**Final Report (AI Capstone 1-2)**

Team *Whatflix*

ByungHyun Lee

JunYeong Ahn

JiHong Jeong

December 20~23, 2020

**<Problem & Goal>**

Since COVID-19 makes our living inactive these days, more and more people are using movie-watching platform during self-quarantine. Therefore, platforms want to help customers find movies they want to be distinctive like Netflix, the industry-leader.

To develop unique movie recommendation ‘system’ using XGBoost ML model mainly based on similarity analysis, focusing on application in real-world. If we input ratings & genres history of the user, it should yield predicted ratings for other movies to be able to recommend top N rated movies.

**<Data>**

Original data : ‘ml-latest-small’ from Movielens, consisting of six files (ratings, movies, links.csv and other three for tags(metadata))

What we use : 1) ratings.csv 2) movies.csv

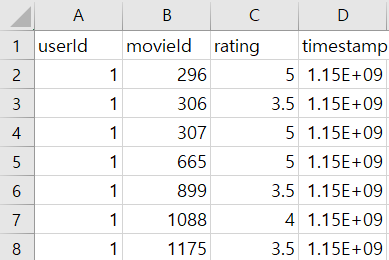
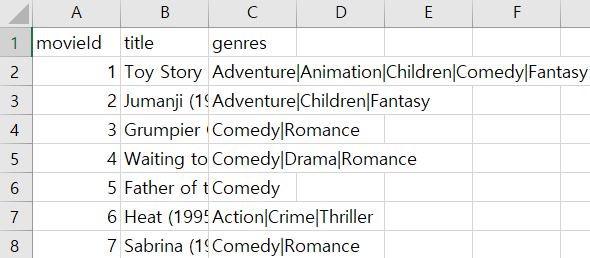
 

Table 1. Left : ratings.csv that contains four columns; userId, movieId, rating, timestamp. Right : movies.csv that contains three columns; movieId, title, genres.

**Main predictors** : ‘sim\_user’s and ‘sim\_movie’s columns created from ‘userId’, ‘rating’(for ‘sim\_user’) and ‘genres’ (for ‘sim\_movie’) columns (elaborated later in **Table 2**)

**Target variable** : ‘rating’ column

\* License : GroupLens, the owner of the Movielens, allow data to be used for any research purposes under the following two main conditions;

* 1) The user may redistribute the data set, including transformations, so long as it is distributed under these same license conditions.
* 2) The user may not use this information for any commercial or revenue-bearing purposes without first obtaining permission from a faculty member of the GroupLens Research Project at the University of Minnesota.

