Movie Rating Predictions with Bayesian SVD, Latent Factor, GLocal-K, SVM, FastFM and DeepFM approaches

Name of author

University of California San Diego
Nov 30, 2022
email@address

Abstract

In this paper, the team conducts predictive analysis on MovieLens 1M datasets to compare the performances of different prediction models, such as Bayesian Singular Value Decomposition, Latent Factor, GLocal-K, Support Vector Machine(SVM), FastFM, and DeepFM models. The team starts with an Exploratory Data Analysis (EDA) to better understand the properties & relationships between each features and maximize the datasets' insights. Based on previous dataset observations, the team then implements various predicting models. To provide further conclusions, the teams display MSE values of each model with different hyperparameters through different visualization techniques such as histograms and bar graphs.

Keywords: Rating Prediction, Movielens, Recommender Systems

1. Dataset

1.1. Introduction

The MovieLens Datasets were initially collected and released by a research group at the University of Minnesota in 1998[12]. It takes participants' expressed preferences for movies to give personalized movie recommendations. The MovieLens Datasets contain five different versions, which are "MovieLens 100K," "MovieLens 1M," "MovieLens 10M," "MovieLens 20M," and "MovieLens 25M." All datasets contain information about movies and ratings, which is joined by the attribute "MovieIDs." Additionally, the 1M and 100K datasets contain users' demographic data, such as age, gender, occupation, and zip code. The team decided to use the "MovieLens 1M" Dataset because the team would like to explore the relationship, or lack of, between demographic information and movie ratings. The team chose the 1M datasets over the 100K one because it provided larger data for further analysis.

1.2. Structure

The MovieLens 1M Dataset contains three sets of data, which are rating, user, and movie. The ratings.dat includes ratings for each movie[12]. The movies.dat contains the following core attributes:

- MovieID: a unique numerical identifier of the rated movie
- MovieTitle: a unique name of the rated movie with the release year in parentheses
- MovieGenres: a list of genres to which the rated movie belongs to

The users.dat contains the following core attributes:

- UserID: a unique numerical identifier of the user who provides ratings
- UserAge: the bucketized age values of the user who provides ratings
- Gender: the gender of the user who provides ratings. M represents male and F represents female.
- Occupation: the occupation of the user who provides ratings
- Zipcode: the physical location of the user who provides ratings

1.3. Exploratory Data Analysis

By observing the "MovieLens 1M" Dataset, the team discovered some patterns among attributes, which can be used as potential features to predict ratings.

- Movie Genres: there are 18 different genres among all movies. The number of Drama and Comedy movies is much higher than the others. If participants are more likely to rate certain types of movies, their ratings are more likely to be affected.[Appendix A.2]
- Movie Release Year: the dataset contains movies that were released from 1900 to 2000, but the majority were released between 1990 and 2000. The release year of a movie might affect ratings.[Appendix A.5]
- Gender: it is observed in the dataset that the number of male participants is twice higher than the number of female participants. Therefore, it is reasonable that we take gender differences into feature consideration.[Appendix A.3]
- Age: participants in the dataset vary in age. The team observed that participants aged between 25-34 years old are more active in rating movies.[Appendix A.1]
- Occupation: people with different jobs might have different movie preferences. The team can further explore whether there are connections between occupations and movie ratings.[Appendix A.4]
- Location(state): although participants are located in different states, most participants live in California. Therefore, it is important to consider the physical location of the participants.[Appendix A.6, A.7]

2. Movies Prediction

2.1. Predictive Task

For the predictive task, the team decided to evaluate the performance of Bayesian Singular Value Decomposition, Latent Factor, GLocal-K, FastFM, Support Vector Machine(SVM), and

DeepFM models for predicting movie ratings. To better understand the connections between various features and movie ratings, the team first developed a baseline model with various demographic features, which include age, gender, occupation, and zip code. Then the team calculated each model's mean square error (MSE) and compared the results with the baseline model for further analysis.

2.2. Baseline Model

The team started with determining which demographic features could be used potentially to build the baseline model. First, the team created helper functions, including MSE and Jaccard, to calculate the mean square error of the prediction results. Since the team is still in the early developmental stage of the baseline model, and it is time-consuming to run all models on the 1M datasets, the team decided to run baseline models on the 100K datasets and discover whether there were any significant findings.

