**Introduction:**

Recommendation systems aim to use machine learning technique or algorithms to narrow down choices and provide user with service recommendation, which plays an important role in our everyday lives and is widely used in various fields such as social media and online shopping [[1]](#endnote-1). Today, almost any platform can utilize a recommendation system to study user behavior and predict responses to other products. Netflix, for instance, uses a recommendation system to suggest movies to specific users based on user data. Therefor, we choose to study recommendation systems.

The working principle of recommendation is using suitable algorithms to process collected data and make prediction. This includes collaborate approach (used in this project), content-based approach, and hybrid filtering approach. So far, most researchers use machine learning(ML) algorithms to study recommendation system, such as regression[[2]](#endnote-2), k-nearest neighbor[[3]](#endnote-3), and clustering[[4]](#endnote-4).

This project aims to explore and compare performance multiple machine learning algorithms in building a recommendation system. The biggest challenge of a recommendation system is the data sparsity. Returning to the Netflix example, it is highly unlikely that a user has watched, let alone rated, every single movie on the platform. The dataset for studying in this project is the MovieLens, found at [[5]](#endnote-5), which is a very famous recommendation system dataset. The data sparsity can be at the range of lower than 4%, which poses a big challenge in making prediction with high accuracy. We first use singular value decomposition[[6]](#endnote-6) (SVD) to learn the ratings of the dataset, which can fill in the Nan values in the dataset. Our second method is to study the metadata including the occupation, age, gender, and genres of the movies using different kinds of regression algorithm. At last, we combine the ratings (by SVD method) and the metadata for the learning process. Note that in our application, classification may not be the best approach as we aim to predict the rating. We use a regression methods and evaluate performance of different algorithm using mean absolute error (MAE). In all, we studied and compared the performance of different machine learning algorithms for recommendation system, which can provide a good reference for researchers who work in this field.

**Literature Review:**

The techniques used for recommendation system includes collaborate approach (used in this project), content-based approach, and hybrid filtering approach. [Isinkaye, F.O., Folajimi, Y.O. and Ojokoh, B.A., 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), pp.261-273. ]

Content-based filtering focus on analyzing user profiles and make prediction for this user. For instance, recommend a movie based on the city, country, and other movies that the user watched before. Content-based filtering does not need the data of other user but one needs to have a depth knowledge of the user profile, which might be different in some cases.

Collaborative filtering predicts objects to a users according to the analysis of all the previous users communally. This includes memory based and model based techniques such as clustering, decision tree, regression, and matrix completion technique, SVD, for instance. Collaborative filtering can outperform Content-based techniques when we do not have much knowledge about the profile of the items. However, Collaborative filtering also suffers from disadvantages such as Cold-start problem [Burke R. Web recommender systems. In: Brusilovsky P, Kobsa A,Nejdl W, editors. The Adaptive Web, LNCS 4321. BerlinHeidelberg (Germany): Springer; 2007. p. 377–408], Data sparsity problem, and Scalability.

Hybrid filtering combines the above two methods to improve the performance of the recommendations system.

In this project, we adopted Collaborative filtering to study the dataset. The first thing to consider is choosing our loss function and performance metric. Our initial approach was treating this as a classification problem, adopting 0-1 loss and measuring our performance using CCR. However, such a method would consider it wrong if a ground truth label is 3 and our prediction was 3.5, even though such a case is relatively correct. For this, we decided to use mean absolute error in measuring performance and treat this as a regression problem with the following constraint. Since ratings are only allowed in 0.5 increments from 0.5 to 5, we will be rounding our predictions to the nearest 0.5.

We approached the problem of recommendation system on the MovieLens dataset in 3 ways. The first is a very simple approach where we consider just the user ratings to build a model that recommends movies. Here we plan to build a synthetic dataset and use algorithms like correlation, Matrix Factorization to predict ratings and recommend movies and then apply the algorithm on MovieLens 1M. The second approach would be to consider the metadata of the users to group them into clusters and use that information to recommend movies. Finally, the third approach would be the combination of the first two, where we use both movie ratings and user metadata to predict ratings of movies that the user has not watch and recommend movies.

Correlation: Calculating the correlation coefficient might be the easiest to study the relation between different movies. However, this method fails to provide informative results in our dataset. (1). Most of movie are not rated (rating count is low), which means sparsity will be a problem during the next step. (2) Due to the sparsity of the data, the result of process two vectors will be zero and thus most of the correlation will be zeros. (3) This correlation only considers the rating of all users and fails to consider the preference of each user.

Matrix Factorization using SVD: One of the main things to consider when building this dataset is that although the matrices involved in recommender systems are extremely large, their rank is usually relatively small. It is for this reason that many people suggest the implementation of matrix factorization and reconstruction on such problems. As shown in [[7]](#endnote-7), a matrix (R) can be deconstructed into smaller matrices which can then be multiplied together reproduce the original matrix.

As discussed in [3], the intuition behind the deconstruction of our matrix is as follows: This deconstruction of the ratings matrix R, will yield some insight into user preference and movie features, U and V respectively. The matrix is diagonal matrix which will then somehow weight different users and movies accordingly to restore the original ratings.

**Reference:**

1. Portugal, I., Alencar, P. and Cowan, D., 2018. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, *97*, pp.205-227. [↑](#endnote-ref-1)
2. Purushotham, S., Liu, Y. and Kuo, C.C.J., 2012. Collaborative topic regression with social matrix factorization for recommendation systems. *arXiv preprint arXiv:1206.4684*. [↑](#endnote-ref-2)
3. Adeniyi, D.A., Wei, Z. and Yongquan, Y., 2016. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. *Applied Computing and Informatics*, *12*(1), pp.90-108. [↑](#endnote-ref-3)
4. Nilashi, M., Bagherifard, K., Rahmani, M. and Rafe, V., 2017. A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques. *Computers & industrial engineering*, *109*, pp.357-368. [↑](#endnote-ref-4)
5. GroupLens. (2019). *MovieLens*. [online] Available at: https://grouplens.org/datasets/movielens/ [Accessed 26 Oct. 2019]. [↑](#endnote-ref-5)
6. Hsu, D., Kakade, S.M. and Zhang, T., 2011. Robust matrix decomposition with sparse corruptions. *IEEE Transactions on Information Theory*, *57*(11), pp.7221-7234. [↑](#endnote-ref-6)
7. nick becker. (2019). Matrix Factorization for Movie Recommendations in Python. [online] Available at: https://beckernick.github.io/matrix-factorization-recommender. [↑](#endnote-ref-7)