Comparative study of movie recommendation system using feature engineering and improved error function

Dev Kumar
Computer Science and Engineering
NIT Calicut, India
devkumar.dklv@gmail.com

***Abstract*—Movie recommendation systems or Proposal framework suggest movies to users based on their online activities on the site. These systems give ideas to users which are best suited to the user's requirements at a moment when they don't know about it. Recently, these systems become an imperative part in expanding the income for some organizations such as Netflix, web based business destinations and many more. In this paper, we provide comparative study of different machine learning recommendation models which are based on the collaborative and content-based filtering using an effective error function generating relevant features over Netflix dataset. Experiment is performed on the data set comprising over 17K motion pictures and 500K+ clients. Here, Models are compared based on achieved Root Mean Square Error (RMSE) value. The experiment results on the dataset indicates that singular value decomposition provides an effective model that is precise and generates more personalized movie recommendations.**

***Index Terms*—movie recommendation system, supervised machine learning, regression, feature engineering**

I. Introduction

Movie Recommendation system is a machine learning support system which recommend movie to users which are of interest for them based on their perspective. Relevancy of the movies of user's interest is decided and sorted based on generally historically data created when users perform activities online. The recommendation system plays a key role to give important ideas to users/clients by recommending movies which is more relevant based on their needs. Netflix[1] manages a major assortment of TV projects and films, by streaming it whenever by means of on the web. In current scenario, movie recommendation system is an active area of research as its growing demand on practical applications. Many movie recommendation systems have been proposed based on content-based and collaborative filtering [13] [14] [15].

Content-based algorithms recommend movies based on similarity of their attributes. However, collaborative filtering algorithms recommend movies based on history of interactions between users and movies. Despite of these advances in this area, researchers are looking forward for more effective systems which recommends movie to users more precisely.

In this paper, we perform feature engineering to generate thirteen handicrafted features for a standard/traditional XGBoost regressor. In addition, we also generate relevant features using different machine learning models to improve the performance of XGBoost regressor. Here, error functions are also improved as per machine learning models. Further, we compare the performance of different models keeping Surprise BaselineModel as a standard model based on Root Mean Square Error (RMSE) value. The model which produce lower RMSE value is considered as a better recommendation model.

Rest of this paper is organized as follows. Section II presents a brief literature review. Section III presents details about data set and its exploratory analysis. Section IV provides methodology of proposed model. Section V includes results and discussion. Section VI mentions limitations of the work. Finally, conclusion and future direction of research are presented in Section VII.

II. LITERATURE REVIEW

Movie recommendation system play significant role in the user decision making Process to select movies of their own choice especially in increasingly large search space [16]. We can broadly classify movie recommendation approaches into Content-based methods, collaborative filtering methods, and hybrid methods [16]. Content-based Recommendations work based on kinds of movie towards which the user has shown interest in the past. It prepare relevancy list of movies for the user based on text associated with movies which describes their features.

Collaborative filtering (CF) method follows two steps for movie recommendation; first, it gathers the movie ratings or preference of different users and then it suggests movies to different users based on similar history in terms of preference and interests [17]. CF is popularly used for the film recommendation frameworks which relies upon the closest neighborhood mechanism. Breese et al. [3] classify collaborative filtering frameworks into two branches: Memory based collaborative filtering and Model based collaborative filtering.

[1]https://www.netflix.com/

Memory based collaborative filtering works concerning the entire customer space to search for nearest neighbors for a working customer, and as needs be produced a fast outline of proposed movies to recommend. In item based collaborative filtering [1], the relationship with items is set up to shape the neighbors for an objective item. Hybrid methods take advantages of both Collaborative filtering methods and Content-based methods [16].

In this work, we performed comparative study on recommendation systems based on neighborhood methods and matrix factorization-based method, which are collaborative filtering approaches. To improve the performance of XGBoost regressor, additional relevant features generated by machine learning algorithms are also given along with thirteen handcrafted. In addition, error function is also improved to enhance performance of the models.