Then the team began to evaluate the performance of the baseline models using the user and movie as features. For each movie rating record in the dataset, the team used the Jaccard function to calculate the similarity between every movie the users had rated. The team assigned weights to the similarity value and the average rating of that movie to predict the movie rating. As shown in Figure 1, the team achieved an MSE of 0.98758 on the valid set and 0.97086 on the test set.

The team wanted to improve the initial MSE by solving the "cold-start" issue. The "cold-start" issue occurred when the movies were in the valid and the test set but not observed in the training set. It is essential to solving the "cold-start" issue because the MSE on the cold-start part was 2.00292, which was fairly high.

Therefore, the team modified the baseline model by taking demographic information such as age, location, and occupation into consideration to solve the cold-start problem. The team created a new function to integrate all demographic information into weighted prediction. By calculating each demographic feature's similarity, the team expected a lower MSE value. However, the rating performance on the entire dataset worsened, which might be caused by equally assigned weight on features.

Therefore, the team decided to zoom into every demographic feature and tried to find the dominating feature in rating prediction. It was observed that occupation features played a significant role in rating prediction. The team then applied the same method to calculate the similarity of occupations and found the MSE decreased to 0.94402 on the valid set and 0.92920 on the test set. It can be concluded that demographic features have influences on rating prediction. Therefore, it is significant to implement demographic features in other prediction models.

Inspired by lecture materials, the team applied sentiment analysis on the titles of movies instead of review texts and calculated the similarity between titles and other features to predict ratings. However, there is no major improvement in the baseline model using sentiment analysis.

The team then repeated the analysis on the 1M datasets and discovered that adding demographic features such as age, gender, occupation, and zip code didn't improve the performance of the prediction. This may be because when the data increases, there are more variances between participants, while the number of features for predictions is limited. Therefore, with a larger dataset, it is more reasonable to use the baseline with user, movie, and rating features. The team then decided on the baseline with an MSE of 0.79991 on the valid set and 0.79696 on the test set.

Feature	Dataset	MSE - valid	MSE - test
user, movie, and rating	100K	0.98758	0.97086
user, movie, and rating, and demongraphic information	100K	1.00824	1.01899
user, movie, and rating, and occupation	100K	0.94402	0.92920
user, movie, and rating	1M	0.79991	0.79696
user, movie, and rating, and demongraphic information	1M	0.95923	0.95955
user, movie, and rating, and occupation	1M	0.82675	0.82569
user, movie, and rating, and sentiment analysis on movie titles	100K	1.01325	1.02235
user, movie, and rating, and sentiment analysis on movie titles	1M	1.00787	1.01924

Figure 1: *MSE values for baseline models*

3. Model

3.1. Definition

- Bayesian Singular Value Decomposition: This model is a Bayesian treatment of the traditional SVD model by adding priors to the precisions of noise and the feature vectors. This model is more robust than a simple normal assumption [1].
- Latent Factor: Latent Factor model is a state-of-the-art methodology for model-based collaborative filtering. The basic assumption is that there exists an unknown low-dimensional representation of users and items where user-item affinity can be modeled accurately [2].
- Support Vector Machine: A support vector machine (SVM) model is a machine learning algorithm that uses supervised learning models to solve complex classification, regression, and outlier detection problems by performing optimal data transformations that determine boundaries between data points based on predefined classes, labels, or outputs [3].
- GLocal-K: GLocal-K model applies two types of kernels in two stages, respectively: pretraining (with the local kernelized weight matrix) and fine-tuning (with the global-kernel based matrix) [4].
- FastFM: Factorization Machine implementation (fastFM) provides easy access to many solvers and supports regression, classification and ranking tasks. Such an implementation simplifies the use of FM for a wide range of applications [5].
- DeepFM: Deep FM is a model that combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. It is equipped with a shared input to its "wide" and "deep" parts, with no need for feature engineering besides raw features [6].

3.2. Models

The team explored various approaches (e.g., latent factor model and FastFM, AFM) for solving the problem before finalizing GLobal-K and DeepFM. Since the latent factor model and GLobal-K are both based on Matrix Completion, and FastFM and DeepFM are both based on Factorization models.

