Fashionista: A Clothing Application with Collaborative Filtering Based Recommendation System

Sahil Birwadkar[1], Rutuja Firke[2], Vidula Raje[3], Ruchira Vaity[4], Dr. Archana Mire[5]

[1]B.E student, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India

[2]B.E student, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India

[3]B.E student, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India

[4]B.E student, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India

[5]Professor, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India

Abstract *- Recommender systems are used by various e-commerce sites to recommend different products to their customers. The products can be recommended based on the ratings given by the customers and various other implicit parameters, like purchase history or no of times a particular item was viewed. The recommendations in this project will be based on explicit feedback in the form of ratings. We use Matrix Factorization, which is a technique that discovers latent features and complex hidden relationship patterns among users or items with similar preferences or features, given a matrix to decompose. Gradient Descent is used to optimize the cost function. The e-commerce application will enable users to perform all the essential activities like rating, buying or adding a product to the wishlist and cart. Users can browse for items or explore the recommendations made to them.*

Key Words: Recommender System, Matrix Factorization, Gradient Descent, Collaborative Filtering, Mobile Application, e-commerce application

1. INTRODUCTION

E-commerce applications allow to buy and sell products over the internet and globalization has caused a huge hike in the development and usage of such applications. E-commerce platforms today offer an enormous amount of products to users and it would be difficult and inefficient for a user to look through a huge number of products before they find the one they desire. This issue was tackled in 1990 by introducing Recommendation Systems. Recommendation systems analyze implicit and explicit feedback to recommend items to users, helping them to find the right products with ease, which increases the customer satisfaction

In general, the recommender system can be categorized into 3 categories i.e., content-based recommender, collaborative filtering and hybrid recommender.

The content-based recommender system suggests products to users based on user profiles that show user's interests. It takes into consideration item descriptions and user interests to make recommendations. Whereas collaborative filtering uses the historical preferences of users to make recommendations. CF tries to identify users with similar interests or similar products decided based on user ratings. Matrix factorization is one of the prime collaborative filtering techniques that decompose the given ratings matrix into 2 different matrices and finds out hidden latent features which aid in finding patterns.

E-commerce applications these days provide more functionalities than normal buy and browse products. Applications aim to make the whole process more convenient to users by making the UI simple and allowing them to store and organize their preferred products into different lists like wishlists, carts, orders etc.

2. LITERATURE REVIEW

Recommendation systems are widely used in e-commerce systems to increase customer efficiency and hence customer satisfaction. One of the most widely used recommendation approaches in the industry is content-based RS. This type of RS recommends items by matching item description with the user profile created by the system. The user profile may be created using the user's search and purchase history [1]. Another recommendation system that uses a user profile for recommendation is Demographic RS. User profile created in this system is based on age, sex, education, occupation, locality etc instead of search and purchase history. Users are then grouped based on their profiles to make recommendations [2]. Constraint-based RS is another popular RS method which is a type of knowledge-based RS. It takes into consideration the user's preference to make recommendations [3].

Collaborative Filtering is highly accepted as a RS technique for its good prediction performance. J. Leeetal. proposed a collaborative filtering RS to tackle the sparse matrix problem by identifying unrated uninteresting items that are likely to receive low ratings from users, and selectively impute them as low values [4]. Clustering is another technique of CF. V. Subramaniyaswamy et al. proposed an adaptive K-Nearest Neighbour (KNN) based RS through the mining of user preferences [5]. The associative Rules approach is a technique to discover the hidden relationship among items. T. Osadchiyet al. proposed a recommender system based on pair-wise association rules. [6]

Matrix Factorization is one of the latent factor-based models used for more accurate CF recommendations. Koren et al. proposed SVD++, which incorporates implicit feedback to improve the efficiency of traditional SVD used to implement MF in collaborative filtering [7].