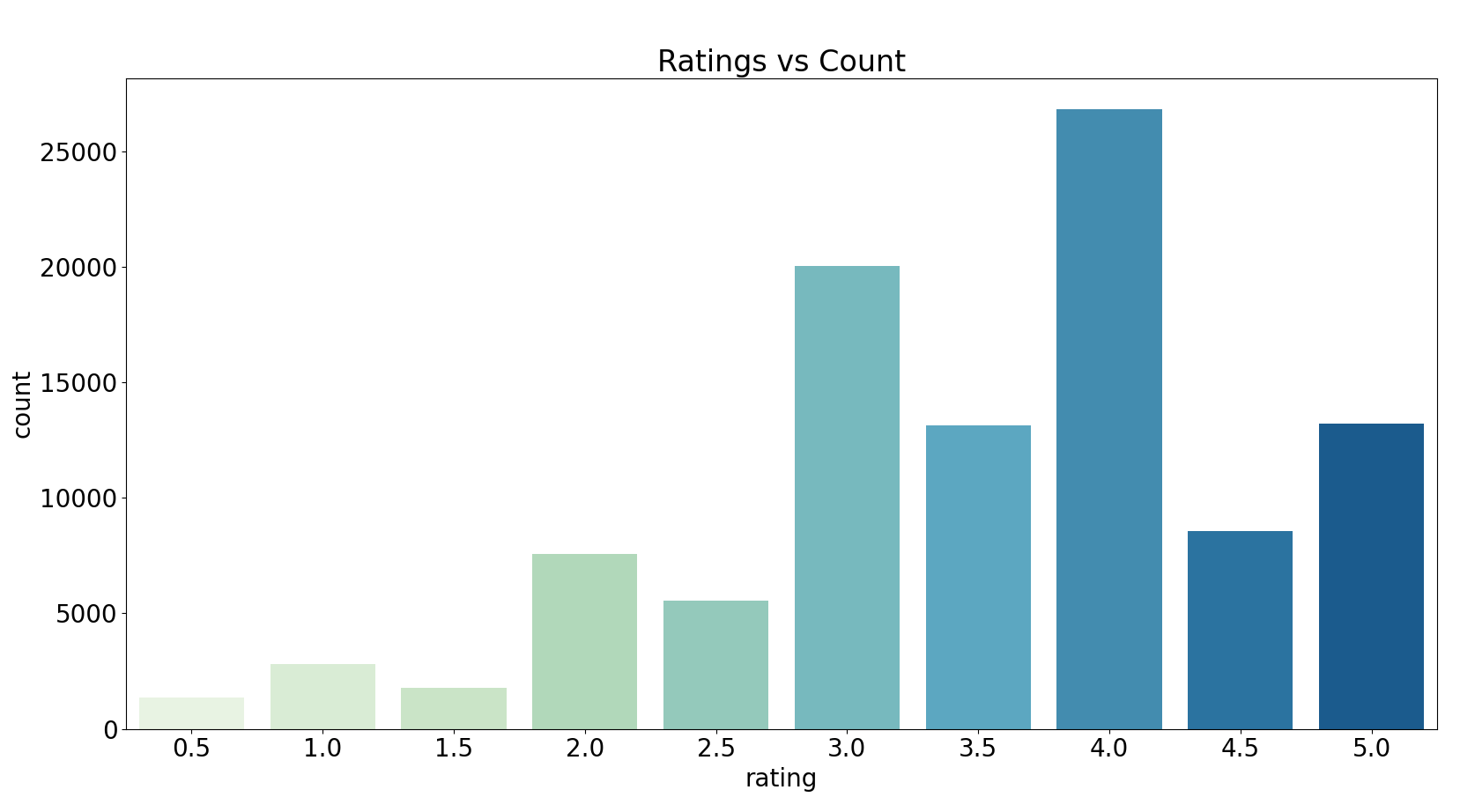
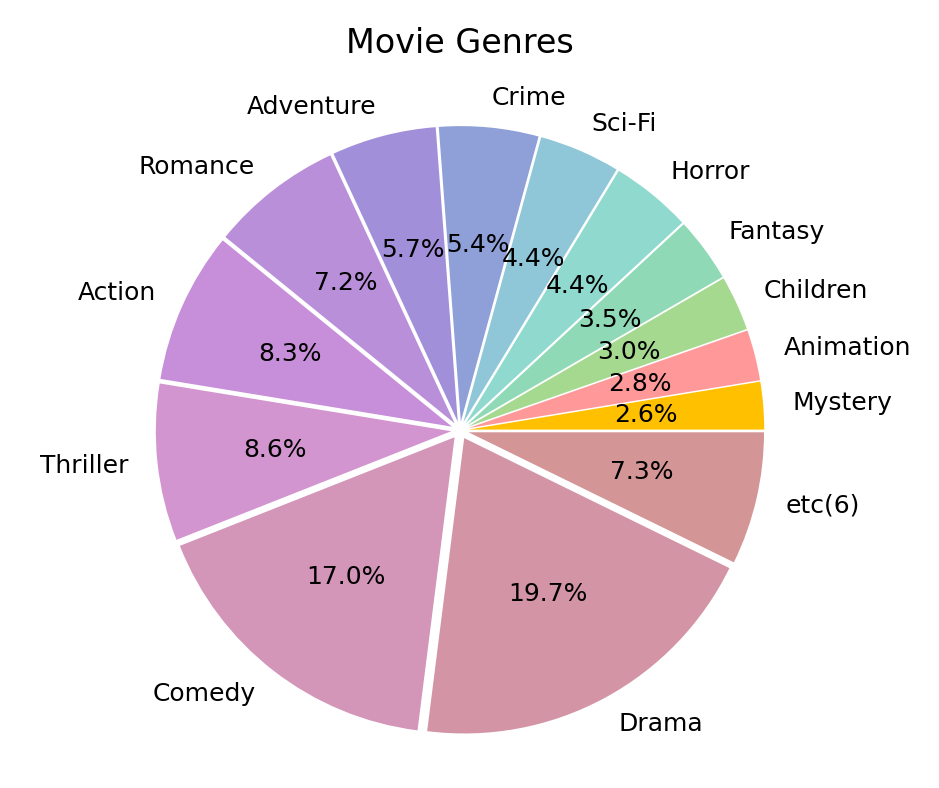
