Amazon Recommender System

Aishwarya Harpale
Rutgers University
ach149@scarletmail.rutgers.edu

Vaishnavi Manthena
Rutgers University
vm504@scarletmail.rutgers.edu

ABSTRACT

The goal of this project is to build a recommendation system for the Digital Music domain. The key component of this project is to predict the rating a user might give to an item in the Digital Music domain based on a given amazon dataset of known user to item interactions. Making smart and personalized predictions is consequential for accurate recommendations and user-friendliness of an e-commerce website [1]. We use several algorithms to form prediction models. We evaluate each prediction model and corresponding recommendations to understand which algorithms are best and appropriate to perform recommendations.

KEYWORDS

Massive Data Mining, K Nearest Neighbors (KNN), Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), Collaborative Filtering (CF)

1 INTRODUCTION

Recommender systems are important for an e-commerce website to increase their revenue by recommending users products that are best suited to their needs. A recommendation can be said to be good when the user decides to interact with the recommended product or ends up purchasing it [8]. In this project, we have used the Amazon Reviews dataset to train and build recommendation systems using multiple algorithms. Initially, we explored the data and performed visualizations to understand the user and product patterns. Based on these patterns, we processed the data such that it would best reflect in the models to perform predictions. Then, we experimented with Model-based algorithms such as KNN and SVD and Memory-based algorithms such as User-User and Item-Item based algorithms. The performance of these algorithms was analysed on a held-out test set and they were compared so that the best performing model could be used.

2 DATASET DESCRIPTION

We used the Digital Music subset of the Amazon Review Dataset for our project from the link: http://deepyeti.ucsd.edu/jianmo/amazon/categoryFilesSmall/Digital_Music_5.json.gz.

It consists of 169,781 records and 12 columns. For preprocessing the following steps were performed:

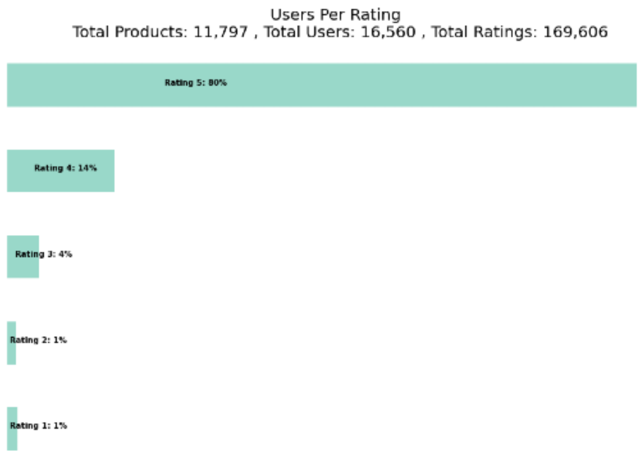
1) Removing columns that had > 80% empty values.
2) Removing records that had empty values.
3) Renaming columns to better represent the data.
The final shape of data was - 169,606 records with 9 columns.

3 EXPLORATORY DATA ANALYSIS

We performed various analysis on the dataset and visualized our understanding in the following figures.

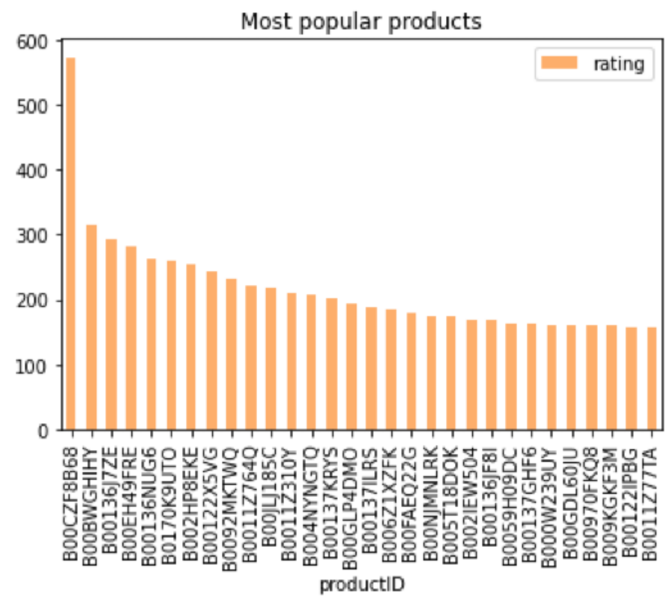
We collected the number of times each particular rating was given. Users gave a rating to the product that ranged between 1 to 5. From the dataset, we tried to find out the number of users that belonged to each of these classes. Along with this, the percentage of users out of the total users that belong to each of these classes was calculated. The fact that no zero rating was given caused us to consider the models involving making predictions from utility matrices (rows: users, columns: items) because now we don't have the disadvantage of not being able to distinguish between no rating and 0 rating.