To deal with the scarcity of negative feedback in traditional MF technique, M. Li proposed a model named as TimeMF, which is based on inferred feedback and consolidates temporal information[8] Various other algorithms have also been developed based on SVD++, for example, TrustSVD [9], TimeTrustSVD [10], and EnhancedSVD [11] which take into consideration the trust relationship, time information and active learning for matrix completion respectively.

Another paper used Gradient Descent, which is another popular matrix factorization technique, to make predictions for student's performance [12].

3. SYSTEM ARCHITECTURE

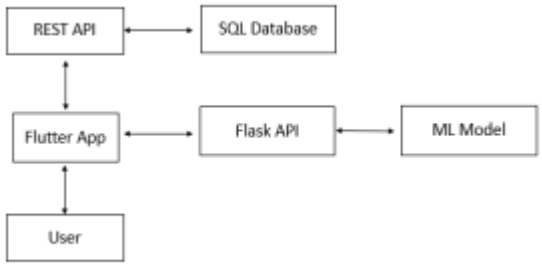

Fig-1: System Architecture

Flutter is an application deployment framework. Flutter SDK is based on Dart programming language which is our technology of choice for this project. This Flutter application needs to interact with the ML model of the Recommendation system and the database containing ratings of the items. This interaction can be handled by using APIs. API is a software intermediary that allows two applications to exchange data between them in the form of request and service. We have used 2 APIs in this project. One API will allow Flutter to connect to our SQL database, and the other will allow the application to get item recommendations from RS.

REST API transfers a representation of the state of the resource to the requester once a request has been made. This representation can be transferred in many formats via HTTP, and for this project, we will use JSON which is the preferred format. REST API is a web API that conforms to the constraints of REST architectural style.

The rating used to train the model, are on a scale of 1-5. Once the user logs in to the system using his credentials, he will be presented with recommended items based on the items previously rated. The predictions from the ML model will be communicated with the application

using a Flask API. Flask API is an API created using the flask framework which implements REST API at its core but allows for comparatively easier interaction with the Machine learning models. The users can then view the items, purchase them, wish list them and all these changes will be communicated with the central database using a REST API.

4. METHODOLOGY AND ALGORITHM

Fig-2: Basic Functioning of Recommender System

4.1. The Dataset

We have used Amazon's Clothing and Fashion dataset for this project. The 'ratings' table used for building the recommendation system has 3 important entities, viz- customer-id, product-id and ratings.

4.2.Pre-processing And Utility Matrix Creation

Pre-processing step mainly deals with analysing the data and getting a utility matrix with cust_id as row indices, item_id and column indices and corresponding ratings as values in each cell.

The ratings were standardized to bring them to the range of 0-1 from their original scale of 1-5.

4.3. Train-Test Split

The data needs to be split to get training and testing dataset. Testing dataset allows us to see how well the user-feature and item-feature matrices predict for the unseen data. For every user we randomly selected 10 ratings given by that particular user as a test set and replaced those values by 0 in the original dataset.

4.3. Matrix Factorization

We will be using the Gradient Descent algorithm to minimize the loss function and factorize the matrices. It divides the given matrix into 2 matrices to reduce the dimensionality, save memory space and retain only the important features of the data.

This optimization technique can work on sparse data, and hence we do not need to impute values, which reduce the possibility of introducing noise to the data.

4.3.1. Gradient Descent

Gradient Descent is an optimization technique that updates the latent features with respect to all users and items to minimize the error.

Updates to the parameters are done after one epoch i.e., after one pass through the entire dataset.

The cost function(J) used for gradient descent optimization along with the regularization term is given below-

$$\min_{Q*,P*} \sum_{(u,i)\in K} (r_{ui} - P_u^T Q_i)^2 + \lambda(||Q_i||^2 + ||P_u||^2) \quad \text{eq. (1)}$$

Where P is latent user feature matrix, Q is latent item feature matrix and rui is rating corresponding to (u, i) pair. Lambda is regularization rate introduced to avoid overfitting the model

The gradient of the above loss function is used to update the P and Q matrices to reduce the error.

