DSA5101 Project Report

Name: Wong Zi Xin, Avellin

Student ID: A0225646B

Introduction

In this project, I plan to explore different types of recommender systems on the MovieLens100k dataset. More specifically, I will be exploring user-based collaborative filtering, item-based collaborative filtering, non-negative matrix factorization, as well as content-based filtering.

Visualisations of Dataset

MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota.

This data set consists of:

- 100,000 ratings (1-5) from 943 users on 1682 movies.

- Each user has rated at least 20 movies.

- Information about the movies (genre information, title, release date, URL)

- Simple demographic info for the users (age, gender, occupation, zip code)

The data was collected through the MovieLens web site (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up – users who had less than 20 ratings or did not have complete demographic information were removed from this data set. This makes the dataset easier to work with.

In addition to the ratings, movie information and user demographics datasets, there are also 5 disjoint dataset pairs of files (u1.base, u1.test to u5.base, u5.test) that are each 80%/20% splits of the training and test data. This will subsequently be used for 5 fold cross validation using non-negative matrix factorization.

Below are some visualisations and graphs that I performed on the dataset to gain a better understanding of the data.

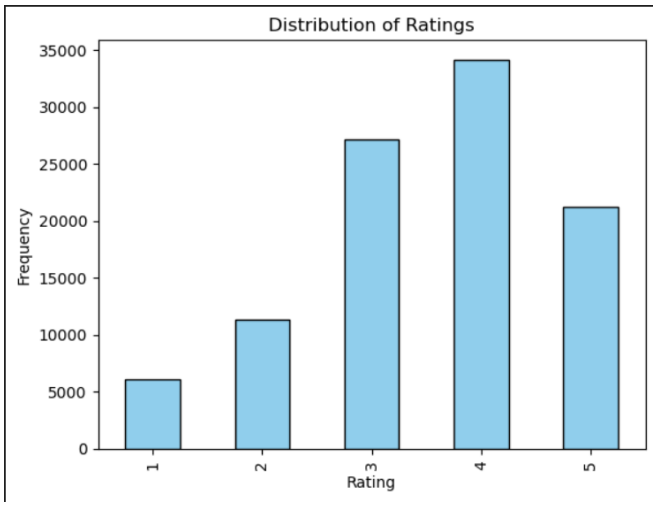

Figure 1: Distribution of ratings

We can see that most users gave ratings of 3 and 4.