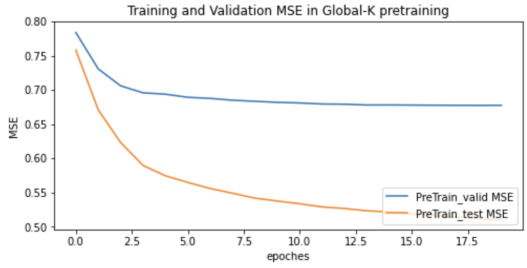


Figure 2: *Global-K pretrain mse vs epoch*

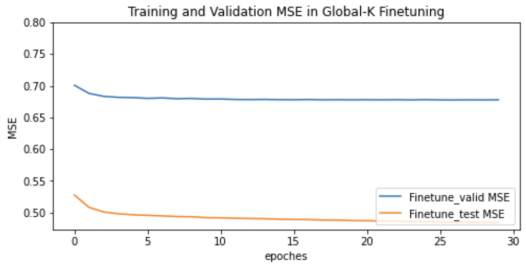


Figure 3: *Global-K Finetune mse vs epoch*

3.2.1. FastFM

Since there are connections between features, it is significant to study the relationship between them for better rating predictions. Since features in this dataset are sparse, the matrices used to represent data are also sparse. Since the Factorization Machines mechanism is suitable for handling sparse matrices, the team applied the FastFM model to predict ratings.

3.2.2. GLocal-K

It has been discovered that calculating sparse matrices is time-consuming. Therefore, the team decided to introduce the Global Local Kernel-based matrix completion framework to generalize and represent a high-dimensional sparse user-item matrix entry into a low-dimensional space with a small number of essential features, which amplifies the connections between features. The group pretrained the model in figure 2 and finetuned it in figure 3.

3.2.3. DeepFM

Deep FM combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. It's a more complex model to capture the relationship between features.[figure 4]

3.3. Model Optimization

3.3.1. Hyperparameter Finetuning

For FastFM model, the team finetuned the l2-reg-V(L2 penalty weight for pairwise coefficients), l2-reg-w(L2 penalty weight for linear coefficients) in Figure 5.

For AFM model, the team finetuned the attention factor(positive integer, units in attention net) and l2-reg-linear(L2 penalty weight for linear coefficients) in figure 6.

For DeepFM model, the team finetuned epoch, dnn-dropout(the probability we will drop out a given DNN coordinate), l2-reg-linear, l2-reg-embedding (L2 regularizer strength applied to embedding vector) in figure 7.

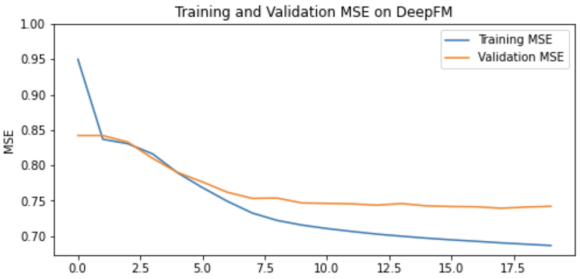


Figure 4: *DeepFM mse vs epoch*

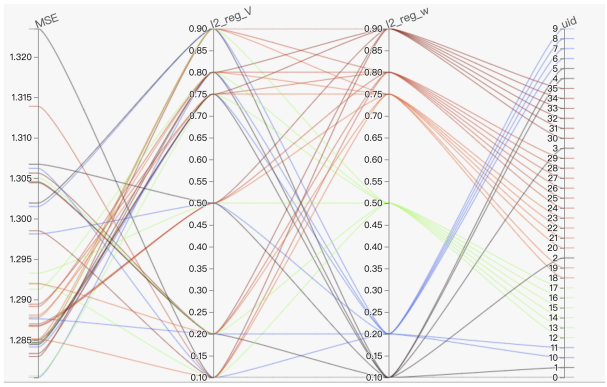


Figure 5: *Hyperparameters finetune of FastFM.*

3.3.2. Features building

The group has turned zip code into state, timestamp into month, Genres into Genres-text and tried different features combination, and the features are in figure 8. And the team finally built the one-hot coding features based on "MovieID", "UserID","Hour", "Gender", "Age", "Occupation", "State","And genre-text" to improve the FastFM, DeepFM, which performs best.

3.3.3. Attention mechanism

The pattern between the movies and users is so complicated that filling the matrix in a simple way would lead to poor performance. Since the attention mechanism has been introduced to neural network modeling, it has been widely used in many tasks. The group wanted to employ the attention mechanism on feature interactions by performing a weighted sum on the interacted vectors which have been done in AFM(Attentional Factorization Machines)[7].