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)} \quad \text{eq. (2)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)}. \quad \text{eq. (3)}$$

The update is done as follows-

$$x_k^{(i)} = x_k^{(i)} - \alpha \frac{\partial J}{\partial x_k^{(i)}} \quad \text{eq. (4)}$$

$$\theta_k^{(j)} = \theta_k^{(j)} - \alpha \frac{\partial J}{\partial \theta_k^{(j)}} \quad \text{eq. (5)}$$

Where, alpha is the learning rate, another hyperparameter that needs to be tuned.

Pseudocode of Gradient Descent is-

Input: input matrix, feature count, no of epochs, learning rate, regularization rate

Output: predicted matrix i.e., multiplication of 2 matrices, p and Q

Initialize user-feature(P) and item-feature(Q) matrices;

For every epoch {
    Calculate the gradient of given loss function using formula 2;
    Update P and Q matrices using formula 3;

}

4.4. Making Recommendation

The app will get user-id via a Flask API and find out the unrated items having high predicted rating values for the particular user-id.

4.5. The User Interface

The application provides with variety of functionalities-

1. Login or create new accounts

Fig-3: Login Page

2. Home Page-User can see the recommendations made by the Machine Learning model on this page

Fig-4: Home Page 2

3. View Category wise products

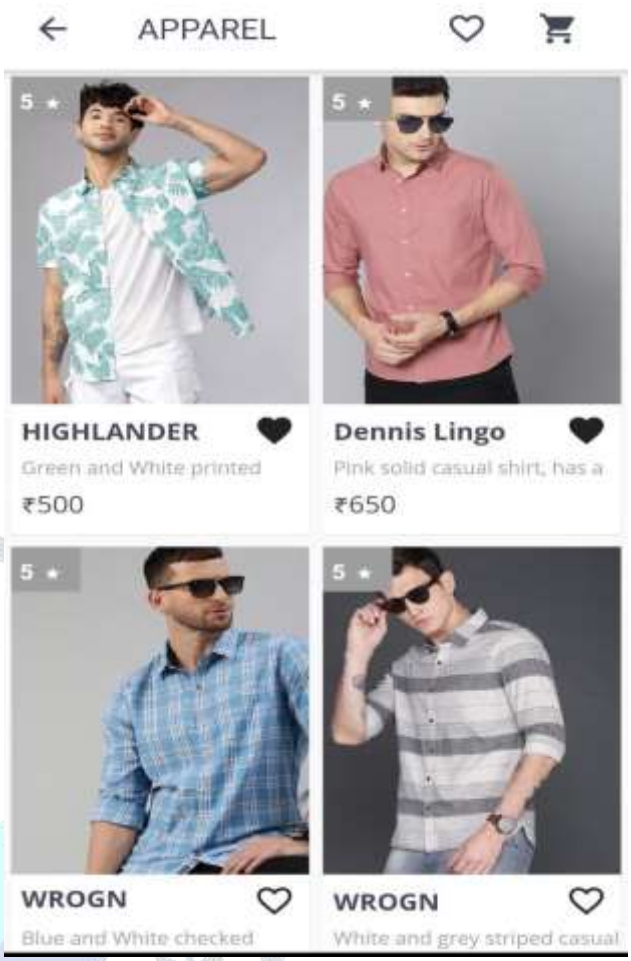

Fig-5: Men Apparel

4: Create and maintain wishlists

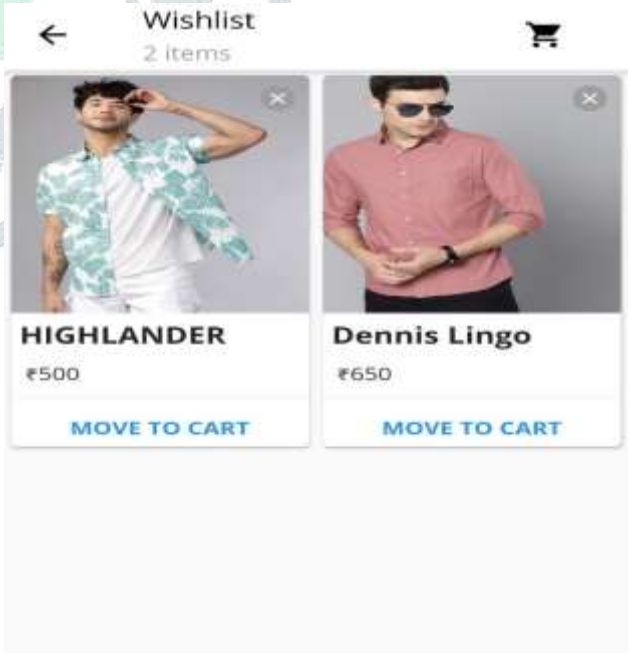

Fig-6: Wishlist

5: Create and view carts

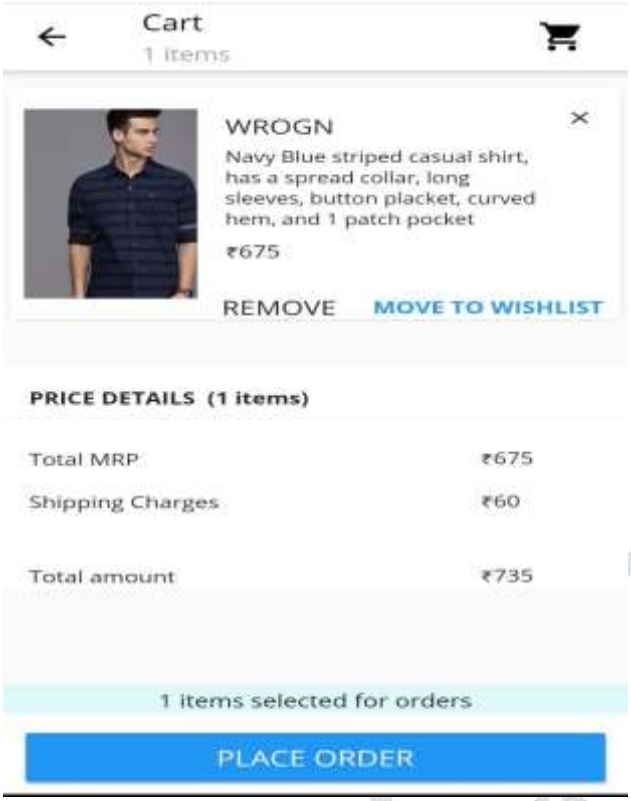

Fig-7: Cart

6: Manage the details of the account and user from the Profile page

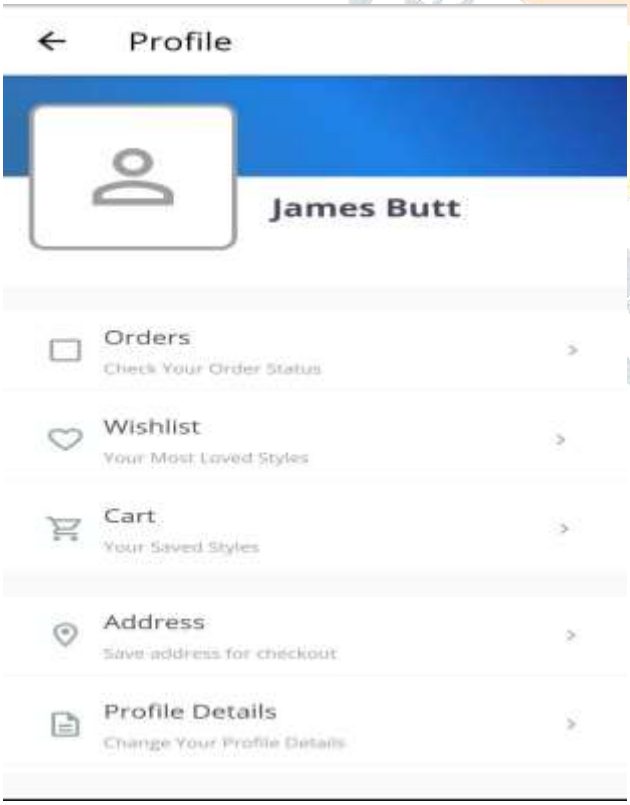