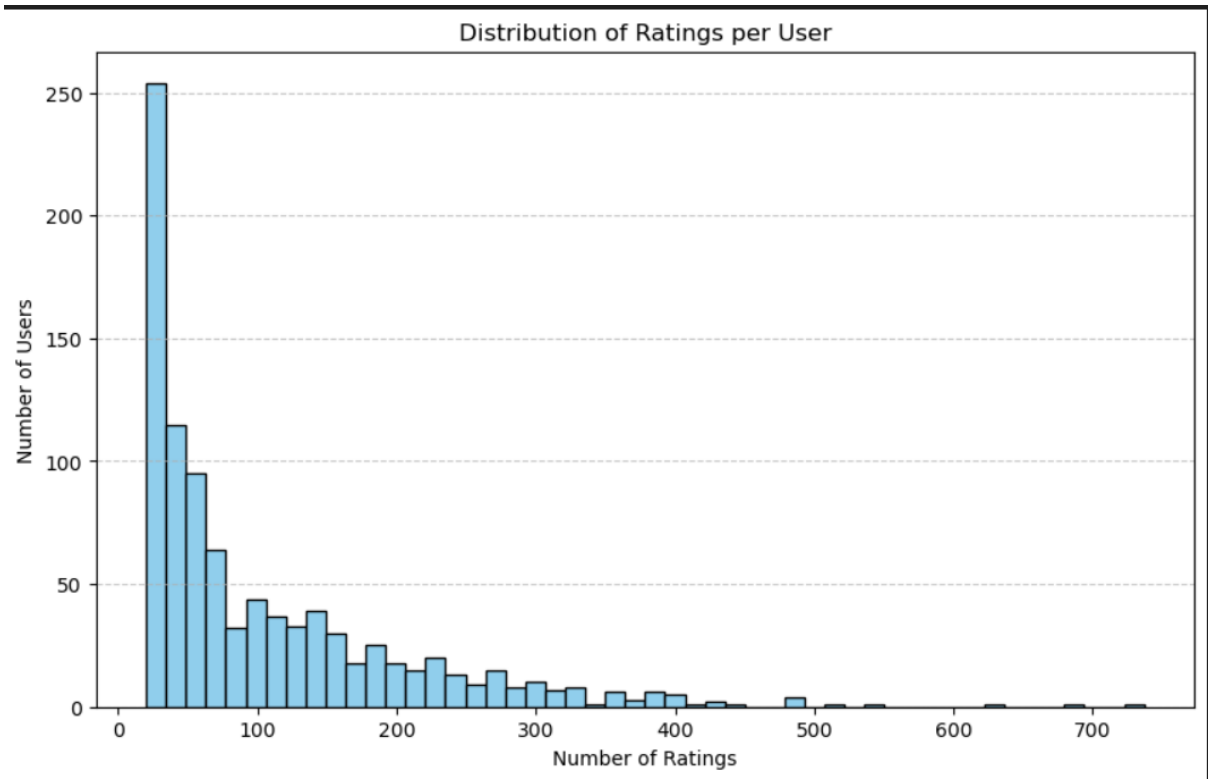

Figure 2: Distribution of ratings per user

We can see that there are a few users that rated over 400 movies. The majority of users rated a few movies (however, not less than 20, based on the dataset description).