Figure 1: Users per rating class with percentage of users per class



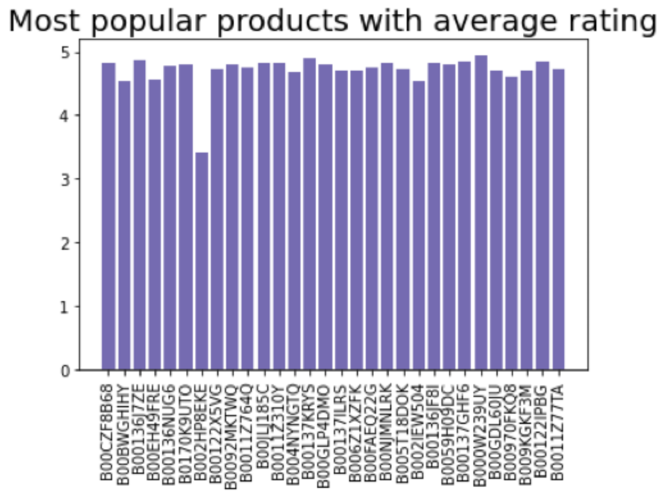
Next, we found the most popular products in the dataset. A product was called popular if it had a high number of ratings. So we plotted the most popular products with the number of ratings that each product received.

Figure 2: Most Popular Products with Ratings for each Product



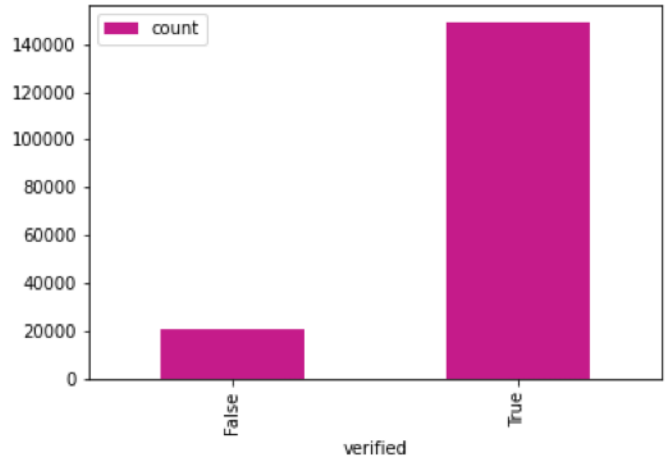
The most popular products may not necessarily have a high rating. For this, we analysed the average rating that each popular product had. This is represented in the following plot from which we see that in general the popular products do tend to have high ratings.

Figure 3: Most Popular Products with Average Rating per Product



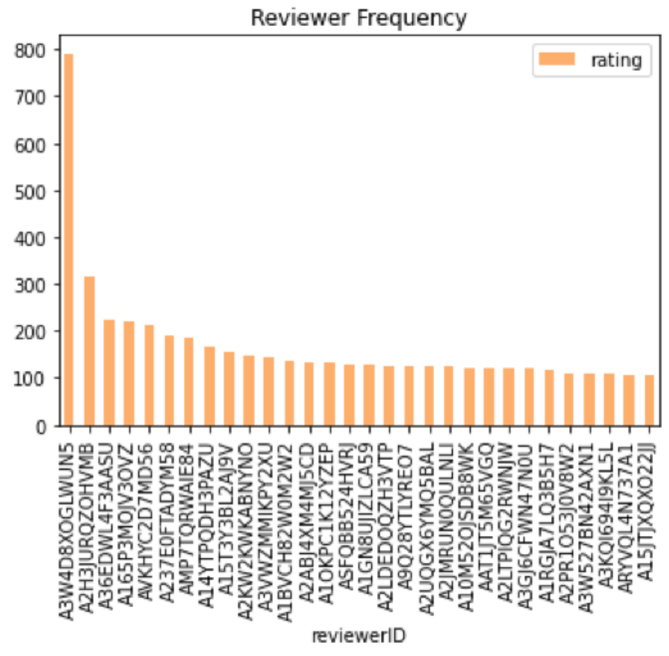
The users that submitted reviews were not necessarily verified. The number of verified verses non-verified users were analysed in the following plot.

