CS 550 Final Project Report

Movie Recommendation System

Adithya Shrivastava as3652

Jong Seong Shim jss412

Sanchit Thakur st976

Abstract

Recommender systems are algorithms aimed at suggesting relevant items to users, for e.g., movies to watch, text to read, products to buy or anything else depending on industries. Recommendation systems are highly useful as they help users discover products and services they might otherwise not have found on their own. Our aim with this project is to build a movie recommendation system using the Movielens dataset. In building a movie recommendation system, we aim to predict ratings given by a user to a movie, as well as generate a list of recommended movies to the user. Various models have been implemented to meet this aim, such as content based model, collaborative filtering model, SVD model and KNN based model. We have also implemented an enhanced hybrid model which combines the neural model with collaborative filtering. The predicted ratings and recommended movies are evaluated for their performance using various metrics, and it is found that the hybrid model has the best performance. A detailed analysis of each model is provided in the report.

Keywords

recommendation system, content-based model, collaborative filtering, SVD, KNN, hybrid model, neural collaborative filtering

1 Introduction

A recommender system, or a recommendation system is a subclass of information filtering system that provide suggestions for items that are most pertinent to a particular user [1]. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what online news to read. Recommender systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer. Recommender systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions.

Recommender systems are used in a variety of areas, with commonly recognised examples taking the form of playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services.

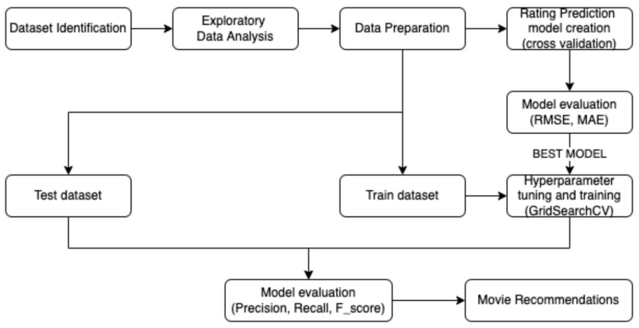

Dataset Identification
Exploratory Data Analysis
Data Preparation
Rating Prediction model creation (cross validation)
Model evaluation (RMSE, MAE)
BEST MODEL
Hyperparameter tuning and training (GridSearchCV)
Test dataset
Train dataset
Model evaluation (Precision, Recall, F_score)
Movie Recommendations

Figure 1: Main workflow of movie recommendation system

Specifically, a movie recommendation system is a type of software or application that provides users with personalized movie suggestions based on their viewing history, preferences, and behavior. The primary goal of a movie recommendation system is to provide a better viewing experience for users by suggesting movies that are relevant and appealing to them. This project aims to develop a movie recommendation system that can analyze user data and provide recommendations that suit their tastes. Different models are implemented to fulfill the same purpose of predicting movie ratings and generating movie recommendations. The various models implemented are: content-based filtering, collaborative filtering, SVD based, KNN based and a hybrid model combining a neural network with collaborative filtering. The rest of the paper contains the following sections: Section 2 gives a brief description of the related work and literature survey, Section 3 defines the main problem statement of this project, Section 4 describes the dataset being used, Section 5 describes the process followed for the preprocessing of the data, Sec 6 describes in detail the models mentioned above, Section 7 discusses the results obtained and Section 9 discusses the main takeaways and future plans.

2 Related Work

There exist various types of recommendation systems that use different techniques to determine the most appropriate items to recommend to a particular user. The two major types of recommendation systems used are content-based and collaborative filtering-based recommendation systems. Content-based recommendation systems make recommendations by finding items similar to those already liked or purchased by a user, based on the inherent properties of the item itself. For example, in our problem of interest (movie recommendation), similar movies to those liked by a user can be found by comparing properties of the movies such as genre, actors, directors, tags, and so on. The other major type of recommendation system, collaborative filtering-based recommendation systems make use of user-item interaction data to make recommendations. They are of two types: user-user and item-item collaborative filtering. To evaluate the performance of these recommendation systems, the most popular metrics used are the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), while popular metrics used for evaluating the effectiveness of recommendations are precision, recall, F-measure, NDCG, and Mean Average Precision (MAP).