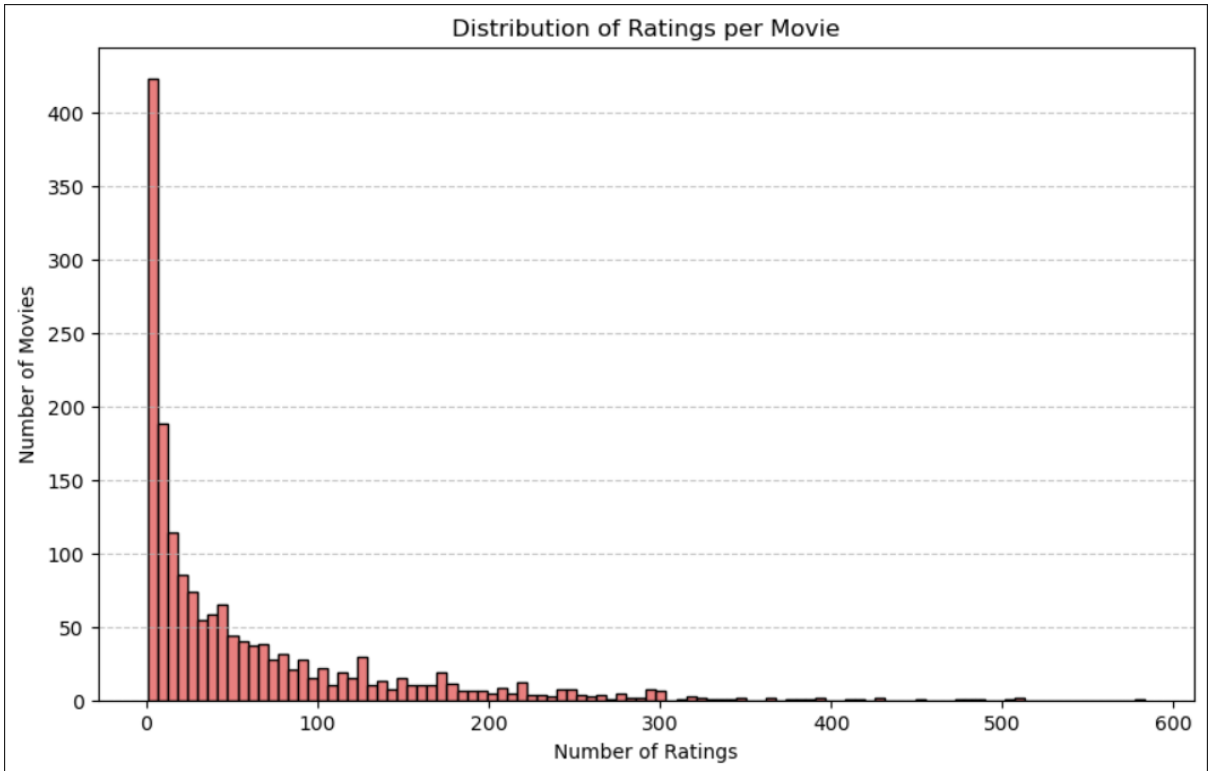

Figure 3: Distribution of ratings per movie

A similar trend can be seen for the distribution of ratings per movie. A few popular movies have over hundreds of ratings, while the majority have less.

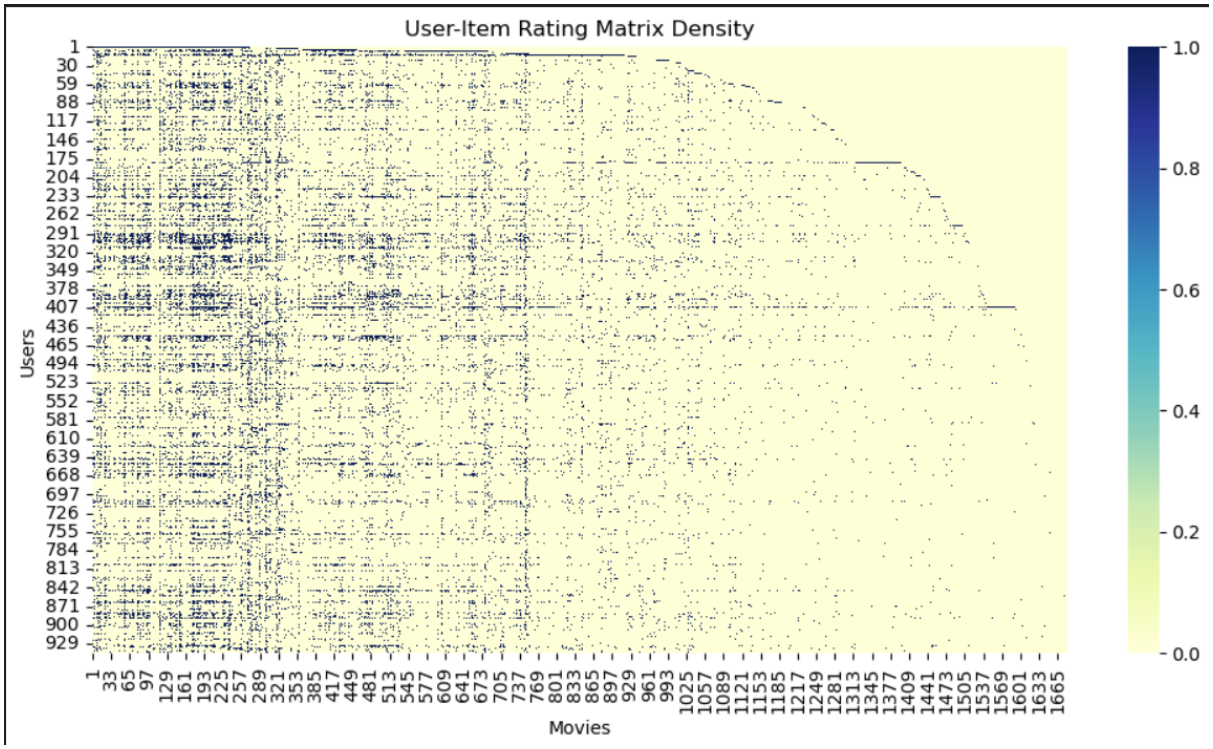

Figure 4: Heatmap showing the density of the user-item (movie) rating matrix

We can see that the matrix is quite dense especially towards the left side, with many regions of blue.