III. Dataset and Exploratory Data Analysis

A. Dataset DESCRIPTION

Data set taken for experiment is in ".txt files" (combination of four files combined-data-1, combined-data-2, combined-data-3, combined-data-4) [10]. In addition, it have one movie title file named "titles.csv". In each ".txt files", MovieID is followed by a colon and each MovieID contains record of different clients with three features namely customerID, rating, and date. The range of movieIDs are mentioned from 1 to 17770. Range of customerIDs are given from 1 to 2649429. The scale on which ratings are given ranges from 1 to 5 and are integers. The format of dates are YYYY-MM-DD. The data set does not have any missing value and duplicates.

IV. Exploratory Data Analysis

Then the whole dataset (refer Table I) is splitted into training and testing data of which 80 percent of the dataset is used as training dataset and 20 percent is used as testing dataset. Graph between different types of ratings and the number of rating marked by users is plotted from training dataset so that we can observe its distribution over the training dataset (refer Fig. 1). In Fig. 1, we can observe that 4 is the rating given by most of the users(27.5 million approx) and 1 is the rating given by very few users(4 million approx).

TABLE I
Details of the dataset

Total no of ratings :	100480507
Total No of Users :	480189
Total No of movies :	17770

The Fig. 2 indicate a graph with number of rating given by users corresponding to different months of the year. Here, we can observe that there is an exponential growth of number of rating given by users from the year 2000 to mid 2005. After that, there is a steep downfall in the number of ratings. Fig. 3 shows number of ratings by users for movies sorted based on number of ratings. It shows that few of the movies gets number of ratings rated by users in thousands. However,

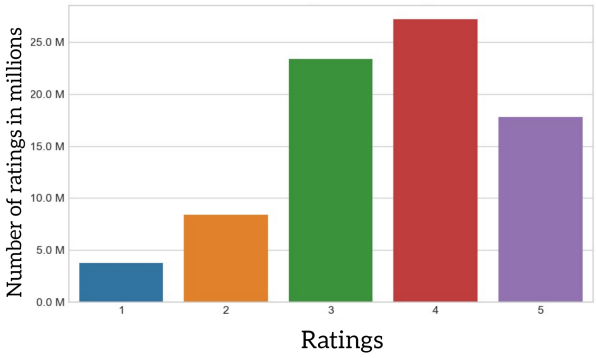

Fig. 1. Distribution of rating over training dataset.

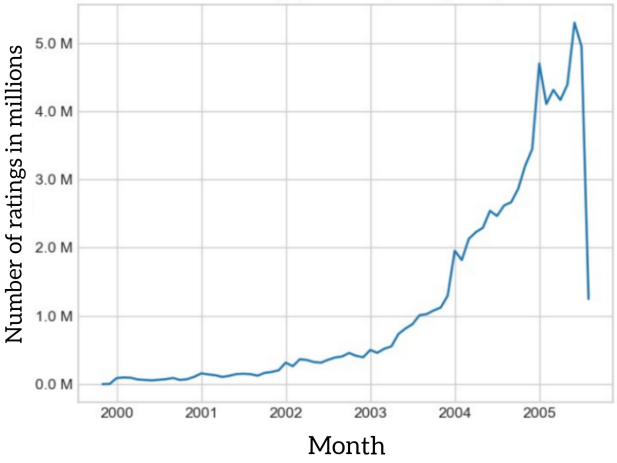

Fig. 2. Number of rating per month on training dataset.

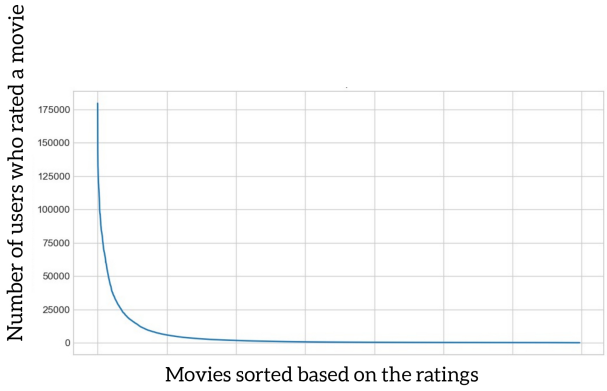

Fig. 3. Number of ratings per movie.