3.3.4. Problems in Optimization

For hyperparameters finetuning, when the epochs come to 50, the DeepFM model becomes overfit(figure 9). In the training of model AFM, when training MSE decreases dramatically, the valid MSE remains unchanged. For feature building mechanism, word similarity has been used between titles as the features, and maybe since it is not a sparse feature, the attempt is unsuccessful.

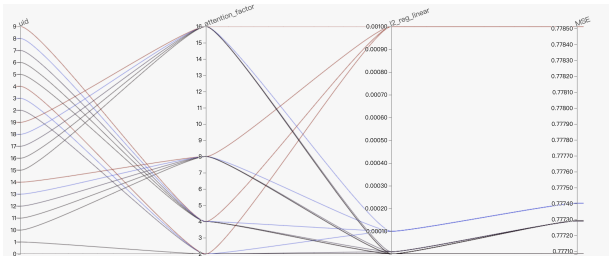


Figure 6: *Hyperparameters finetune of AFM.*

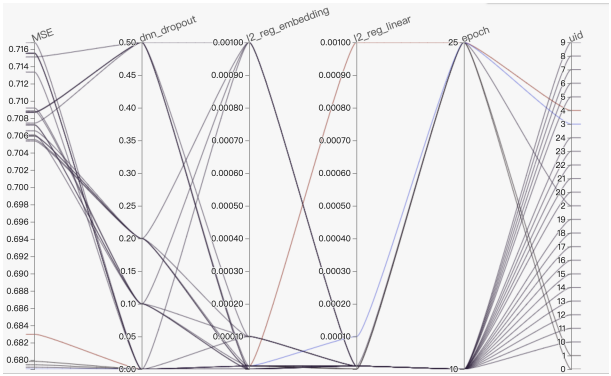


Figure 7: *Hyperparameters finetune of DeepFM.*

3.4. Models Comparision

The team has also used SVD, SVD++, SVM for comparision which both works pretty good on most tasks and also tried to use the word similarity between titles as the features. And maybe since it is not a sparse feature, the attempt is not successful.

After calculating the MSE values for all models mentioned above in Table1, the team discovered that the DeeFM and GLocal-K approaches perform the best. In contrast, the FastFM method has the worst performance. Therefore, the team decided to compare some strengths and weaknesses of models with distinct characteristics.

Overall, the FastFM method has the worst performance and it is not the most time-efficient. Therefore, it can be concluded that the FastFM model might not be suitable for this dataset. The GLocal-K and DeepFM have the best performance however, they are also the most time-consuming models. In addition, GLocal-K performs better on larger datasets. When taking both time efficiency and overall performance into consideration, BayesSVD and BayesSVD++ has relatively good performance and is the fastest model.

When considering the size of datasets, the deep learning model with its complicated structure performs better on the bigger dataset than the traditional methods. And both methods perform better on the bigger dataset since the model can get more information.

	UserID	MovieID	Rating	Month	Hour	Gender	Age	State	Occupation	Genres	sim_title_Word2Vec	Genres_text
246667	1490	648	4	11	1	M	50	CA	7	[Action, Adventure, Mystery]	0.434298	Action Adventure Mystery
449484	2771	47	4	11	1	M	18	CA	1	[Crime, Thriller]	0.630584	Crime Thriller
750684	4478	2385	2	3	21	M	1	MA	12	[Comedy, Romance]	0.177307	Comedy Romance
935407	5643	1758	4	5	6	F	35	UT	1	[Drama]	0.808181	Drama
447463	2756	2028	5	11	19	M	18	OR	4	[Action, Drama, War]	0.849127	Action Drama War

Figure 8: *df-rating features*

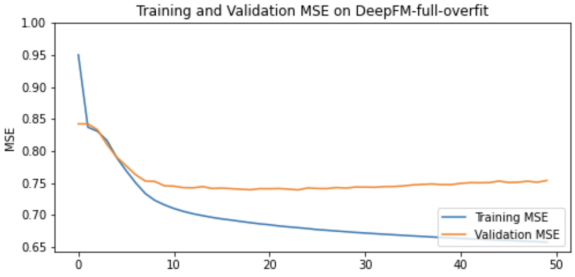


Figure 9: *overfit : DeepFM mse vs epoch.*

4. Literature Review

4.1. Realated Literature

Recommender systems work on user-item interaction matrices, and these matrices are typically sparse with very high dimensionality. Matrix completion is complex for predicting one's interest based on millions of other users having each seen a small subset of thousands of items [4]. Particularly, we studied rating prediction approaches of recommender systems.