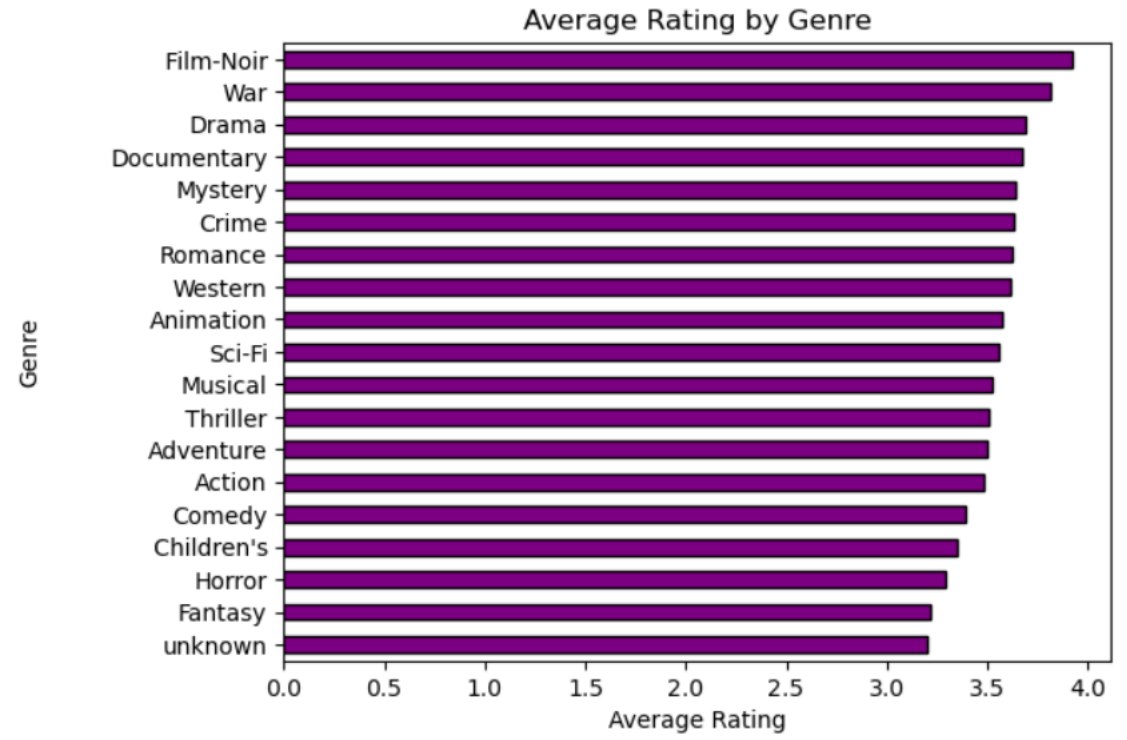

Figure 5: Average ratings per movie genre

We can see that the ratings for all genres are quite consistent and between 3-4.

Methods applied

Collaborative filtering - User-based and Item-based filtering

The goal of user-based filtering is to identify similar users and recommend a user things that similar users liked. "Similarity" is measured based on ratings, and not other features like user demographics. To do so, a user-item matrix of ratings is computed (users as rows, items as columns), and then similarity scores between each user is computed. After that, for an item i, find the top k number of similar users (k is a number to be set), and compute predictions of the rating of that item by user u using the formula:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u, v)}$$

where N is the set of k most similar users to u

Item-based filtering is similar to user-based filtering except that we instead identify similar items and estimate the rating for a particular item based on ratings for similar items. Instead of a user-item matrix, we compute an item-user matrix of ratings instead. Then, similarity scores between each item is computed. After that, for an item i, find the top k number of similar items, and compute predictions of the rating of that item by user u using the formula:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{sim}(i, j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} \text{sim}(i, j)}$$

where N is the set of k most similar items to item i, rated by user u

In both cases, the rating predictions on the test dataset are then compared to the actual ratings, and metrics such as root mean squared error (RMSE) and mean absolute error (MAE) are employed to measure the performance.