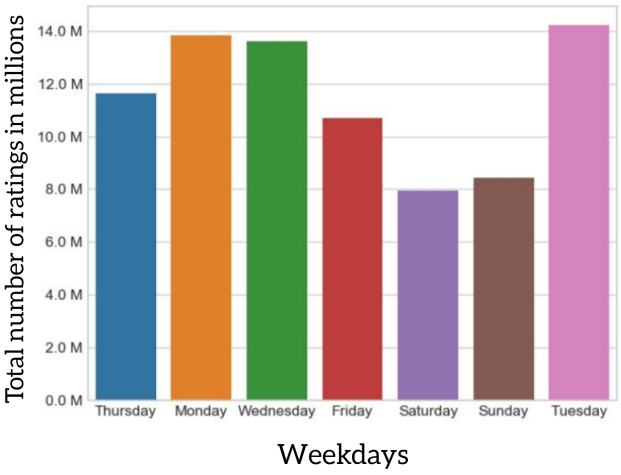

Fig. 4. Number of rating on each day.

the huge number of movies are unrated (not a single user has rated). Fig. 4 shows total number of ratings given by users verses different days. We can observe that Tuesday is the busiest day and Saturday has the least traffic.

In the dataset, we arrange the data instances with respect to time. Further, we split it into training and testing data with ratio 4:1. Here, the statistics of data indicates that the number of users which is not present in our training data but present in our testing data is calculated which is found to be 75,148 which is 15.68 percent of all the users which seems to be huge which can affect our accuracy. Similarly, the number of movies which is not present in our training data but present in our testing data is found to be 346 which is 1.95 percent of all the movies that seems very little to affect our accuracy. Consequently, we may encounter a "cold start problem" [18].

V. Methodology

A. *user-user similarity matrix*

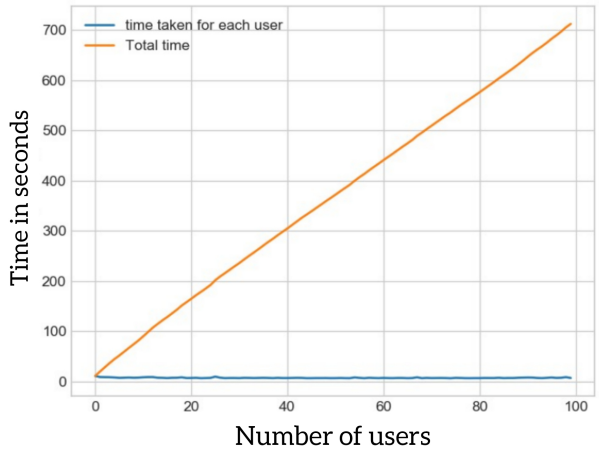

Fig. 5. Computation of similar user for each user

We calculate user-user similarity matrix to generate a feature for XGboost regressor. It is computationally expensive process. It takes 8.8 seconds on average to compute similar users for each user (refer Fig 5) on a high end desktop (i7 processor, 4 core and 32GB RAM). Therefore, for entire 480189 users, it takes around 10.5 days with approximately 82 billion computations to compute the similarity matrix. For faster computation, we keep information as a binary vector of user-user similarity matrix in a dictionary. Whenever it is need we use that computed similarity index.

TABLE II
Top 5 movies similar of movie id 67

Movie id	Year of Release	Title
1	2003.0	Dinosaur Planet
2	2004.0	Isle of Man TT 2004 Review
3	1997.0	Character
4	1994.0	Paula Abdul's Get Up and Dance
5	2004.0	The Rise and Fall of ECW