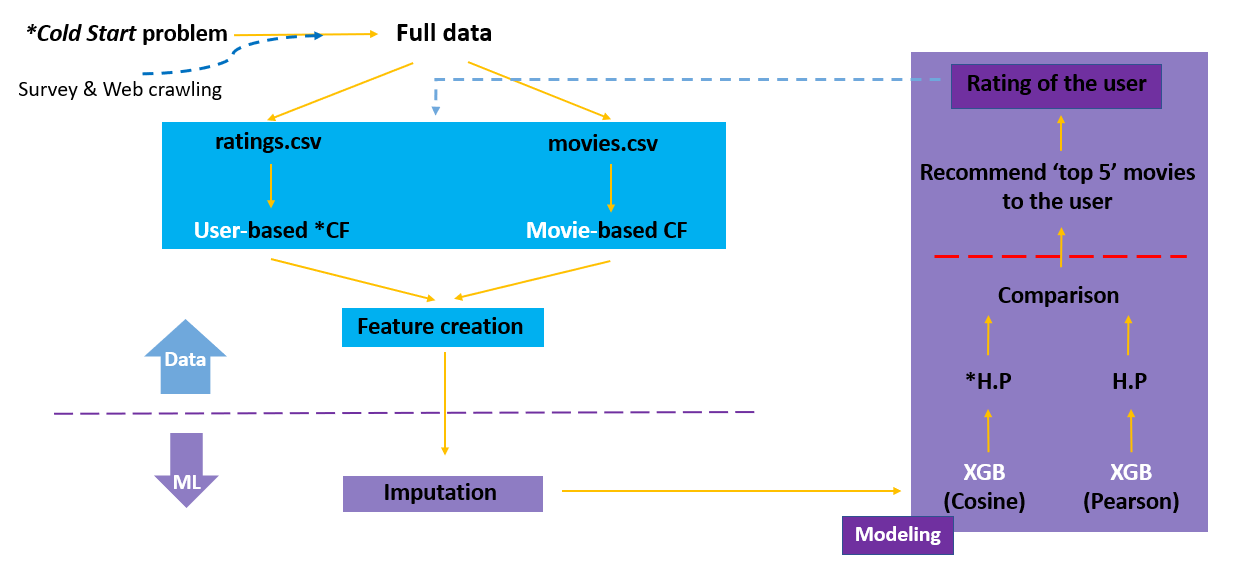
 

