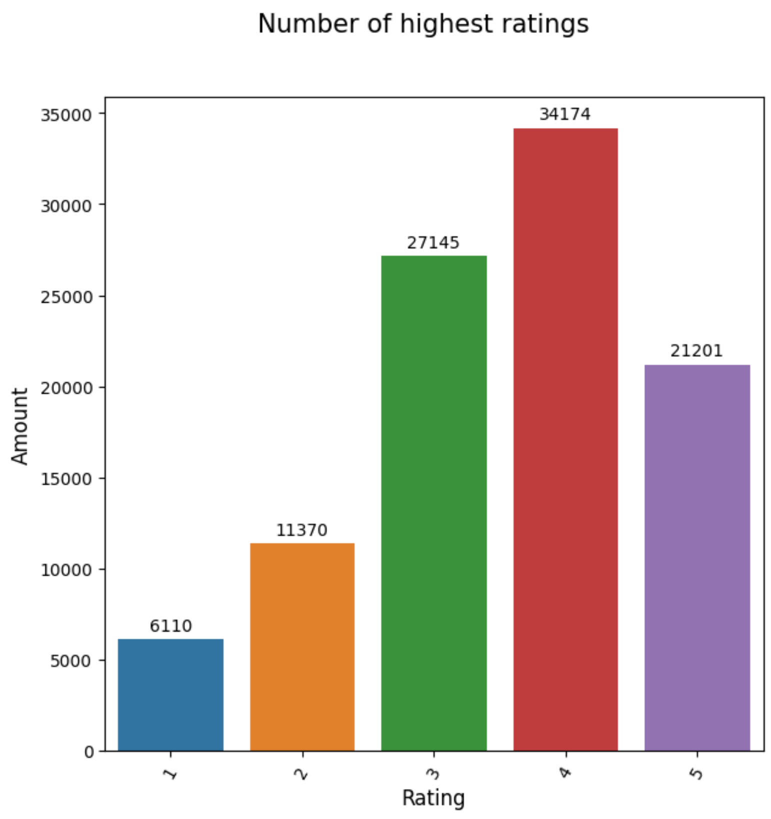
In addition, I constructed a genre graph in which the edge appeared for those genres whose correlation was above average.



In addition, in the process of analysis, I found that the data on ratings is imbalanced. The number of positive reviews (3, 4, 5), significantly exceeds the number of negative reviews (1, 2).
Rating Distribution:

- 6110 films with rating 1
- 11370 films with rating 2
- 27145 films with rating 3
- 34174 films with rating 4
- 21201 films with rating 5

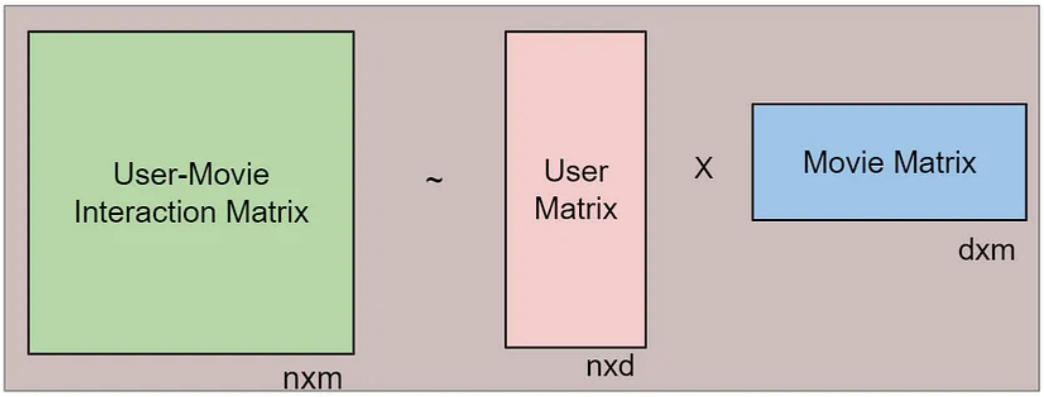
The rating 4 stands out the strongest.



Model implementation

I made the decision to predict the users' ratings for the films they haven't seen yet in order to provide movie recommendations. Then, based on these predicted ratings, customers receive suggestions for the best movies.

To find the similarity between movies/users I used a cosine similarity function. The first step was creating a matrix factorization based model. I decided to use the output of this model as additional features for the initial dataset, which will be passed to the final model.
I performed matrix factorization by the SVD from the Surprise library, which is used to build and test recommendation systems.