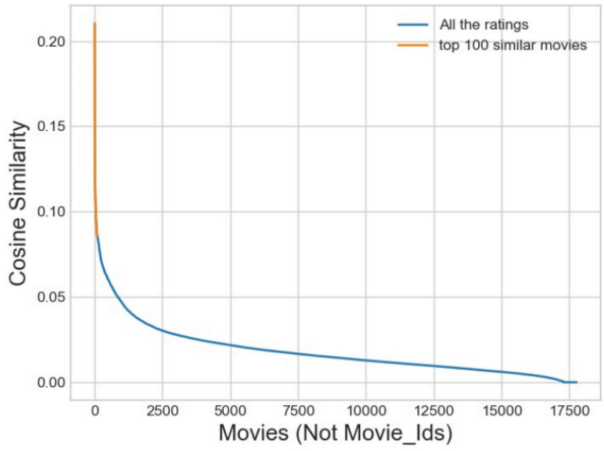

Fig. 6. Computation of movie-movie similarity of movie id 67.

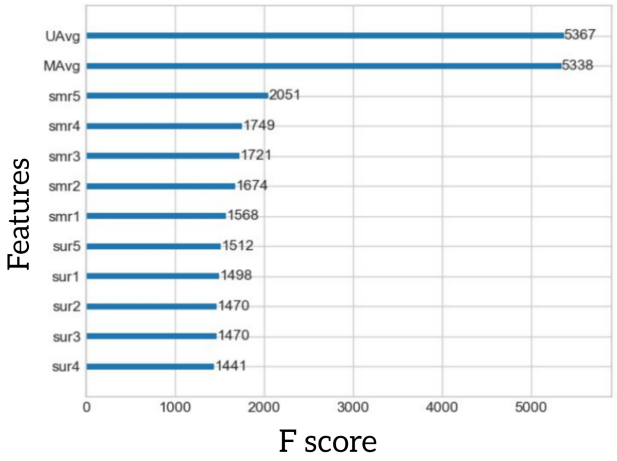

Fig. 7. Features with their feature importance.

B. movie-movie similarity matrix

To calculate the movie-movie similarity matrix for XGboost regressor, cosine similarity is calculated in pair of movies and stored in a matrix. A graph is plotted (refer Fig 6) between cosine similarity of movie id 67 and other movies in the descending order of cosine similarity values. The graph shows that the cosine similarity value reaches below 0.1 after 100 movies and it becomes negligible after 1000 movies and finally converges to zero. To check the performance of cosine similarity for the data chosen dataset, we shows top 5 similar movies to movie id 67 in Table II which indicates that all fetched movies are from generic horror and belong to the same vampire theme with years of release in the 1990's and 2000's.

C. Proposed Design

Initially, we perform feature engineering and derive 13 handcrafted features from the training data sets. First feature is named as average rating of all the movie rating. The second feature is average rating given by a user. The third feature is average rating of that movie. The features from fourth to eight are top five similar users according to similarity index who rated that movie. The features from nine to thirteen are top 5 similar movies according to similarity index rated by that user. Fig. 7 indicates feature importance of twelve features out of thirteen handicrafted features based on F1 score. Here, thirteenth feature is not shown in Fig. 7 as it has comparatively very high value. With these 13 handcrafted features, rating is predicted with the XGBoost regressor (refer Part A of Fig. 8). Here after, the set of thirteen handcrafted features are named as "1" in Fig. 8. In part B of Fig. 8, purpose of different machine learning models is to generate more set of features which help to make prediction more effective by XGBoost regressor. In this regard, we select different machine learning models such as Surprise BaselineModel, Surprise KNN model with user similarity, Surprise KNN model with item similarity, singular value decomposition (SVD), and SVD++ (SVDpp). Surprise[2] is a python library which facilitates various ready-to-use prediction algorithms for designing and analysing recommender systems. The baseline algorithms, neighborhood algorithms (e.g., KNN algorithms), and Matrix Factorization-based Algorithms (e.g., SVD, SVD++) are ready-to-use predictive machine learning algorithms in Surprise library of python. The input of these machine learning models are the training dataset converted into a list of tuples (movie ID, user ID, Rating). In part B of Fig. 8, first machine learning model employed is Surprise BaselineModel. For this model, the error function (refer Eq. 1) is improved based on error function suggested in [13].