FIGURE 1. Left : Distribution of the ratings in rating.csv file. Right : Pie chart of the ratio for movie genres in movies.csv file. These kind of insights on distributions would be helpful when we reflect general popularity on recommendation in face of \**Cold Start problem*[[1]](#footnote-1).

**<System Structure>**



**Data part (blue boxes)**

In terms of real-world application, the system starts with \**Cold Start problem*. The system copes with it through 1) a quick survey asking about movie preferences, such as preferred genres 2) hot – movies recommendation from Box Office rankings through *web-crawling*[[2]](#footnote-2) 3) popularity-based recommendation using others’ data as mentioned like priorly recommending comedy or drama movies in **FIGURE 1**.

After sufficient data(assuming that it’s movielens data) is collected, each ratings data and movies data is used in constructing \**CF*[[3]](#footnote-3)s. In each CFs, similarities between users and between movies are calculated so that the system could recommend movies based on similarity using *cosine similarity*[[4]](#footnote-4) and *PCC(pearson correlation coefficient)*[[5]](#footnote-5).

Based on these CFs, ‘sim\_user1~5’ and ‘sim\_movie1~5’ columns are created because predicting ‘rating’ means that it would be regression problem for ML model.

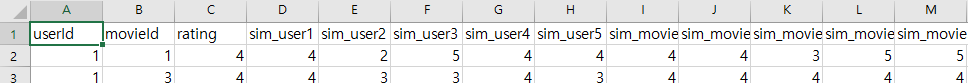


Table 2. Final data - created features(column D~M) are added to ratings.csv (A~C, timestamp column deleted). Sim\_user : Ratings given to this Movie by top 5 similar users with this User. Sim\_movie : Ratings given by this User to top 5 similar movies with this Movie.

**ML part (purple boxes)**

After filling missing values with mean value of the row, XGBoost(Extreme Gradient Boosting) regressor will predict ratings for movies using eight features(‘sim\_user’ & ‘sim\_movie’) – ‘sim\_user1’ and ‘sim\_movie1’ are deleted since their values always mean themselves. XGBoost is one of tree-based ensemble algorithms, which produces multiple decision trees ‘one by one with iterations toward minimized error’(Gradient Boosting). X(Extreme) means it has really short runtime because it limits the maximum depth of trees during each iteration.

To get improved model in terms of prediction error, \*H.P(Hyperparameter Tuning) is conducted against each model using cosine similarity and PCC. Four parameters, well-known for their huge influence on prediction, will be tuned through checking all the 256 combinations (4x4x4x4) and picking the best parameter combination in terms of two evaluation metrices - RMSE(Root Mean Square Error) and MAE(Mean Absolute Error) through 5-fold CV;

Eta(learning rate) : 0.1 / 0.3 / 0.5 / 0.7

Gamma(regularization parameter) : 0 / 1 / 3 / 5

Max\_depth(maximum depth of each added tree) : 4 / 6 / 8 / 10

Min\_child\_weight(minimum sum of weights in child nodes) : 1 / 3 / 5 / 7

Then the best model is chosen (XGB using cosine vs. XGB using PCC) with two performance scores – but this time using train/test validation to figure out the real-world power. After this final comparison, the model predicts ratings for all the movies (regardless of watched or unwatched by the user) then recommends top 5 movies to the user. As a response, the user will rate the movie for real and this data including movie genre will be appended to the previous data again – circular system.

Our works were done within red-dashed line in structure as our work scope and goal is to develop a system. To recommend and get a new user response(rating data) in real world, we should really start our service in web. This is out of our project scope so we didn’t, but these works are sufficient to start the system in web later.

**<WPs>**

**WP1 / Understanding on data and EDAs**

**1.1** - Getting a basic comprehension on Movielens data – how the data were collected and whether there is a special criterion on choosing subjects

**1.2** - Full understanding on all the columns in dataset and conceiving how we can utilize them in unique way

**1.3** - Conduct some EDAs so that we can grasp the distribution of all the columns and correlation between them

**WP2 / Data cleansing and pre-processing**

**2.1** - Merge the given six tables into two data – one is for analyzing user-user correlation, another for analyzing content-content correlation (for **WP 3.1 & 3.2**)