In my implementation, the Python surprise library is used, as it is a scikit for building and analysing recommender systems that contains functions to make it easy to implement the algorithms. I chose KNNBasic as the algorithm to use. For the parameters k (maximum number of neighbours) and min_k (minimum number of neighbours for aggregation), I used GridSearchCV to determine the optimal set of parameters for RMSE and MAE over the ranges of k = [10, 20, 30, 40, 50, 60] and min_k = [1, 3, 5, 7, 10]. The default parameters are k = 40 and min_k = 1. If the optimal parameters differ for both RMSE and MAE, the one for RMSE is used. The similarity measure configuration (name of similarity measure, user or item-based collaborative filtering) can be set by the sim_options parameter of the KNNBasic function. I first split the dataset into 80% training, 20% test, before utilising both cosine similarity and pearson-correlation coefficient for both user and item-based collaborative filtering on the training set, and then evaluating the predictions on the test dataset using RMSE and MAE.

Collaborative filtering – Non-negative matrix factorization (with and without cross validation)

Non-negative matrix factorization (NMF) is a technique used to predict missing entries since not every user is going to rate every item, causing the user-item matrix to be sparse. It works by decomposing the main matrix into the product of 2 lower dimensional matrices (with non-negative

entries). The first matrix has the same number of rows as the main matrix and each row belongs to each person, while the second matrix has a column for each item. Each of the two matrices expresses the power of association between a user or an item with latent factors or hidden features. The product of the 2 matrices should approximate the original matrix with all entries filled. Gradient descent is typically used for the optimization to obtain the 2 lower dimensional matrices. To predict the rating of an item by a user, take the dot product of row corresponding to that user in the first matrix and the column corresponding to that item in the second matrix.

I first implemented NMF without cross validation by utilising the 80%/20% split of the dataset. Then, I implemented NMF on a 5 fold cross validation of the dataset, then computed the average RMSE and MAE across the different folds. I first did this manually on the 5 sets of disjoint data already provided (u1.base to u5.test), before using the Surprise's library cross_validate function to verify the results.

Content-based filtering

Content-based filtering takes into account features of the users and items in making recommendations.

Similar to collaborative filtering, I split the ratings dataset 80%/20% into a train and test set.

For the movie (item) profile, the features chosen are average rating of the movie, genre (one-hot encoding, already provided in the dataset), and the average user demographics that rated the movie at least 3 and above (from the training set). This is done by one-hot encoding of the user demographics (gender and occupation). Zip code was not considered. Eventually, we obtain a movie profile matrix.

The user profile for each user was then obtained using a weighted average of rated movie profiles (using the ratings from the training set). The formula is shown below:

$$\text{User Profile} = \frac{\sum(\text{Feature Vector} \times \text{Rating})}{\text{Sum of Ratings}}$$

Combining all individual user profiles into a matrix, we obtain a user profile matrix.

Next, compute the cosine similarity between the user profiles matrix and the movie profiles matrix. The cosine similarities are then used to recommend the top 20 items to each user. Metrics such as precision, recall and f1 score are used to evaluate the performance of the model on the test set.

Analysis of Results

Collaborative filtering - User-based and Item-based filtering

The GridSearchCV results are tabulated below:

	User-based	**Item-based**
Cosine	k=60 (RMSE)/k=40 (MAE), min_k = 3	k=60, min_k=1
Pearson	k=60, min_k=3 (RMSE)/min_k = 1 (MAE)	k=60, min_k=1

Using the optimal values of k and min_k for RMSE found through grid search, the collaborative filtering results are tabulated below:

RMSE

	User-based	**Item-based**

Cosine	1.0191	1.0246
Pearson	1.0131	1.0337

MAE

	User-based	**Item-based**
Cosine	0.8047	0.8092
Pearson	0.8031	0.8261

First, we notice that user-based filtering performed slightly better than item-based filtering. This suggests that on this dataset, predicting a user's rating based on other similar users aligns more closely with actual ratings compared to predicting based on similar items. This could be because users tend to have more consistent preferences compared to the similarity patterns among items. As we can see from Figure 5, the average rating of the movies by genre is quite similar across genres (between 3 to 4), showing that users generally have consistent taste among the movie items.