Figure 4: Number of Verified and Non-verified Users



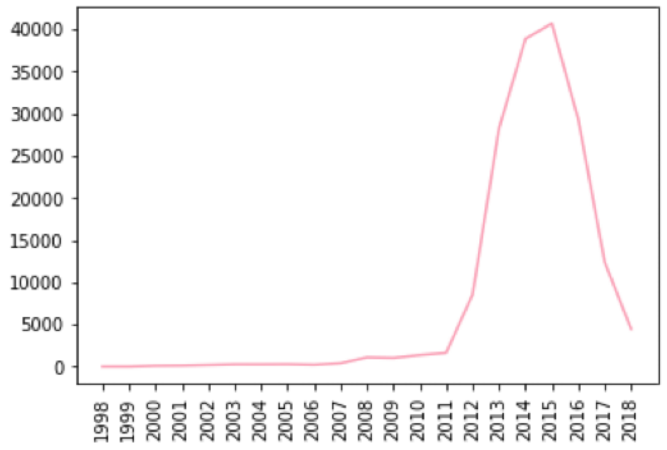
Similar to the most popular products, we tried to analyse the most frequent users. Users are said to be highly frequent if they submitted a large number of reviews.

Figure 5: Most Frequent Reviewers with their Number of Reviews



Finally, the ratings that were submitted, were reported over a large time frame. So we visualized the distribution of ratings over each year of the time frame. Following is the result.

Figure 6: Number of Reviews Submitted per Year



4 PREPROCESSING

We split the given dataset into 2 parts: 80% training data and 20% testing data. Using the information in the training data create a model to predict user to item ratings. Based on the predictions from this model construct a list of recommended items for each user.

The goal and evaluation criteria for the predictions is to minimize the mean absolute error (MAE) and the root mean square error (RMSE). Here are there formulas:

$$\text{MAE} = \frac{1}{n} * \sum_{i=1}^{n} |p_i - r_i|$$

$$\text{RMSE} = = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (p_i - r_i)^2}$$

Notation Used:
n: the number of ratings in test data
p_i: the predicted value of rating according to our model.
r_i: the real value of the rating according to test data

The goal and evaluation criteria for the recommendation lists is to have high precision, recall, F_1 measure, and Normalized Discounted Cumulative Gain (ndcg) values. Here is a brief intuition about these values:

Precision: percentage of testing items in the recommended items.

Recall: percentage of recommended items in all the testing items.

F_1 Measure: $\frac{2*precision*recall}{precision+recall}$

NDCG Measure: Takes into account the location of rated test data items in the recommendation lists.

5 THE PROPOSED MODELS

Description. There are two main algorithm categories we have tried for the making smart predictions from data. These are model-based methods and memory based ones.

5.1 Memory-Based Methods

For the memory based models we basically used the approach of collaborative filtering [6].

5.1.1 User-User Model. To predict the rating by user 'x' on item 'i' consider the similarity of other users to user 'x' using the cosine similarity. Now take the weighted average (based on user-user similarity) of the ratings given by other users to item 'i,' and use this as the prediction.

5.1.2 Item-Item Model. To predict the rating by user 'x' on item 'i' consider the similarity of other items to item 'i' using the cosine similarity. Now take the weighted average (based on item-item similarity) of the ratings given by user 'x' to other items and use this as the prediction [5] [4].

As can be seen in Figures 8, 9, and 10, the memory-based approach gives high MAE and RMSE values and low precision, recall, F_1 score, and ndcg values. Hence, using collaborative filtering methods as a baseline, we turn to the below learning/model based algorithms [2].