In rating prediction scenario, many researchers have achieved promising performance in rating prediction domain by generalizing matrix factorization to neural collaborative filtering [8]. Previous studies also proposed a deep learning model to learn features by employing denoising autoencoders to encode bag-of-words representations of the descriptive text of items [9]. Those research have been exploited to produce more precise latent features for users and items to predict ratings.

Recently, many variants based on Factorization Machine(FM) have been proposed and achieved promising performance. Factorization Machine has been proposed to solve the feature combination problem of sparse data scenarios and is widely adopted in advertising and the recommender system area [10]. Neural Factorization Machine(NFM) seamlessly combines the linearity of FM in modeling second-order feature interactions and the non-linearity of neural network in modeling higher-order feature interactions [11]. A generic feature-based recommendation model has been developed and a convergent updating rule has been proposed to resolve the challenging discrete optimization of DFM [12].

4.2. MovieLens: Where and How

The team performed a movie rating prediction task, and there were many papers focusing on similar topics. The team used an existing dataset called MovieLens and studied this dataset(conducted many experiments) and many of its variants(ML-100K, ML-1M, etc.).

The MovieLens datasets documented participants' preferences for movies which were collected and generated through a recommender system and then provided personalized recommen-

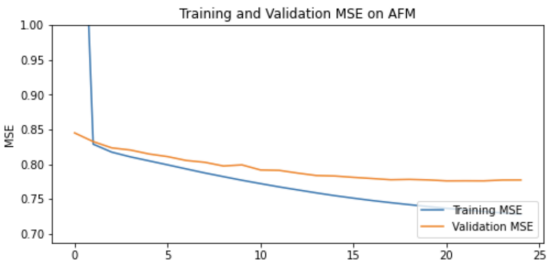


Figure 10: *DeepFM mse vs epoch*

dations to participants [13]. ML-100K, 1M, 10M, and 20M are the product of participants interacting with MovieLens recommender system.

GroupLens Research Group in the University of Minnesota developed the datasets and maintained that website originally as an alternative to a commercial movie recommender, EachMovie. By running this live online system for decades, the research group had great flexibility when designing various experiments and validating key elements for better recommendation performance.

Before the release of v4, the rating/recommendation cycle is enabled by the collaborative filtering (CF) algorithm. The movies that appear first in search results for a user are those that the algorithm predicts that the user will rate the highest [13]. And v4 used a content-driven algorithm(using tags) rather than ratings-based techniques.

4.3. Similar Dataset

Some similar datasets collected from previous research are listed below. These datasets share similar features, such as ratings, users, and items, and are commonly used to perform rating prediction tasks. These datasets differ from MovieLens in size, shape, and interaction context.

- Yelp: This dataset includes 8,021,122 reviews from 209,393 businesses in 10 metropolitan areas. These data are structured in JSON files, including business, review, user, check-in, tip, and photo data. This dataset can be used in different ways, such as sentiment analysis and 5-star rating classification. It focuses on rating prediction for restaurants based only on their review texts. Multiple feature generation methods, everal machine learning models, and some deep learning models were used to analyze this dataset. [14].
- Amazon Product: The ratings in this dataset contain product views from May 1996 to July 2014 from Amazon. It is one of the largest datasets, which contains 82 million ratings from 21 million users across 10 million items. This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs), and the ratings are in the range of 1 to 5 [15].
- Netflix Prize: Netflix dataset was part of the open competition for the best algorithm to predict user ratings for films. It included data from 480,189 users, 17,770 movies, and about 100,000,000 ratings. Each rating in the training dataset consists of four entries: user, movie, date of grade, and grade. Users and movies are represented with integer IDs, while ratings range from 1 to 5 [16].
- Book-Crossing: The BookCrossing (BX) dataset was collected by Cai-Nicolas Ziegler in a 4-week crawl (August / September 2004) from the Book-Crossing community with kind permission from Ron Hornbaker, CTO of Humankind Systems. It contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books [17].