The dataset used in our project is the Movielens dataset. This dataset has two versions, the Movielens full dataset and the Movielens small dataset. The Movielens dataset has frequently been used in prior work on movie recommendation systems. For example, Chawla et al. [4] have used the Movielens dataset to build a content-based model, a collaborative filtering-based model, a latent factor model that uses singular value decomposition (SVD), a combined model which is a linear combination of the collaborative filtering-based model and the latent factor model, a popularity model, and a hybrid model which combines all of these models. The performance of each of these models was analyzed individually. Similarly, Mu and Wu [5] have implemented a multimodal movie recommendation system that uses the hidden features of users as well as that of the movies to predict movie ratings by using a deep learning network algorithm model. They found the performance of this method to be better than the vanilla content-based, collaborative filtering-based, and SVD approaches. In this project, we aim to analyze the performance of the various recommendation system approaches, which include the basic content-based, user-user collaborative filtering, and item-item collaborative filtering recommendation approaches, a KNN-based recommendation approach, an SVD-based collaborative filtering approach, as well as a hybrid neural collaborative filtering-based model, and evaluate the performance of each of these approaches using the appropriate evaluation metrics.

3 Problem Formalization

The main problem statement and tasks to be performed for this project are stated below:

- Data preprocessing: creating a training dataset and a testing dataset for experiment. For each user, randomly select 80% of his/her ratings as the training ratings, and use the remaining 20% ratings as testing ratings. At last, the training ratings from all users consist the final training dataset, and the testing ratings from all users consist the final testing dataset

- Rating prediction: developing an algorithm to predict the ratings in the testing set based on the information in the training dataset, and evaluating the predictions based on MAE (Mean Absolute Error) and RMSE (Root Mean Square Error)

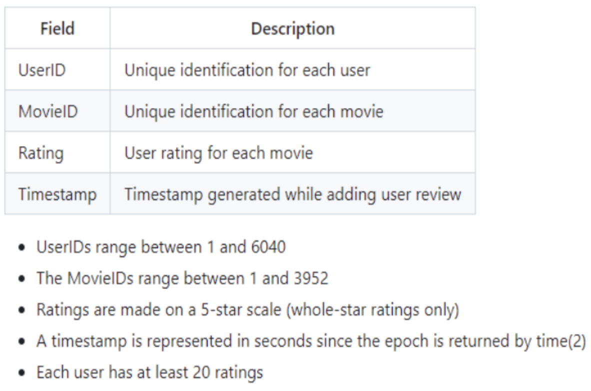
- Item Recommendation: constructing a recommendation list for each user, and then evaluating the recommendation quality based on precision, recall, F-measure and NDCG (Normalized Discounted Cumulative Gain)

4 Dataset

The dataset being used for this project is the Movielens small Dataset (ml-latest-small). The dataset consists of:

- Number of Ratings: 100k
- Number of Users: 60
- Number of Movies: 9k

This dataset (ml-latest-small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. Each rating has a value between 0-5. It contains 100836 ratings and 3683 tag applications across 9742 movies. This data was created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018. The data are contained in the files links.csv, movies.csv, ratings.csv and tags.csv.



Field	Description
UserID	Unique identification for each user
MovieID	Unique identification for each movie
Rating	User rating for each movie
Timestamp	Timestamp generated while adding user review

- UserIDs range between 1 and 6040
- The MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- A timestamp is represented in seconds since the epoch is returned by time(2)
- Each user has at least 20 ratings

Figure 2: The different fields in the dataset

5 Preprocessing Data

In the preprocessing of the data, we split the dataset into test and train on the basis of UserID. This means that for each user, randomly select 80% of his/her ratings as the training ratings, and use the remaining 20% ratings as testing ratings. At last, the training ratings from all users consist the final training dataset, and the testing ratings from all users consist the final testing dataset. Because each rating is accompanied with a single piece of textual review, so the reviews are also automatically split into training and testing sets.
Cleaning of data: The data was also cleaned, which included removal of redundant and non-contributing features from the dataset. Since we need at least 1 data point for testing, we also proceeded to exclude any user having less than 5 data points. In addition, genres and other categorical features were converted to one-hot vector.

6 Exploratory Data Analysis

To gain a better understanding of the Movielens dataset, we performed an exploratory data analysis. Through this process, we made several notable observations about the data:

1. **Distribution of Ratings** (Figure 12):
   The most common rating given by users was 4, followed by 3 and 5. This suggests that users tend to rate movies positively.