**2.2** - Based on overall apprehension on data from **WP 1.3**, imputation will be carried out here

**2.3** – Collect ‘Box-Office rankings’ to convert them into dataframe as a solution for Cold Start from **WP 2.2** through web crawling so that we can deal with Cold Start based on popularity tactics

**WP3 / Feature creation for performance improvement**

**3.1** - Calculation for ‘movie-movie similarity’ and ‘user-user similarity’, so-called Collaborative Filtering(CF), using cosine similarity and pearson correlation with data from **WP 2.1**

**3.2** - Based on two similarity-calculations above, create a new column such as ‘Ratings given to this Movie by top 5 similar users with this User’ and ‘Ratings given by this User to top 5 similar movies with this Movie’

**3.3**– Delete ‘sim\_movie1’ and ‘sim\_user1’ features so that we don’t count redundant features

**WP4 / Modeling using two similarity calculations**

**4.1** - Hyperparameter tuning (learning rate, min\_child\_weight, max\_depth and gamma) will be conducted against the XGBoost model using cosine similarity (**WP 3.4**) with 5-fold CV

**4.2** - Hyperparameter tuning (on No. of trees, learning rate, ,,,) will be conducted against the XGBoost model using pearson correlation (**WP 3.4**) with 5-fold CV

**4.2** – Choose the best model so far from each **WP 4.1 & 4.2** based on 5-fold CV then compare those two based on train/test validation using RMSE and MAE

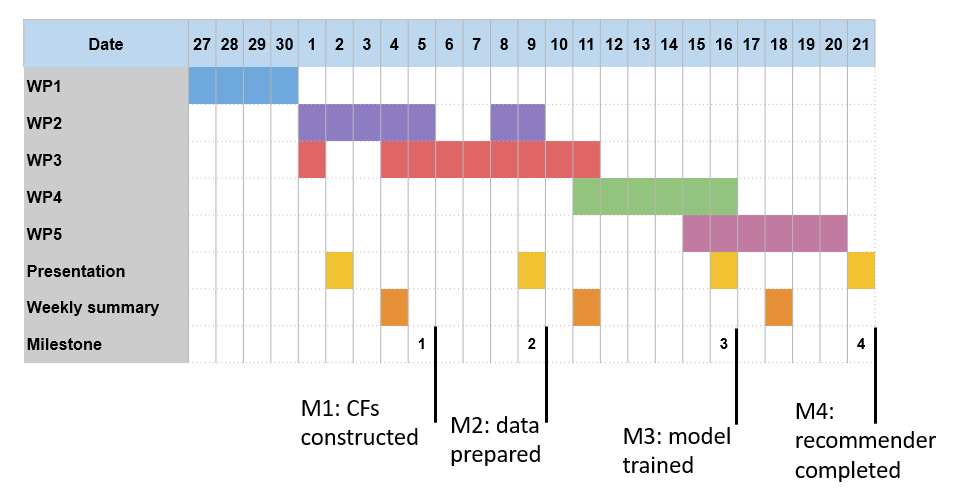
**WP5 / Analysis & evaluation for application in real-world**

**5.1** - All the results from previous steps are visualized into easily understandable format

**5.2** - Evaluation on the final model through traditional ‘test/train’ based on evaluation metrices – RMSE, MAE

**5.3** - Prudent verification of whether the recommendation model will work well in real-world business, including comparison to pre-existing contents recommenders

**<GANTT Chart>**



Start: November 27, 2020

End: December 21, 2020

**<Results>**

In hyperparameter tuning step, we should decide the best one with our brain since computer has no subjectivity to weigh RMSE and MAE. However, to find the best parameter combination among 256 cases in terms of two performance scores is almost impossible for human. Thus, only top 10 (5 for RMSE and others for MAE) for each cosine-similarity-based XGB and PCC-based one are stored into dataframe format in python. All the scores are recorded through 5-fold CV since CV(Cross Validation) can generalize the results more than train/test.