$$\hat{r}_{\mathrm{u_i}} = \mu + b_{\mathrm{u}} + b_{\mathrm{i}} \tag{1}$$

$$E = \sum\nolimits_{\mathrm{r_{u_i}\ in\ R_{train}}} (r_{\mathrm{u_i}} - \hat{r}_{\mathrm{u_i}})^2 + \lambda({b_{\mathrm{u}}}^2 + {b_{\mathrm{i}}}^2)[mininize b_{\mathrm{u}}, b_{\mathrm{i}}] \tag{2}$$

where $\mathrm{r_{u_i}}$ is the actual rating given by user $\mathrm{u_i}$, μ is global average of all ratings, $\mathrm{b_u}$ is bias term for user u, $\mathrm{b_i}$ is bias term for movie i, $\hat{r}_{\mathrm{i}}$ is oberserved rating and λ is regularization term for L2 regularization. Here the observed rating is broken into three terms global average, user bias term and movie bias because in collaborative filtering large dataset show a pattern that for some users they rate higher or lower than global average and same for movies to get higher or lower rating

[2] http://surpriselib.com/

than global average rating thus for every iteration the error gets minimised and we get more accurate results on testing data because user and movie behaviour has been taken into account. The output of the model is a predicted rating which is considered as a feature named " 2". Now, feature " 2" along with 13 handcrafted features (now total features is 14 features), in given to XGBoost regressor to perform prediction. Further, we deploy two Surprise KNN Baseline predictor for the training data set. First is with user-user similarity and another one is with item-item similarity which predicts the rating and generate new features "3 " and "4th" based on error functions in Eq 3 and Eq 4 suggested in [12] respectively.

Predicted Rating: (based on User-User similarity)

$$\hat{r}_{u_i} = b_{u_i} + (\sum_{v \text{ in } N^k{}_i(u)} Sim(v,u).(r_{v_i} - b_{u_i})/Z \quad (3)$$

$$\text{where } Z = \sum_{v \text{ in } N^k{}_i(u)} Sim(v,u))$$

b_{u_i} - Baseline prediction of (user,movie) rating.
$N^k{}_i$ (u) - Set of K similar users (neighbours) of user (u) who rated movie(i).
Sim (v,u) - Similarity between users v and u).

Predicted rating : (based on Item Item similarity)[12]

$$\hat{r}_{u_i} = b_{u_i} + (\sum_{j \text{ in } N^k{}_{(u)}i} sim(i,j).(r_{u_j} - b_{u_j})/X \quad (4)$$

$$\text{where } X = \sum_{j \text{ in } N^k{}_{(u)}i} sim(i,j)$$

After training the model with the training data and the predicted rating from these two models are taken as feature 3 and feature 4 and given to our previous XGBoost regressor which now contains 16 features(13 handcrafted features,feature 2, feature 3 and feature 4) and the rating is predicted.In Part-B again the converted training data is given to SVD and SVDpp with predicted rating and error function as follow:-

Prediction ($\hat{r}_{u_i}$) for SVD is set as[11]:-

$$\hat{r}_{u_i} = \mu + b_u + b_i + {q_i}^T p_u \quad (5)$$

q-Represents item(movie) in latent factor space.
p-Represents user in new latent factor space.
${q_i}^T p_u$-Represents movie-user interaction.
Optimization Problem of SVD- Find bu , bi ,qi , pu such that error(E) is minimum and its error function is given as follow[11]:

$$E = \sum_{r_{u_i} \text{ in } R_{train}} (r_{u_i} - \hat{r}_{u_i})^2 + \lambda({b_u}^2 + {b_i}^2 + \|q_i\|^2 + \|p_u\|^2) \quad (6)$$

Prediction $\hat{r}_{u_i}$ for SVDpp with implicit feedback is set as[11]:-

$$\hat{r}_{u_i} = b_i + b_u + \mu + {q_i}^T(|I_u|^{-0.5} \sum_{j \text{ in } I_u Y_j} + p_u) \quad (7)$$