4.4. SOTA Methods

Many research groups tested their rating prediction models on MovieLens datasets. And some existing works have achieved very promising results in terms of MSE metrics. Some reimplemented state-of-the-art methods and improved versions with their MSE results can be seen in Table 1, and a brief description of selected SOTA methods are listed as follows:

- GLocal-K: GLocal-K can be divided into two stages: first is pre-training an auto encoder with the local kernelized weight matrix, which transforms the data from one space into the feature space by using a 2d-RBF kernel. Then, the pre-trained auto encoder is fine-tuned with the rating matrix, produced by a convolution-based global kernel, which captures the characteristics of each item [4].
- CF-NADE: CF-NADE is proposed as a neural autoregressive architecture for collaborative filtering (CF) tasks, which is inspired by the Restricted Boltzmann Machine (RBM) based CF model and the Neural Autoregressive Distribution Estimator (NADE). It can be extended to a deep model with only moderately increased computational complexity. Experimental results show that CF-NADE is one of the state-of-the-art methods with a single hidden layer on MovieLens datasets, and performance can be further improved by adding more hidden layers [18].
- Bayesian SVD++: It has been shown that results for baselines that have been used in numerous publications for the Movielens 10M benchmark are suboptimal. And with a careful setup of a vanilla matrix factorization baseline, the reported results can be improved upon for this baseline and even outperform the reported results of newly proposed method [19].

4.5. Comparison with Existing Conclusions

The team reimplemented some SOTA methods and made some improvements by incorporating features selected through feature-building optimization. Compared to previous literature, the team's findings confirmed the solidity of existing state-of-the-art methods with respect to the MSE metrics.

5. Conclusions

5.1. Result & Analysis

Among all models used to predict ratings on the MovieLens Dataset, the FastFM model has the worst performance based on MSE value, and it is not a relatively time-efficient model.

Table 1: *MSE of models*

Model	ML-100K	ML-1M
BayesSVD	0.786	0.743
BayesSVD++	0.782	0.740
GLobal-K	–	0.668
SVM	–	1.023
FastFM-Basic	0.910	0.832
FastFM-Feature	1.191	1.178
AFM-Basic	–	0.778
AFM-Feature	–	0.729
DeepFM-Basic	0.729	0.709
DeepFM-Feature	0.689	0.675

To improve the model, AFM was introduced and got a slight improvement in the performance of prediction while keeping its time-saving characteristics. In contrast, the GLocal-K and DeepFM models can significantly improve the rating performance. Although they are both time-consuming models, they have the lowest MSE values among all models. When the volume of data is large enough, the performance of GLocal-K is better than the DeepFM model. However, with limited data, the performance of GLocal-K would drop dramatically. DeepFM, on the other hand, has a stable high performance regardless of the size of the datasets by using the feature-building method, which includes integrating UserID, MovieID, gender, occupation, age, state, and genres of the movie as features into the prediction implementation. When considering both time efficiency and overall performance, BayesSVD has a relatively good performance and is the fastest model.

Clearly, some models didn't meet the team's expectations in rating predictions because they couldn't properly capture the relationship between features, such as FastFM, Latent Factor, and SVM.

The team also explored and applied many optimization mechanisms, such as feature-building, hyperparameter tuning, and attention mechanisms. Those mechanisms enhanced the performance of all prediction models and offered the team new insights into rating prediction.

5.2. Feature Representation

As mentioned in the baseline model section, the following features are chosen to use for rating prediction.

MovieID: int
UserID: int
Gender: Category
Age: int
Occupation: int
State: Category
Genre: Category

5.3. Parameter Selection

The team only used the feature mentioned above in implementing various models. Other attributes in the MovieLens dataset, such as timestamps and movie titles, are redundant for implementations because they have limited effects on the prediction. For instance, movie titles can be fully substituted with movie id, and the timestamps only indicate when participants rated movies rather than when they watched movies. Therefore these attributes can't reflect the rating properly.

5.4. Significance

This research not only offers the team an opportunity to get hands-on experience with existing deep-learning techniques for rating predictions but also provides some insights into choosing the most suitable model under different circumstances. Based on the analysis of MovieLens datasets, when time is sufficient and the dataset is small, it is recommended to use the DeepFM model. When time is sufficient and the dataset is large, it is recommended to use the GLocal-K model. When considering both time efficiency and performance, using the Bayesian SVD model is recommended.