2. **Number of Movies by Genre** (Figure 13):
   The most common genre is Drama, followed by Comedy, and Action. The least common genre is Film-Noir

3. **Genres with Highest Average Rating** (Figure 14)
   Film-Noir, Documentary, and War movies had the highest average ratings. In contrast, Horror movies had the lowest average rating.

4. **Top Ten Movies with Highest Average Ratings** (Table 1):

Movie Names	Rating
Sanjuro (1962)	4.608696
Seven Samurai (The Magnificent Seven)	4.560510
Shawshank Redemption, The (1994)	4.554558
Godfather, The (1972)	4.524966
Close Shave, A (1995)	4.520548
Usual Suspects, The (1995)	4.517106
Schindler's List (1993)	4.510417
Wrong Trousers, The (1993)	4.50793
Sunset Boulevard (1950)	4.491489
Raiders of the Lost Ark (1981)	4.477725

Table 1: Top 10 Movies with Highest Ratings

7 Model Description

7.1 Content-Based Filtering

The content based recommendation system is a type of recommendation system that uses the attributes of items or products to recommend similar items to users. In a content-based recommendation system, the system first analyzes the attributes or features of an item, such as text descriptions, tags, or metadata and creates a profile for that item. This profile is then used to compare user's past preferences and recommend other items that share the same characteristics. For example, suppose a user has previously watched and highly rated Toy Story 1995, which falls under the Animation and Children's genres. In such a scenario, our system would suggest other titles belonging to the Animation and Children's genres (Aladdin and the King of Thieves and Bug's Life, A) to this user (See Figure 3).

In our project, we employed content-based filtering with genre serving as the primary feature to provide movie recommendations to users. To begin, we identified all of the key genres in our dataset. We then created a user profile for each user based on their past interactions with the system, which was used to determine the genres of movies that the user is most likely to enjoy. Next, we utilized cosine similarity (Equation 1) and TF-IDF (Term Frequency-Inverse Document Frequency) to analyze both the movie genres and ratings that the user has favored in the past (Equation 2, 3, 4). By calculating the cosine similarity between a user's past movie preferences and the genre of other movies in our recommendation system, we were able to recommend similar movies that align with the user's tastes. Additionally, our system took into account the user's ratings to recommend movies that not only matched their preferred genres but also their preferred quality of content. Through this content-based approach, our system was able to recommend movies with similar genres and ratings to users, based on their past interactions with the system.

The cosine similarity between item i and item j is as follows:

$$\text{similarity}(i,j) = \frac{\sum\limits_{u\in U_{i,j}} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum\limits_{u\in U_i} r_{u,i}^2} \cdot \sqrt{\sum\limits_{u\in U_j} r_{u,j}^2}} \tag{1}$$

where i and j are the two items being compared, $U_{i,j}$ is the set of users who have rated both items i and j, $r_{u,i}$ is the rating user u gave to item i, and $r_{u,j}$ is the rating user u gave to item j. $\sum\limits_{u\in U_i} r_{u,i}^2$ and $\sum\limits_{u\in U_j} r_{u,j}^2$ are the sums of the squared ratings for items i and j, respectively.

The formula for calculating the TF-IDF (Term Frequency-Inverse Document Frequency) is:

$$\text{tf-idf}(t,d,D) = \text{tf}(t,d) \times \text{idf}(t,D) \tag{2}$$

where t is a term, d is a document, and D is the entire corpus.

The term frequency $\text{tf}(t,d)$ is the frequency of term t in document d, and can be calculated as:

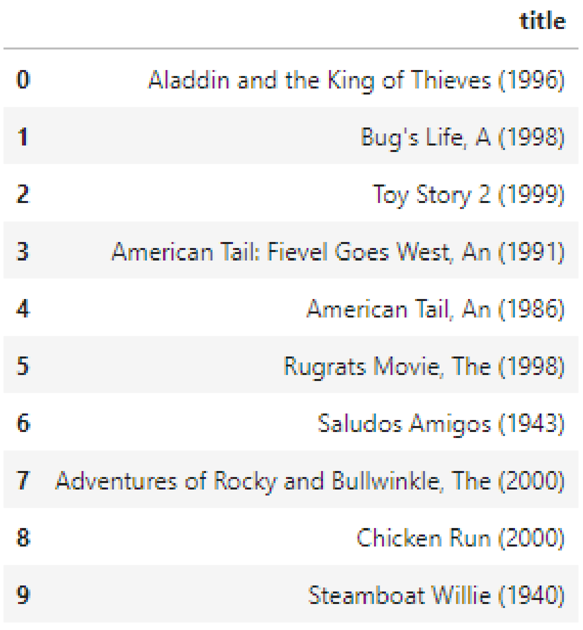
$$\text{tf}(t,d) = \frac{f_{t,d}}{\sum_{t'\in d} f_{t',d}} \tag{3}$$

where $f_{t,d}$ is the frequency of term t in document d.