I_u,i_u the set of all items rated by user u, is a length of that set.
Optimization Problem of SVDpp- Find b_u , b_i ,q_i , p_u and y_j such that E is minimum and its error function is given as follow[11]:

$$E = \sum_{r_{u_i} \text{ in } R_{train}} (r_{u_i} - \hat{r}_{u_i})^2 + \lambda({b_u}^2 + {b_i}^2 + \|q_i\|^2 + \|p_u\|^2 + \|y_j\|^2) \quad (8)$$

and the rating is predicted by SVD and SVDpp after training it with the training data and the predicted rating is taken as feature 5 and feature 6 and given to the XGBoost regressor which contains feature 3, feature 4 and 13 handcrafted features and rating is predicted and again feature 5 and feature 6 is given to a XGBoost regressor containing feature 2, feature 3, feature 4 and 13 handcrafted features to make the regressor a total of 18 features and from that 18 features rating is predicted. From all of the above models after predicting the rating mean absolute percentage error and root mean square error is calculated and from these errors models are compared.

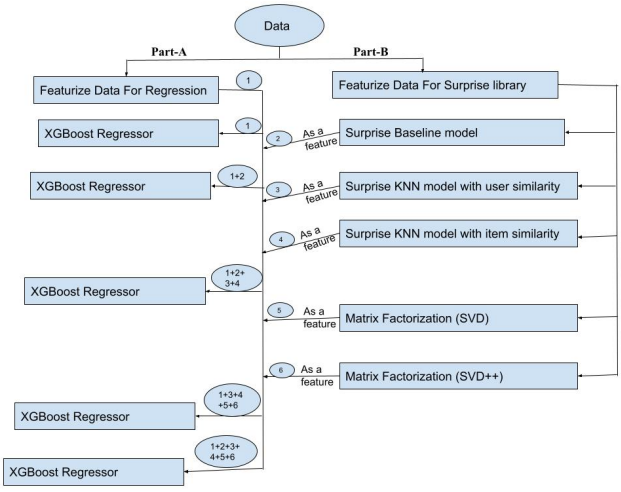

Fig. 8. Proposed Design Model

VI. Results and Discussion

The root mean square error of the above models are given below:

TABLE III
Result Table

Models	RMSE
XGBoost regressor with 13 features	1.1624
Baseline model of Surprise Library	1.0730
XGBoost regressor with 13 features + Baseline model of Surprise Library	1.1048
Surprise KNN model with user-user similarity	1.0726
Surprise KNN model with item-item similarity	1.0728
XGBoost with 13 features + Surprise Baseline model + Surprise KNN model with user-user similarity + Surprise KNN model with item-item similarity	1.2147
SVD	1.0726
SVD++	1.0728
XGBoost with 13 features + Surprise KNN model with user-user similarity + Surprise KNN model with item-item similarity + SVD + SVD plus plus	1.0892
XGBoost with 13 features + Surprise Baseline model + Surprise KNN model with user-user similarity + Surprise KNN model with item-item similarity + SVD + SVD plus plus	1.0753

In our experiment SVD performed best with RMSE 1.072 which is least among all the models and is 7.72 percent better than the basic XGBoost Regressor with 13 handcrafted features. In this experiment we tried to improve the accuracy of XGBoost regressor with 13 handcrafted features and it improved by adding some features which are the predicted rating of models in part B but it performed worst when useless features were added to it. Among all the models SVD performed best because it's error function contains the interaction term($q_i{}^T p_u$) among with its L2 regularization which contains the interaction between a user and a movie uniquely which is not present in other models except SVD++. Here our idea to capture implicit rating hasn't worked because in our dataset, rating given by a user to a movie is a bit different according to user's implicit feedback for many cases therefore RMSE(root mean square error) of SVD++ is 0.0002 more than SVD and SVD++ didn't performed better than SVD.