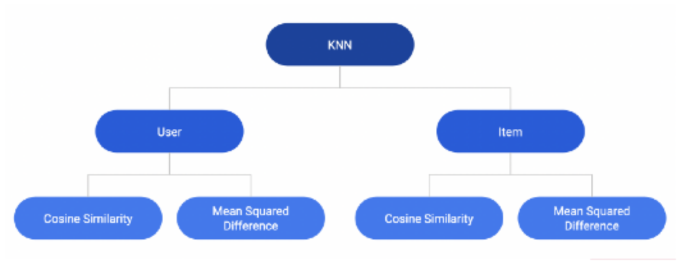
5.2 Model-Based Methods

We use cross-validation to get final predictions using these algorithms. The way this works is that the training data is split into 'k' parts (in our case 3 parts). Then there are 'k' iterations in the learning process. In each iteration, a part of the data is chosen as the testing set and the other $k-1$ parts are used as training data. Cross-validation is used to fine tune the hyperparameters and stabilize the performance metrics. This helps in reducing overfitting as the algorithm gets tested on every item at least once.

Looking at the EDA results we felt that there was scope for splitting the data into training and test sets smartly for high quality results. For example, from figure 1 we see that it may be useful to split the data in a way that the train and test sets have equal proportion of different ratings. However, we started to code by doing a simple randomized split, which ended up giving good results.

5.2.1 K Nearest Neighbors (KNN). K-Nearest Neighbors is a supervised learning method to find similarity between users or items. It helps us to capture the underlying similarity structure in the data. The similarity can be calculated either between items or users and based on the similarity between them, we can perform recommendations for a given user. The hyperparameter K is used to estimate the neighborhood to look at when finding the similarity of an item. The similarity of an item is calculated wither using Cosine Similarity or the Mean Square Difference. The specific forms of the KNN algorithm implemented in this project are depicted in Figure 7.

Figure 7: KNN and its Types



5.2.2 Singular Value Decomposition (SVD). The utility matrix (rows: users, columns: items, entries: ratings) is predicted to be the matrix product PQ^T. The entries of the P and Q matrices are learned by trying to minimize the error in the training data. This is the clear minimization problem:

$argmin_{PQ} \sum_{xi \in R} (r_{xi} - p_x q_i)^2$

R: all the ratings in training data
r_{xi}: actual rating on item 'i' by user 'x' in the training data.
$p_x q_i$: predicted rating on item 'i' by user 'x'

5.2.3 Non-Negative Matrix Factorization (NMF). NMF is a learning algorithm to extract hidden semantics from data [7]. We learn the following approximate factorization: V = W*H, with the goal that W*H forms a good estimate of V (the true ratings).
V: (user * item)
W: (user * topic)
H: (topic * item)

So (user * item) = (user * topic) * (topic * item). The "topic" helps us to understand the latent semantics of the data.

The W, H matrices are initially guesses. They are iteratively updated to minimize a loss function. The process is continued ensuring that the entries of W, H matrices are non-negative. Iteration stops when the W, H matrices are stable enough [3]. The major difference between NMF and SVD is that the matrices in the factorization are ensured to be non-negative. This makes NMF more interpretable.

6 EXPERIMENTS AND RESULTS

Based on the experiments that we performed, we tabulated our results in the following tables. The KNN variations are represented in Table 1 and the rest of the models have been tabulated in Table 2.

Table 1: KNN Results

Algorithm	MAE	RMSE	Precision	Recall	F1-score	NDCG
User(MSD)	0.4310	0.7353	0.9386	0.9494	0.9440	0.8385
User(Cosine)	0.4755	0.7799	0.9356	0.9457	0.9406	0.8386
Item(MSD)	0.3397	0.6215	0.9400	0.9522	0.9461	0.8451
Item(Cosine)	0.3561	0.6419	0.9366	0.9494	0.9430	0.8445

Table 2: Other Results

Algorithm	MAE	RMSE	Precision	Recall	F1-score	NDCG
SVD	0.3577	0.5847	0.9420	0.9620	0.9519	0.8271
NMF	0.5877	0.7798	0.9023	0.8876	0.8949	0.8242
Item-Item CF	4.673	4.728	0.0083	0.0147	0.01058	0.0358
User-User CF	4.692	4.745	0.0252	0.0440	0.0321	0.1306