Fig-8: Profile Page

7: View all previous orders at one glance

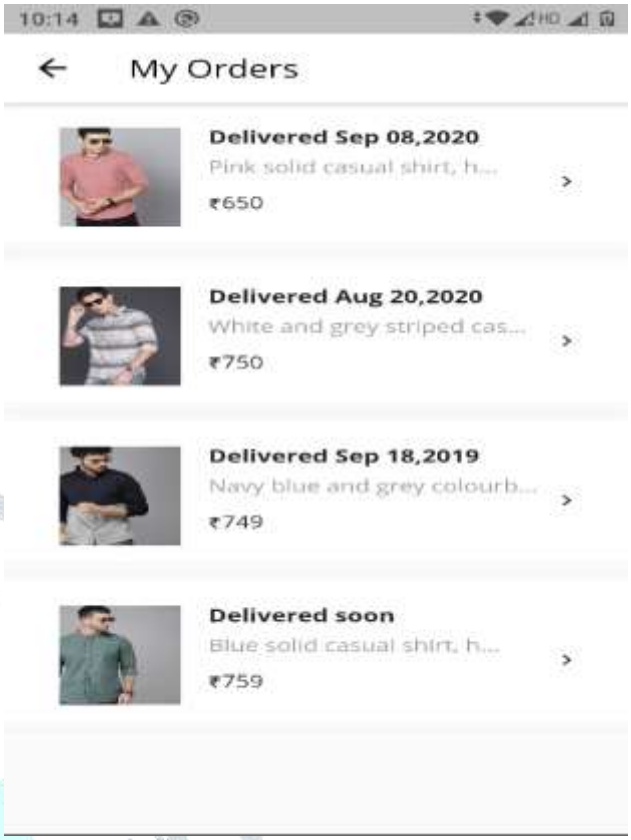

Fig-9: My Orders Page

8: View orders' details

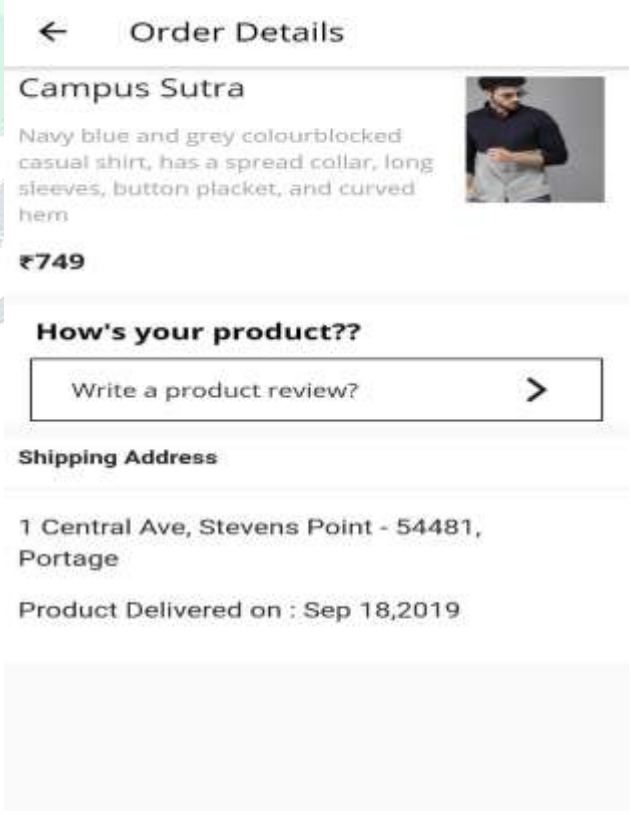

Fig-10: Order Details Page

9: Rate items and write reviews

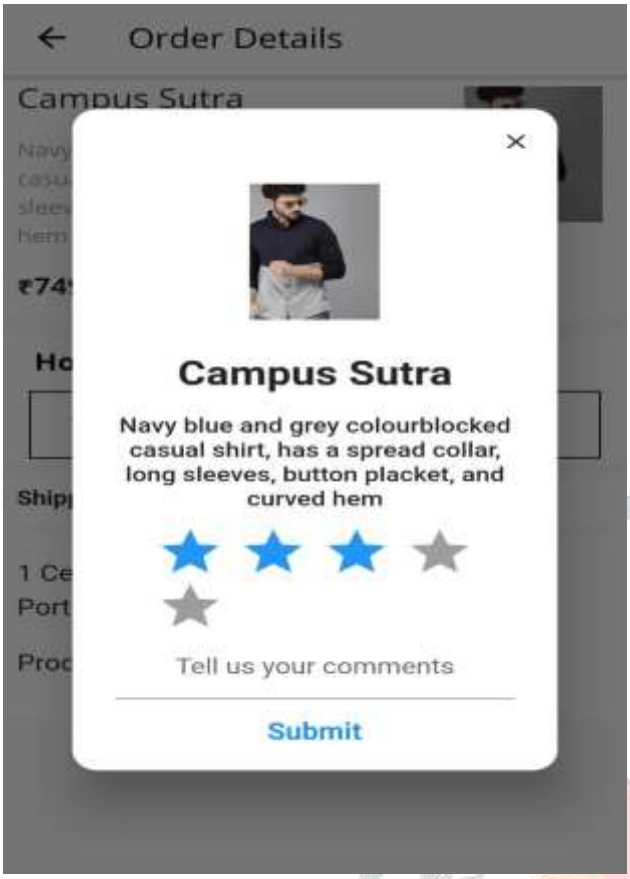

Fig-11: Rating Pop-up page

5. RESULTS

The RMSE measure is used to estimate the performance of the system. It is calculated after every epoch and was observed to have steadily reduced. We calculate the predicted ratings from user-feature and item-feature matrices updated after every epoch and consider only the fields that have been rated by the users for calculating the RMSE and checking the accuracy of the system.