VII. CONCLUSION AND FUTURE DIRECTION OF RESEARCH

SVD with improved error function has performed 7.72 percent better than the basic machine learning model which is XGBoost regressor with 13 handcrafted features, on the grounds that the less dimensional recommenders are attempting to catch the preference and inclinations, and is realized which we are assuming need to give suggestions dependent on individuals' inclinations, SVD is a decent methodology. In any case, it is additionally realized that this system accomplishes better and more exact outcomes in enormous datasets due to the guess of SVD with the angle plunge. Since we utilized only a part of the dataset , it might very well be the justification that its little improvement in execution contrasted with the XGBoost regressor. Further exploration will be intriguing to think about the models without lessening the informational collection, it will be all the more computationally exorbitant yet we may see various outcomes.

Building a framework that accomplishes great proposals in new clients or cold-start situations is as yet a job to be completed. To make a model with satisfactory outcomes, it could be important to tally with more data, about the client's profile as well as about the films, this could permit us to carry out different approaches.

REFERENCES

[1] B.M. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Item-based collaborative filtering recommendation algorithm," in: Proceedings of the 10th International WWW Conference, Hong Kong, pp. 285–295, 2001.

[2] B.M. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Application of dimensionality reduction in recommender system—a case study," in: Proceedings of ACM WebKDD Workshop, Boston, MA, 2000.

[3] J.S. Breese, D. Heckerman, C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin, USA, pp. 43–52, 1998.

[4] Q. Li, B.M. Kim, "Clustering approach for hybrid recommendation system," in: Proceedings of the International Conference on Web Intelligence, Halifax, Canada, pp. 33–38, 2003.

[5] K. Kim, H. Ahn, "A recommender system using GA K-means clustering in an online shopping market", Expert Syst. Appl. 34 (2) 1200–1209, 2008.

[6] A. Kohrs, B. Merialdo, "Clustering for collaborative filtering applications, in: Proceedings of Computational Intelligence for Modeling," Control and Automation (CIMCA), Vienna: IOS Press, pp. 199–204, 1999.

[7] D. Goldberg, D. Nichols, B.M. Oki, D. Terry, "Using collaborative filtering to weave an information tapestry," Commun. ACM 35 (12) 61–70, 1992.

[8] D. Billsus, M.J. Pazzani, "Learning collaborative information filters," in:Proceedings of the 15th International Conference on Machine Learning, Madison, pp. 46–53, 1998.

[9] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, "Eigentaste: a constant time collaborative filtering algorithm," Inf. Retrieval. 4 (2) 133–151, 2001.

[10] Netflix Prize Data, Kaggle, 2019, [Online]. Available: https://www.kaggle.com/netflix-inc/netflix-prize-data/data.

[11] Yehuda Koren, Robert Bell and Chris Volinsky, "Matrix factorization techniques for recommender systems," Computer ieee journal, Volume 42, Issue 8, pp 30–37, August 2009.

[12] Liang, Xijun et al, "Measure prediction capability of data for collaborative filtering," in: Knowledge and Information System, 2016.

[13] S.S. Choudhury, S.N. Mohanty, and A.K.Jagadev, " Multimodal trust based recommender system with machine learning approaches for movie recommendation," International Journal of Information Technology, 13(2), pp.475-482, 2021.

[14] U. Thakker, R. Patel, and M. Shah, " A comprehensive analysis on movie recommendation system employing collaborative filtering," Multimedia Tools and Applications, pp.1-26, 2021.

[15] V. Verma, and R.K.Aggarwal, "A comparative analysis of similarity measures akin to the Jaccard index in collaborative recommendations: empirical and theoretical perspective," Social Network Analysis and Mining, 10, pp.1-16, 2020.

[16] S. Kulkarni, and S.F. Rodd,"Context Aware Recommendation Systems: A review of the state of the art techniques," Computer Science Review, 37, p.100255, 2020.

[17] R. Ahuja, A. Solanki, and A. Nayyar, "Movie recommender system using K-Means clustering and K-Nearest Neighbor," In 2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence) (pp. 263-268). IEEE. 2019.

[18] S. Gupta, and S.Goel, "Handling user cold start problem in recommender systems using fuzzy clustering," In Information and Communication Technology for Sustainable Development, pp. 143-151, Springer, Singapore, 2018.