\*E:Eta / G:Gamma / D:max\_Depth / C:min\_Child\_weight

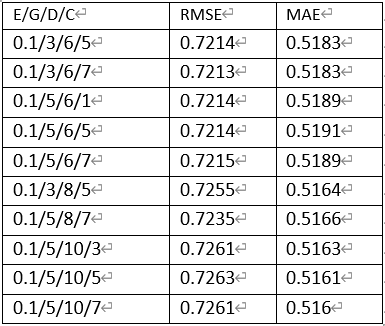
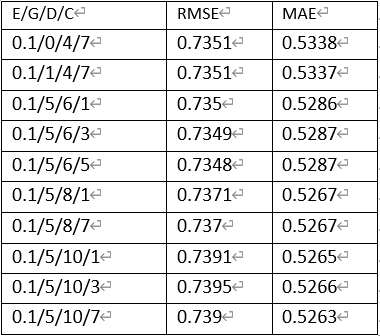


Table 3. Each has 10 rows – upper fives for top 5 lowest RMSE, lower fives for top 5 lowest MAE. Left : Scores for each parameters combination of XGBoost using cosine similarity. Right : Scores for each parameters combination of XGBoost using PCC. Orange-surrounded parameters show the best performance in terms of two scores in each table.

Since the scores in **Table 3** were recorded with 5-fold CV to tune the parameters, the best model of XGB using cosine similarity vs. the best one of XGB using PCC is competed through train/test validation (80:20) to figure out the real-world impact – A SINGLE MATCH.

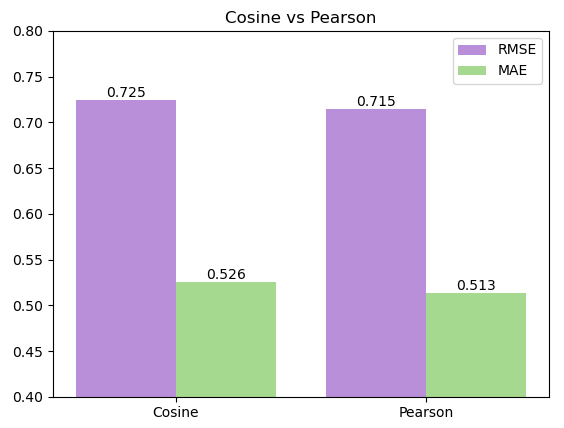


FIGURE 2. A bar chart that shows the train/test result between two XGB models using parameters surrounded by orange square in each table of Table 3. As a result, PCC-based model showed lower RMSE by 0.01 and lower MAE by 0.013 than cosine-based model.

As we can see in **FIGURE 2**, it is clear that PCC-based XGBoost regressor is likely to be better than cosine-similarity-based one in predicting movie ratings of users when real-world application. Therefore, our final model to be used in the system is:

XGBRegressor(eta=0.1, gamma=5, max\_depth=8, min\_child\_weight=7)

To interpret what was going on within our model, we drew a feature importance plot.