The inverse document frequency $\text{idf}(t,D)$ is a measure of how important a term is in the corpus, and can be calculated as:

$$\text{idf}(t,D) = \log\frac{N}{n_t} \tag{4}$$

where N is the total number of documents in the corpus, and n_t is the number of documents in the corpus that contain the term t.



	title
0	Aladdin and the King of Thieves (1996)
1	Bug's Life, A (1998)
2	Toy Story 2 (1999)
3	American Tail: Fievel Goes West, An (1991)
4	American Tail, An (1986)
5	Rugrats Movie, The (1998)
6	Saludos Amigos (1943)
7	Adventures of Rocky and Bullwinkle, The (2000)
8	Chicken Run (2000)
9	Steamboat Willie (1940)

Figure 3: Illustrating Content-Based Filtering: Recommending Top 10 Movies Similar to Toy Story (1995)

7.2 Collaborative Filtering

Collaborative filtering is a recommendation technique that is used to predict a user's preference for items based on the preferences of other users. In general, collaborative filtering can be divided into two categories:

7.2.1 User-User

User-user collaborative filtering is a technique used in recommendation systems to provide personalized recommendations to users based on the preferences and behavior of similar users. This method relies on the idea that users with similar tastes or preferences will like similar items.

The user-user collaborative filtering approach works by identifying a set of similar users to a given user, and then recommending items that those similar users have rated highly, but that the given user has not yet interacted with. The algorithm calculates the similarity between users based on their past interactions with items in the system. It then finds the nearest neighbors of the user based on this similarity metric.

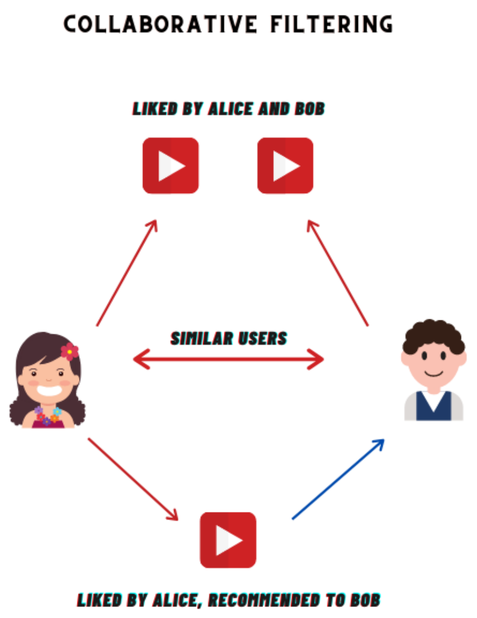


Figure 4: Working of User Based Collaborative Filtering

The system collects data on the interactions between users and items. This can include ratings, reviews, clicks, purchases, or any other form of user-item interaction data. The system then calculates the similarity between users based on their past interactions with items in the system. There are various similarity metrics that can be used, such as cosine similarity, Pearson correlation, or Jaccard similarity. For our project, we have used the Pearson correlation.

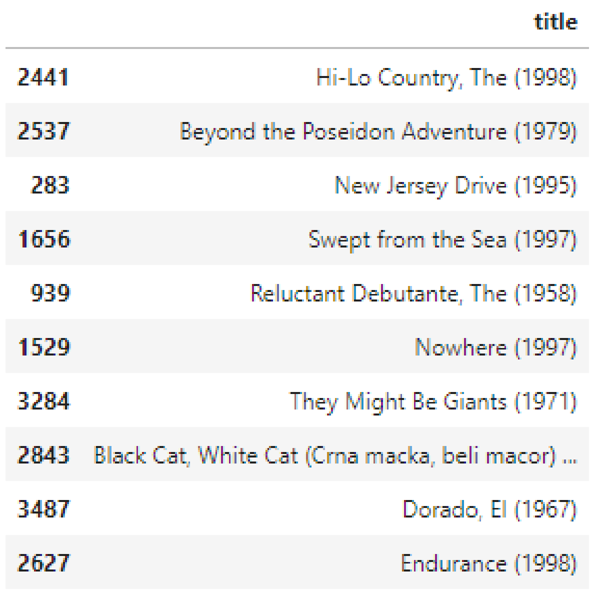
Once the nearest neighbors are identified, the algorithm recommends items that those users have rated highly, but that the target user has not yet interacted with. This approach can provide highly personalized recommendations to users, especially in cases where there are many users and items in the system.