6. References

[1] C. Luo, Y. Xiang, B. Zhang, and Q. Fang, "A bayesian treatment for singular value decomposition," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, 2015, pp. 1761–1767.

[2] J. Fang, X. Zhang, Y. Hu, Y. Xu, M. Yang, and J. Liu, "Probabilistic latent factor model for collaborative filtering with bayesian inference," *CoRR*, vol. abs/2012.03433, 2020. [Online]. Available: https://arxiv.org/abs/2012.03433

[3] V. Kanade, "What is a support vector machine? working, types, and examples," Sep 2022. [Online]. Available: https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine

[4] S. C. Han, T. Lim, S. Long, B. Burgstaller, and J. Poon, "Glocal-k: Global and local kernels for recommender systems," in *Proceedings of the 30th ACM International Conference on Information Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3063–3067. [Online]. Available: https://doi.org/10.1145/3459637.3482112

[5] I. Bayer, "fastfm: A library for factorization machines," *Journal of Machine Learning Research*, vol. 17, no. 184, pp. 1–5, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-355.html

[6] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 1725–1731.

[7] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," 2017. [Online]. Available: https://arxiv.org/abs/1708.04617

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," *CoRR*, vol. abs/1708.05031, 2017. [Online]. Available: http://arxiv.org/abs/1708.05031

[9] H. Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems," *CoRR*, vol. abs/1409.2944, 2014. [Online]. Available: http://arxiv.org/abs/1409.2944

[10] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 995–1000.

[11] X. He and T. Chua, "Neural factorization machines for sparse predictive analytics," *CoRR*, vol. abs/1708.05027, 2017. [Online]. Available: http://arxiv.org/abs/1708.05027

[12] H. Liu, X. He, F. Feng, L. Nie, R. Liu, and H. Zhang, "Discrete factorization machines for fast feature-based recommendation," *CoRR*, vol. abs/1805.02232, 2018. [Online]. Available: http://arxiv.org/abs/1805.02232

[13] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, dec 2015. [Online]. Available: https://doi.org/10.1145/2827872

[14] Z. Liu, "Yelp review rating prediction: Machine learning and deep learning models," *CoRR*, vol. abs/2012.06690, 2020. [Online]. Available: https://arxiv.org/abs/2012.06690

[15] J. McAuley, "Amazon product data." [Online]. Available: https://jmcauley.ucsd.edu/data/amazon/

[16] N. Netflix, "Netflix prize data," Nov 2019. [Online]. Available: https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

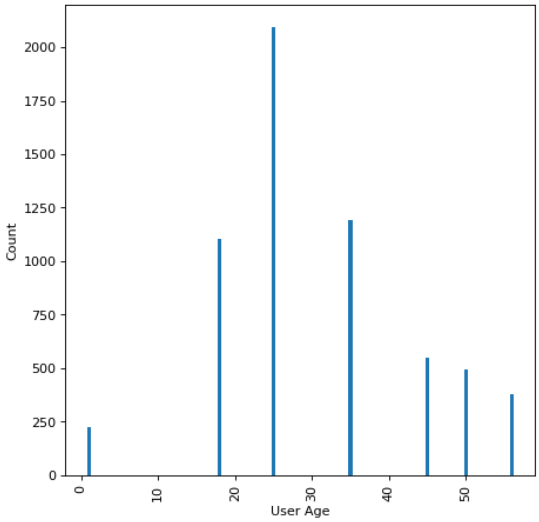
[17] C.-N. Ziegler, "Book-crossing," Oct 2013. [Online]. Available: https://grouplens.org/datasets/book-crossing/

[18] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," *CoRR*, vol. abs/1605.09477, 2016. [Online]. Available: http://arxiv.org/abs/1605.09477