Pearson similarity tends to account for correlations by centering on the mean rating, which may capture preferences more accurately than cosine. However, from our results, pearson similarity did not necessarily perform better than cosine similarity.

We also notice that the differences are quite small. This could be because the MovieLens 100k dataset is relatively dense, with each user rating about 20 movies on average, so there is enough overlap among users and items. This means that both user-based and item-based methods have enough data to find reliable neighbours, leading to similar predictions. In addition, the dataset is designed with well-distributed ratings across users and movies. Users also follow similar rating scales, with most users rating movies between 3-5 (Figure 1). Hence, with dense overlap in ratings (as seen from the Figure 4 heatmap) and relatively homogenous user behaviour, the advantages of Pearson's mean centering become less pronounced.

For small datasets, very sparse user-item matrices, or highly skewed rating scales (one user rates everything low, another user rates everything high), the differences could be way more obvious.

Collaborative filtering – Non-negative matrix factorization (with and without cross validation)

Without cross validation, I obtained an RMSE of 0.9699 and MAE of 0.7632.

Manual iteration across the 5 folds of data already provided resulted in an average RMSE of 0.9664 and an average MAE of 0.7590. Using the cross_validate function provided by Surprise led to comparable results (average RMSE: 0.9630, average MAE: 0.7568). This is slightly lower than the results obtained for user-based and item-based collaborative filtering.

The performance of NMF is slightly better than user and item-based collaborative filtering as the decomposition intro matrices captures latent factors that represent hidden patterns in the data, rather than relying solely on explicit user-user or item-item similarities. By capturing these underlying structures, NMF can make more accurate predictions for users with limited interaction data or items with fewer ratings, leading to lower error rates.

However, since the dataset is already relatively dense, the dimensionality reduction by NMF to mitigate issues of sparsity likely did not have a significant improvement.

Content-based filtering

The metrics I used to evaluate the model are precision, recall and f1 score. Precision measures the proportion of recommended items that are relevant to the user, and can be calculated using the formula below for a top-N recommendation:

$$\text{Precision} = \frac{\text{Number of Relevant Items in Top-N}}{N}$$

Recall measures the proportion of relevant items that were recommended, and can be calculated using the formula below for a top-N recommendation:

$$\text{Recall} = \frac{\text{Number of Relevant Items in Top-N}}{\text{Total Relevant Items for User}}$$

F1 score is defined as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

I defined "relevant" items for each user as movies that were rated at least 3 by each user in the test set. If a particular user does not exist in the test set (hence total relevant items = 0), I skipped over that user. After calculating the precision, recall and f1 scores of each user, I averaged them out to obtain the final result.

Precision	0.0311
Recall	0.0437
F1 score	0.0304

The model does not have a good performance. This could be due to insufficient features (genre + average user demographics contains only gender and occupation), or that the features do not have much impact or correlation with how the user rates different genres of movies. Other recommendation algorithms may be more suitable (like collaborative filtering).

Limitations and Future Work

There are many more KNN-inspired algorithms in the surprise library apart from KNNBasic, such as KNNWithMeans, KNNWithZScore, KNNBaseline etc. More exploration could be done with these other algorithms for collaborative filtering and analyse their impact on the performance.

For collaborative filtering, the differences in results between different algorithms and similarity measures are also quite small. More work can be done to determine if they are statistically significant, such as hypothesis testing.

For content based filtering, other kinds of similarity measures, or changing the top N number of recommendations to provide can be experimented with to improve the model performance.

Conclusion

This project aims to explore different types of recommender systems on the MovieLens100k dataset. We discovered that user-based collaborative filtering performed slightly better than item-based collaborative filtering, and pearson similarity did not necessarily perform better than cosine similarity.

NMF yielded slightly better results than both user and item-based collaborative filtering. However, the differences are small likely due to the distribution of the dataset.

We also attempted to build a content-based filtering system based on features such as movie genre, average demographics (gender, occupation) of users that rated the movie >=3. However, the performance of the model is poor and more investigation is needed for better analysis.