One of the advantages of user-user collaborative filtering is that it can be applied to both explicit feedback (ratings, reviews) and implicit feedback (browsing behavior, purchase history) data. However, it can be computationally expensive to calculate the similarity between users, especially in cases where there are many users and items in the system. To overcome this issue, various optimization techniques, such as matrix factorization, can be used.

7.2.2 Item-Item

The item-item collaborative filtering recommendation system is a type of recommendation algorithm that uses similarities between items [3]. This recommendation system uses a similarity matrix that represents the similarity between different items in the system. This similarity matrix is calculated based on user ratings. Similarity between items can be calculated using various similarity measures, such as cosine similarity or Pearson correlation (Equation 5). Once the similarity matrix is computed, the algorithm finds the most similar items to the ones the user has already interacted with, based on the similarity matrix. It then recommends these similar items to the user. The item-item collaborative filtering algorithm uses a similarity matrix to identify movies similar to those previously rated highly by a user. For instance, the algorithm may recommend other movies that share similarities with a highly-rated movie. Figure 5 shows an example of the top 10 recommended movies for user 1 based on this approach.

In our project, we utilized item-item collaborative filtering to generate personalized movie recommendations for users. To accomplish this, we employed Pearson correlation to measure the similarity between different items (movies) in the dataset, and identify similar movies that a user is likely to enjoy. To prepare the dataset for analysis, we first cleaned up missing values and eliminated duplicate entries. We then transformed the data into a matrix where users were represented in rows and movies in columns. We also standardized the matrix by subtracting each movie's average rating from every entry in the corresponding row. Next, we computed the similarity matrix using the standardized matrix. Pearson correlation was used to calculate the similarity between items in the dataset. Once the similarity matrix was established, we could leverage it to find movies that were most similar to the ones a user had previously interacted with. Using the similarity matrix, we were able to predict the ratings that a user would give to movies they had not yet rated.



	title
2441	Hi-Lo Country, The (1998)
2537	Beyond the Poseidon Adventure (1979)
283	New Jersey Drive (1995)
1656	Swept from the Sea (1997)
939	Reluctant Debutante, The (1958)
1529	Nowhere (1997)
3284	They Might Be Giants (1971)
2843	Black Cat, White Cat (Crna macka, beli macor) ...
3487	Dorado, El (1967)
2627	Endurance (1998)

Figure 5: Top 10 Movie Recommendations for User 1 Using Item-to-Item collaborative Filtering.

The formula for calculating Pearson correlation between item i and item j is:

$$\text{sim}\, i, j = \frac{\sum u(r_{u,i} - \bar{r_i})(r_{u,j} - \bar{r_j})}{\sqrt{\sum_u (r_{u,i} - \bar{r_i})^2} \sqrt{\sum_u (r_{u,j} - \bar{r_j})^2}} \tag{5}$$

where $r_{u,i}$ is the rating given by user u to item i, $\bar{r_i}$ is the mean rating for item i across all users, and $\sqrt{\sum_u (r_{u,i} - \bar{r_i})^2}$ is the standard deviation of the ratings for item i across all users. Similarly, $\bar{r_j}$ and $\sqrt{\sum_u (r_{u,j} - \bar{r_j})^2}$ are the mean rating and standard deviation for item j across all users, respectively.

7.3 SVD Based Model

Singular value decomposition (SVD) is a technique used commonly in recommendation system to predict the ratings that users have not rated. The SVD approach involves decomposing the user-item rating matrix into three matrices: a user-feature matrix, a feature-feature matrix, and an item-feature matrix [6]. The user-feature matrix captures the relationships between users and latent features, while the item-feature matrix represents the relationships between items and latent features. The Σ matrix is a diagonal matrix containing the rankings of each latent feature.