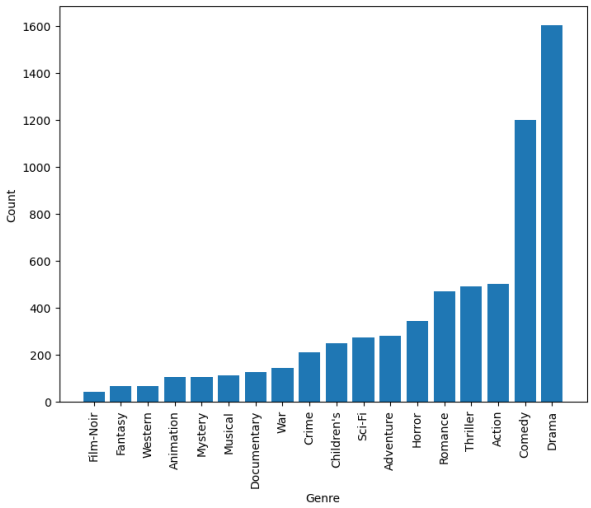
[19] S. Rendle, L. Zhang, and Y. Koren, "On the difficulty of evaluating baselines: A study on recommender systems," *CoRR*, vol. abs/1905.01395, 2019. [Online]. Available: http://arxiv.org/abs/1905.01395

A. Appendix

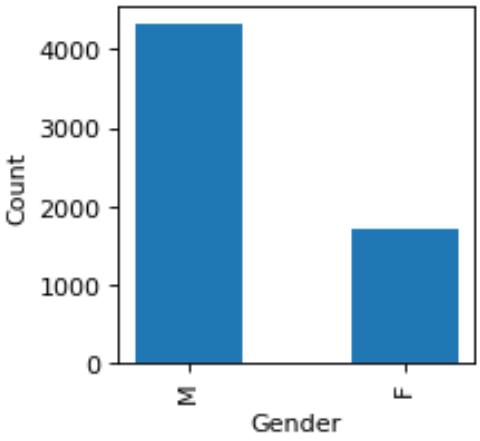
A.1. User Age Distribution



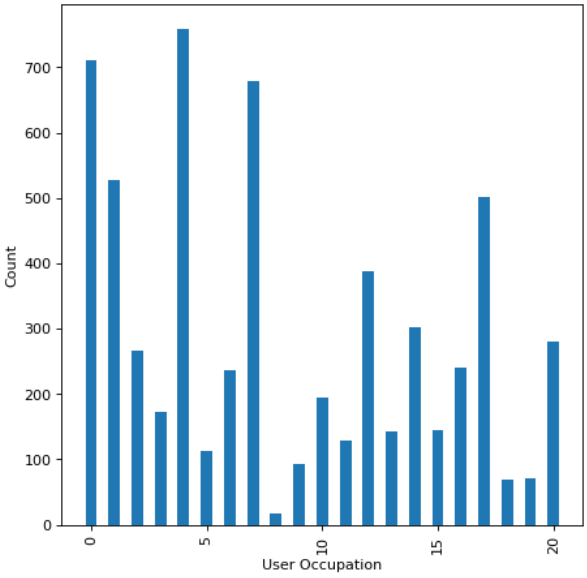
A.2. Genres of Movie



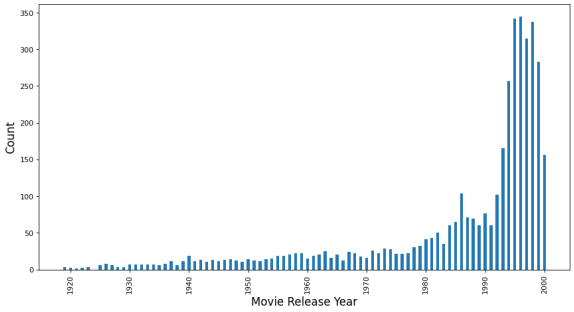
A.3. Gender Distribution



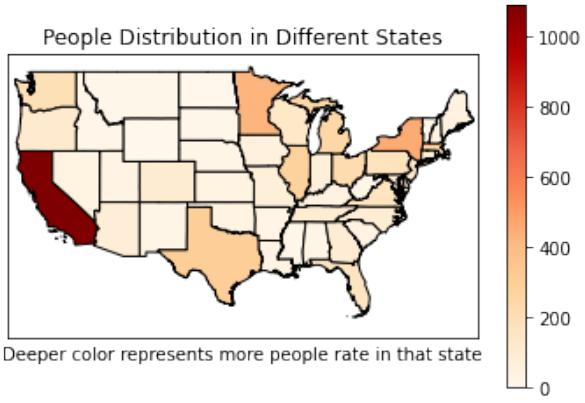
A.4. Occupation Distribution



A.5. Release Year Distribution



A.6. Pariticipants States Distribution



A.7. Rating Counts States Distribution