The graphical results of previous tables are shown below:

Figure 8: MAE values of algorithms

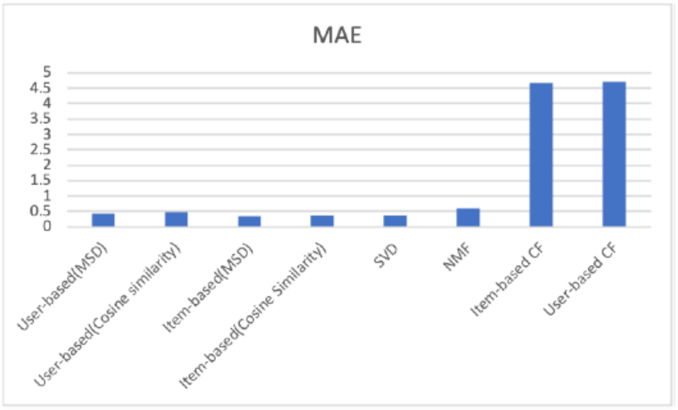


Figure 9: RMSE values of algorithms

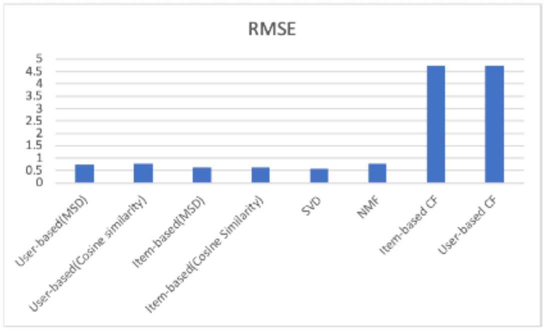
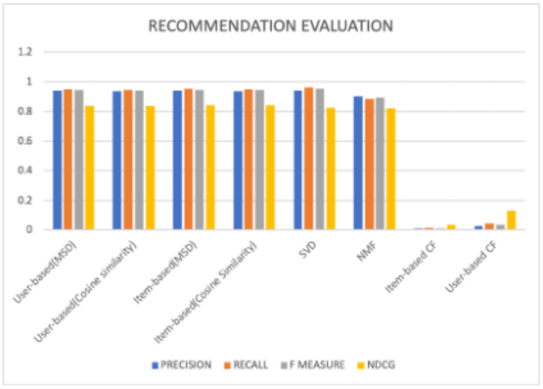


Figure 10: Evaluating recommendation lists produced



7 CONCLUSIONS AND FUTURE WORK

Based on the results, the model based algorithms using randomized splitting clearly give better quality results compared to the memory based models. Looking more closely the SVD algorithm seems to perform well overall compared to the others. It outperforms the other algorithms in terms of the RMSE, precision, recall, and F_1 measure. Therefore, we propose SVD algorithms for recommendation systems as compared to others. We believe the reason for SVD's success is that it specializes in extracting the latent factors in data which is crucial for e-commerce systems. The items (Digital Music) or users which have hidden semantics are recognized as similar and predictions/recommendations are made accordingly. Moreover, this is helpful in noise reduction. We believe it's better than NMF in this case due to higher accuracy from allowing negative matrix entries.

Our future work includes 3 main improvements:
1) Creating hybrid models by combining the ones we experimented with to try and improve results.
2) Creating embeddings for the review text and summary. Possible choices are Bag of Words, GloVE, BERT etc. Using these embeddings to try out NLP models.
3) Trying deep learning models for recommendation.

8 ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Professor Yongfeng Zhang and Zhiqing Hong for their support throughout the semester to help us understand and get familiar with important real world data mining concepts.

REFERENCES

[1] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alex Tuzhilin. Context-Aware Recommender Systems. *Recommender systems handbook*, pages 217–253, 2011.

[2] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

[3] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. *NIPS*, pages 556–562, 2001.

[4] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[6] Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 4, 2009.

[7] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk. Investigation of Various Matrix Factorization Methods for Large Recommender Systems. *Proc. ICDM*, 2008.

[8] Jian Wang and Yi Zhang. Opportunity Models for E-commerce Recommendation: Right Product, Right Time. *SIGIR*, pages 303–312, 2013.