To build a recommendation system using SVD, we preprocessed the dataset by cleaning up missing values and removing duplicates. Next, We used the SVD algorithm provided by the Surprise library to factorize the user-item rating matrix into three matrices - a user-factor matrix, a factor-factor matrix, and an item-factor matrix. Then, we used the GridCVSearch from sklearn library to find hyperparameters that give the lowest RMSE and MAE. For our SVD recommendation system we used the following parameters:

- Number of epochs: 20
- Number of latent factors: 100
- Learning rate: 0.005
- Regularization rate: 0.02

Once the factorization was complete, we used the resulting matrices to predict the missing ratings.

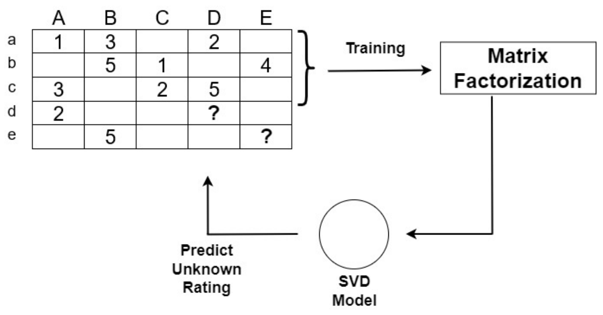


Figure 6: An example of the SVD training process

7.4 KNN Based Model

K-nearest neighbor (KNN) is a popular algorithm used in recommendation systems to provide personalized recommendations to users. KNN is a memory-based algorithm, which means that it uses the entire dataset to make recommendations, rather than building a model or a set of rules. Here's how a KNN-based recommendation system works:

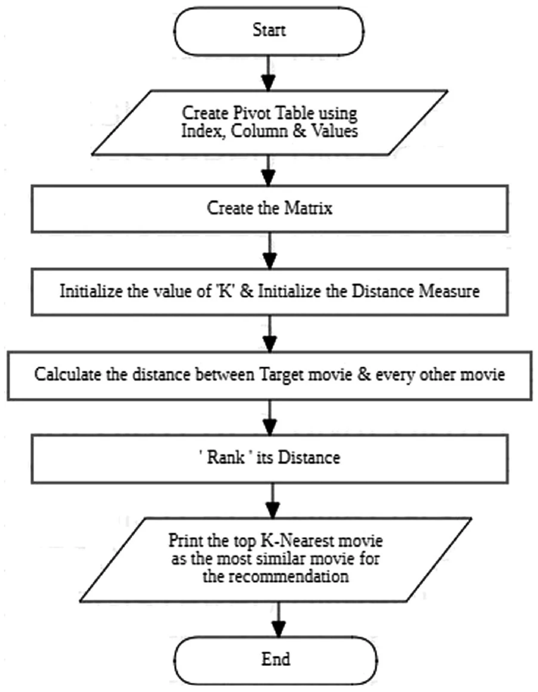

Start
Create Pivot Table using Index, Column & Values
Create the Matrix
Initialize the value of 'K' & Initialize the Distance Measure
Calculate the distance between Target movie & every other movie
' Rank ' its Distance
Print the top K-Nearest movie as the most similar movie for the recommendation
End

Figure 7: Working of KNN based model

- User-item interaction data: The system collects data on the interactions between users and items. This can include ratings, reviews, clicks, purchases, or any other form of user-item interaction data.

- User-item matrix: The system then constructs a user-item matrix, where each row represents a user and each column represents an item. The entries in the matrix represent the interactions between users and items.

- Similarity calculation: The system calculates the similarity between users based on their interactions with items. There are various similarity metrics that can be used, such as cosine similarity, Pearson correlation, or Jaccard similarity.

- Nearest neighbors identification: The system identifies the K nearest neighbors of the target user based on their similarity scores. These are the users who are most similar to the target user based on their past interactions with items in the system.

- Rating prediction: The system predicts the rating that the target user would give to each item based on the ratings that the nearest neighbors have given to that item. This is done by taking a weighted average of the nearest neighbors' ratings, where the weights are the similarity scores.

- Recommendation: The system recommends the top-rated items to the target user.