The hyperparameters were optimized using Grid Search technique which allows us to try and fit the model to all possible combinations of hyperparameter values provided by us. The optimal number of features that efficiently describe the items and user tastes, for the given dataset was found to be 10. Following table depicts different RMSE values obtained for different no of features considered.

Sr. No.	No of Feature	RMSE
1	8	0.1628
2	10	0.1621
3	20	0.1625
4	30	0.1635

Table-1: Change in RMSE w.r.t. no. of features

The RMSE value was the lowest when 10 features were considered. The following graph shows how error reduced with no of epoch for the test dataset as well as the training dataset for 10 features.

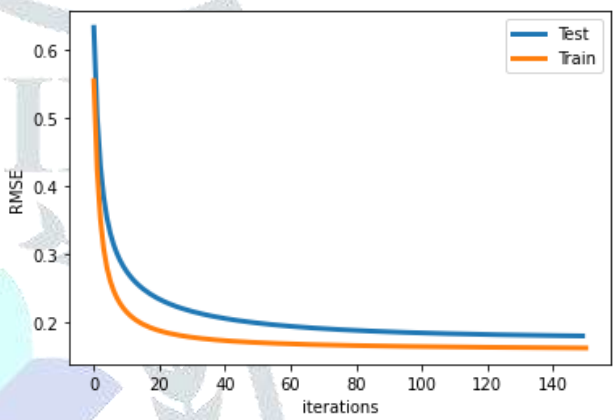

Chart-1: Plot of RMSE v/s no. of iterations

After around 100 iterations, the RMSE does not decrease any further and hence it gives us the optimal state with minimum error.