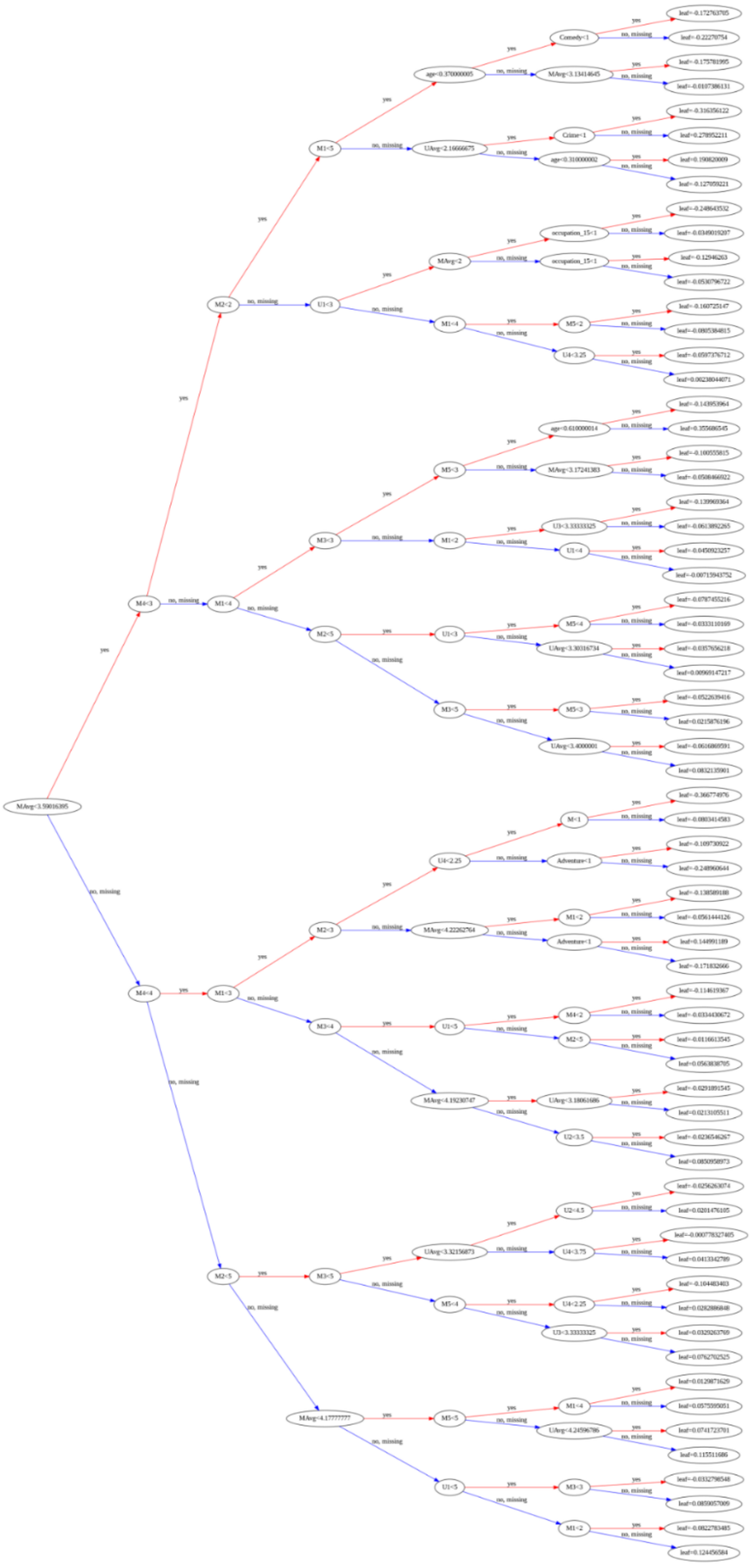
The initial data was converted to format of dataset, which Surprise accepts. Fitted SVD model was used to produce additional features, which will further be passed to the final model.

The next step was to create a user-movie interaction matrix based on the ratings from the training dataset. The scipy.sparse.csr_matrix function was used for this purpose. Using this matrix I decided to create additional support features based on ratings left by similar users or received for similar movies. This means that for each pair of user and movie, I looked for users similar to that particular user and added their ratings for that particular movie to the dataset. Similarly with movies, I took movies similar to the given movie and added the ratings assigned by the given user to those movies. If there were not enough users or movies, the ratings were supplemented with the average rating for that user or for that movie. In particular, I considered the top 5 similar users and top 5 similar movies. Also, for each pair an additional feature was added: average rating for this movie and average rating of this user.

Next step, I decided to encode some features of the original dataset to bring them to one of the following data types: boolean, integer, float, or categorical. For this purpose, OneHotEncoding was applied on the gender and occupation attributes. In addition, age was normalized to the range [0;1].
Finally, I removed some features that are difficult to work with or may not potentially carry useful information, such as movie URL, zip code, and timestamp.

XGBRegressor from XGBoost, an optimized distributed gradient boosting library, was chosen as the basic model. XGBRegressor was implemented with 13 jobs and 100 estimators, as evaluation metric RMSE was chosen.