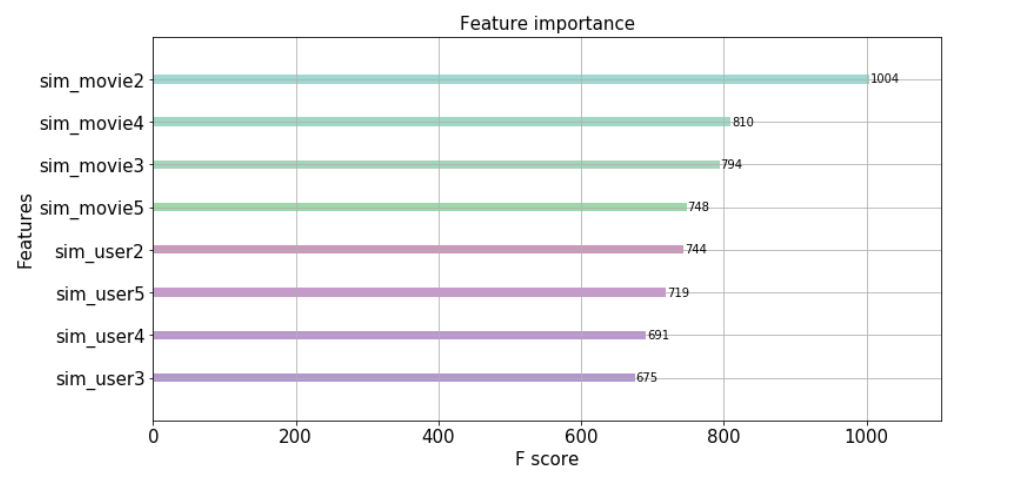


FIGURE 2. The plot shows importance of features used in final model. Although we can expect that the importance rankings will be high in order of 2>3>4>5 since the similarity did follow such order, their rankings, of course, could be easily mixed like this plot since similarity gap can be very different between them for each user.

**<Conclusion>**

Between two similarity calculation methods, PCC was turned out to be better than cosine similarity, showing around 0.01 higher scores in both evaluation metrices. Considering that ratings held place mostly in 4.0, followed by 3.0 & 4.0(refer to **FIGURE 1**), this may mean that calculation method doesn’t much matter – but calculating similarity still does in itself as the model quite a good performance without any features except for 8 similarity-based features.

And the result in **FIGURE 2** can be interpreted in this way; the reason why movie similarity showed higher importance than user similarity is that the number of movies observed in data doesn’t affect how similar movies are, but the number of user’s rating histories does affect in figuring out how similar users are – things could change if with more larger data. Anyway, the PCC-based XGBoost regressor with parameters(eta=0.1, gamma=5, max\_depth=8, min\_child\_weight=7) is likely to predict ratings(0.5~5.0) for all the movies that have their ID in data, within the error of around 0.725.

**< Usage >**

The system has very nice generalizability as movie recommender because it copes with not only recommending movies to normal users, but also to newbie of the platform through being equipped with solutions to *Cold Start*. Also, since the model is dependent on only similarity between users and between contents, our ,,system’’ can be used for other types of video as *Youtube* does, though it is not intended strength. This system also can be used either solely without associated movie-watching platforms or being incorporated in certain platform – but the latter is encouraged so that our circular system reflects newly generated data in time without a hitch.

**<Limits & Future Work>**

Since the study really relied on similarity between users and between movies, the score could get worse or become dependent on movie similarity as mentioned in **Conclusion** when the data is smaller than now. This means it will take quite long for the system to completely overcome *Cold Start* to make somewhat meaningful recommendation. To avoid this, other features such as average ratings or something in the form of time series which are independent from similarity (though we’re not sure whether this would work) can be properly introduced. Plus, since we gave up to carrying out the study with the larger dataset due to runtimes, although we verified the codes have no problem, another way of calculating similarity such as using sparse matrix would be one step of future works.

In addition, by using nDCG(normalized Discounted Cumulative Gain) metric that evaluates how good the model’s movie recommendation is, not how well the model predicted ratings, the system can be improved being equipped with better model in terms of recommending several movie options to users.

**<References>**

Errico, J. H., Sezan, M. I., Borden, G. R., Feather, G. A., & Grover, M. G. (2015). *U.S. Patent No. 8,949,899*. Washington, DC: U.S. Patent and Trademark Office.

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. <https://doi.org/10.1145/2827872>

1. The issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. [↑](#footnote-ref-1)
2. To search the web sites and fetch contents from them using Internet bot. [↑](#footnote-ref-2)
3. *Collaborative Filtering*, a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from users (collaborating). [↑](#footnote-ref-3)
4. The measure of similarity between two non-zero vectors of an inner product space. [↑](#footnote-ref-4)
5. The test statistics that measures the statistical relationship, or association, between two continuous variables. [↑](#footnote-ref-5)