The integration of the Flutter application with the database and Recommendation system helped us provide recommendations based on user's tastes and previous ratings provided by them.

6. CONCLUSION

Gradient descent technique provides faster results avoids noise introduction in the ratings matrix. This algorithm allows direct convergence and avoids local minima.

The biggest issue with this collaborative filtering approach is cold start problem. Accurate recommendations cannot be made to users with less to no interaction with this system. This issue can be dealt with by building a hybrid system. Introducing content-based recommendations along with collaborative filtering can help mitigate the new item issue and also help find similar items. This system would also benefit

from collecting demographic information on users and tracking various user actions like, items viewed, or items added to wishlist. Furthermore, since the recommendations in this system are based on past preferences alone, there is no novelty is recommendation. This can be fixed by making serendipitous recommendations.

The limitations of current system can be overcome by combining it with any or all of these techniques.

Companies across many different areas of the enterprise are beginning to implement recommendation systems to improve their customer's online purchasing experience, increase sales, and retain customers by focusing on customer satisfaction.

REFERENCES

1. M. J. Pazzani and D. Billsus, "Content-based Recommendation Systems," in The Adaptive Web, Berlin, Heidelberg, Springer-Verlag, 2007, pp. 325-341
2. Al-Shamri, Mohammad Yahya H.. "User profiling approaches for demographic recommender systems." Knowl. Based Syst. 100 (2016): 175-187.
3. A. Felfernig and R. Burke, "Constraint-based Recommender Systems: Technologies and Research Issues," in 10th international conference on Electronic commerce (ICEC '08), Innsbruck, Austria, 2008
4. Lee, J., Hwang, W., Parc, J., Lee, Y., Kim, S., Lee, D.: l-injection: Toward effective collaborative filtering using uninteresting items. IEEE Transactions on Knowledge and Data Engineering 31(1), 3–16 (Jan 2019)
5. Subramaniyaswamy, V., Logesh, R.: Adaptive knn based recommender system through mining of user preferences. Wireless Personal Communications 97(2), 2229–2247 (Nov 2017)
6. Osadchiy, T., Poliakov, I., Olivier, P., Rowland, M., Foster, E.: Recommender system based on pairwise association rules. Expert Systems with Applications 115, 535 – 542 (2019)
7. Y. Koren, R. Bell and C. Volinsky, "Matrix factorization techniques for recommender systems", *Computer*, vol. 1, no. 8, pp. 30-37, Aug. 2009
8. M. Li, H. Wu and H. Zhang, "Matrix Factorization for Personalized Recommendation With Implicit Feedback and Temporal Information in Social Ecommerce Networks," in IEEE Access, vol. 7, pp. 141268-141276, 2019, doi: 10.1109/ACCESS.2019.2943959.
9. Koren, "Collaborative filtering with temporal dynamics", Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, pp. 447-456, Jun./Jul. 2009
10. C. Tong, J. Qi, Y. Lian, J. Niu and J. J. Rodrigues, "TimeTrustSVD: A collaborative filtering model integrating time trust and rating information", Future Gener. Comput. Syst., vol. 93, pp. 933-941, Apr. 2017

    X. Guan, C.-T. Li and Y. Guan, "Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems", IEEE Access, vol. 5, pp. 27668-27678, 2017
11. X. Guan, C.-T. Li and Y. Guan, "Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems", IEEE Access, vol. 5, pp. 27668-27678, 2017
12. J. A. Gómez-Pulido, A. Durán-Domínguez, and F. Pajuelo-Holguera, "Optimizing Latent Factors and Collaborative Filtering for Students' Performance Prediction," Applied Sciences, vol. 10, no. 16, p. 5601, Aug. 2020