After training on preprocessed data the following model was obtained:



MAvg<3.59016395
yes
no, missing
M4<3
M2<2
M1<5
age<0.370000005
Comedy<1
leaf=-0.172763705
leaf=-0.22270754
MAvg<3.13414645
leaf=-0.175781995
leaf=-0.0107386131
UAvg<2.16666675
Crime<1
leaf=-0.316356122
leaf=0.278952211
age<0.310000002
leaf=0.190820009
leaf=-0.127059221
U1<3
MAvg<2
occupation_15<1
leaf=-0.248643532
leaf=-0.0349019207
occupation_15<1
leaf=-0.12946263
leaf=-0.0530796722
M1<4
M5<2
leaf=-0.160725147
leaf=-0.0805384815
U4<3.25
leaf=-0.0597376712
leaf=0.00238044071
M1<4
M3<3
M5<3
age<0.610000014
leaf=-0.143953964
leaf=0.355686545
MAvg<3.17241383
leaf=-0.100555815
leaf=-0.0508466922
M1<2
U3<3.33333325
leaf=-0.139969364
leaf=-0.0613892265
U1<4
leaf=-0.0450923257
leaf=-0.00715943752
M2<5
U1<3
M5<4
leaf=-0.0787455216
leaf=-0.0333110169
UAvg<3.30316734
leaf=-0.0357656218
leaf=0.00969147217
M3<5
M5<3
leaf=-0.0522639416
leaf=0.0215876196
UAvg<3.4000001
leaf=-0.0616869591
leaf=0.0832135901
M4<4
M1<3
M2<3
U4<2.25
M<1
leaf=-0.366774976
leaf=-0.0803414583
Adventure<1
leaf=-0.109730922
leaf=-0.248960644
MAvg<4.22262764
M1<2
leaf=-0.138589188
leaf=-0.0561444126
Adventure<1
leaf=0.144991189
leaf=-0.171832666
M3<4
U1<5
M4<2
leaf=-0.114619367
leaf=-0.0334430672
M2<5
leaf=-0.0116613545
leaf=0.0563838705
MAvg<4.19230747
UAvg<3.18061686
leaf=-0.0291891545
leaf=0.0213105511
U2<3.5
leaf=-0.0236546267
leaf=0.0850958973
M2<5
M3<5
UAvg<3.32156873
U2<4.5
leaf=-0.0256263074
leaf=0.0201476105
U4<3.75
leaf=-0.000778327405
leaf=0.0413342789
M5<4
U4<2.25
leaf=-0.104483403
leaf=0.0282886848
U3<3.33333325
leaf=0.0329263769
leaf=0.0762702525
MAvg<4.17777777
M5<5
M1<4
leaf=0.0129871629
leaf=0.0575595051
UAvg<4.24596786
leaf=0.0741723701
leaf=0.115511686
U1<5
M3<3
leaf=-0.0332798548
leaf=0.0859057009
M1<2
leaf=-0.0822783485
leaf=0.124456584

Model Advantages and Disadvantages

Advantages of the resulting model:

- fast training
- usage of ensemble techniques
- reduced risk of overfitting and underfitting
- easily interpretable results
- continuous value as output, which means that model may learn that rate 5 is more than 1 shows that rate 5 is better
- additional features based on similarity between users/movies, which shows users preferences

Disadvantages of the resulting model:

- long preprocessing, which results in long prediction
- large tree, which makes harder to interpret results
- many features were not used in resulting tree, which means that data may be a bit overwhelmed
- training on imbalanced data, as dataset has non-equal number of different rates, genres and etc.
- as users in average watched small part of whole movies, it takes long time to make a prediction as it’s slow to process all unseen movies

Training Process

The resulting dataset after preprocessing was used to train the XGBRegressor model. RMSE was used as a metric. The trained model was saved to a json file. The training process was very fast and took no more than a second.

Evaluation

Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) are the two primary metrics used to assess a recommender system's performance. Whereas MAPE calculates the absolute loss, RMSE calculates the squared loss. Smaller values indicate improved performance due to decreased mistake rates.

My model resulted in 0.879 RMSE, and 26.3 MAPE on the preprocessed test dataset, which may be considered as good result. According to [3] RMSE value of less than 2 is considered good, and a MAPE less than 25 is excellent.

Results

In addition, my model was able to successfully recommend several movies to a user (in particular a user with id 1) based on information about the user, the movies, and the user's preferences. For this reason, as well as the results of the model performance evaluation, this project can be considered a success.

Reference

[1] - F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” ACM Trans. Interact. Intell. Syst., vol. 5, no. 4, pp. 1–19, 2016. http://dx.doi.org/10.1145/2827872

[2] - F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” ACM Trans. Interact. Intell. Syst., vol. 5, no. 4, pp. 1–19, 2016.

[3] - R. Vidiyala, “How to build a movie recommendation system,” Towards Data Science, 02-Oct-2020. [Online]. Available: https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109. [Accessed: 03-Dec-2023].

[4] - P. Aher, “Evaluation metrics for recommendation systems — an overview,” Towards Data Science, 09-Aug-2023. [Online]. Available: https://towardsdatascience.com/evaluation-metrics-for-recommendation-systems-an-overview-71290690ecba. [Accessed: 03-Dec-2023].

[5] - “Recommendation system in python,” GeeksforGeeks, 18-Jul-2021. [Online]. Available: https://www.geeksforgeeks.org/recommendation-system-in-python/. [Accessed: 03-Dec-2023].