7.5 Hybrid Model

The hybrid model implemented was the neural collaborative filtering model, which combines neural networks with the collaborative filtering model. Neural collaborative filtering (NCF) is a recommendation algorithm that uses neural networks to learn the latent representations of users and items in a recommendation system. It combines the power of both collaborative filtering and deep learning to provide accurate and personalized recommendations to users. Here's how the neural collaborative filtering model works:

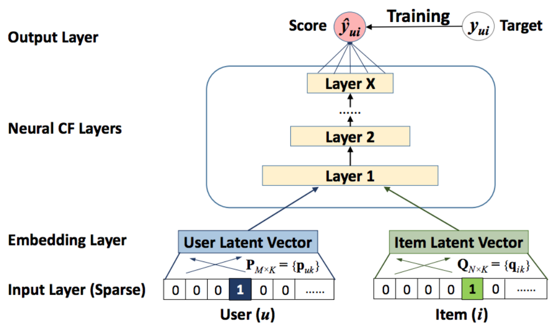


Figure 8: Architecture of Neural Collaborative Filtering Model

- User-item interaction data: The system collects data on the interactions between users and items. This can include ratings, reviews, clicks, purchases, or any other form of user-item interaction data.

- User-item matrix: The system constructs a user-item matrix, where each row represents a user and each column represents an item. The entries in the matrix represent the interactions between users and items.

- Neural network training: The system trains a neural network to learn the latent representations of users and items. The input to the network is the user-item interaction matrix, and the output is a predicted rating or score for each user-item pair. The network consists of several layers of neurons that transform the input data into a lower-dimensional representation, where the interactions between users and items are better captured.

- Loss function optimization: The system optimizes a loss function that measures the difference between the predicted ratings and the actual ratings. The optimization process adjusts the weights of the neural network to minimize the loss function, which improves the accuracy of the predicted ratings.

- Recommendation: The system recommends the top-rated items to the target user based on the predicted ratings.

Hyper-parameters used:

- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 10
- Batch size: 64
- Embedding size: 50

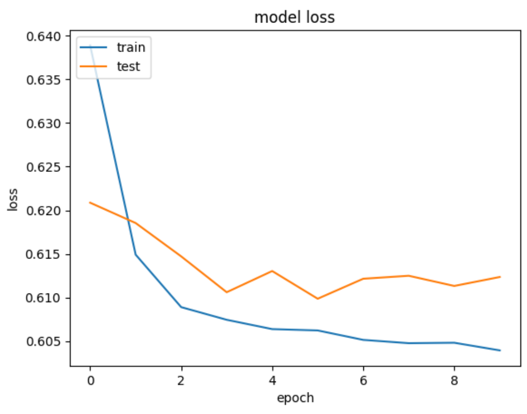


Figure 9: Model Loss in testing and training phase

NCF can overcome some of the limitations of traditional collaborative filtering methods, such as the cold-start problem and sparsity of the user-item matrix, by using neural networks to learn the complex relationships between users and items. It can also incorporate various types of data, such as text, images, or audio, to improve the quality of recommendations. However, it can be computationally expensive to train and optimize the neural network, especially for large datasets.

8 Results

The quality of the predicted ratings in each of the models was evaluated using the **Root Mean Square Error (RMSE)** and **Mean Absolute Error (MAE)** evaluation metrics.

Mean Absolute Error(MAE) computes the sum of the deviations between actual ratings and predicted ratings:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i| \tag{6}$$

Root Mean Square Error (RMSE) computes the square root of the sum of the squares of the deviations, thus putting more emphasis on larger deviations:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2} \tag{7}$$

The values are summarized in the following table:

Algorithm Name	RMSE	MAE
User-user	1.01	0.82
Item-item	0.96	0.76
KNN Based	0.98	0.79
Neural	0.94	0.73
Content Based	1.08	0.85
Surprise - SVD	0.87	0.69

Table 2: RMSE and MAE values for the Models

We plotted a graph comparing the values of these metrics for each of the models, which is shown in Figure 10.

For evaluating the quality of the list of movie recommendations generated by each of the models, **Precision**, **Recall**, **F-measure**, and **Normalized Discounted Cumulative Gain (NDCG)** were calculated.

Precision is a measure of exactness and determines the fraction of relevant items recommended out of all items recommended:

$$Precision = \frac{tp}{tp + fp} \tag{8}$$

Recall is a measure of completeness and determines the fraction of relevant items recommended out of

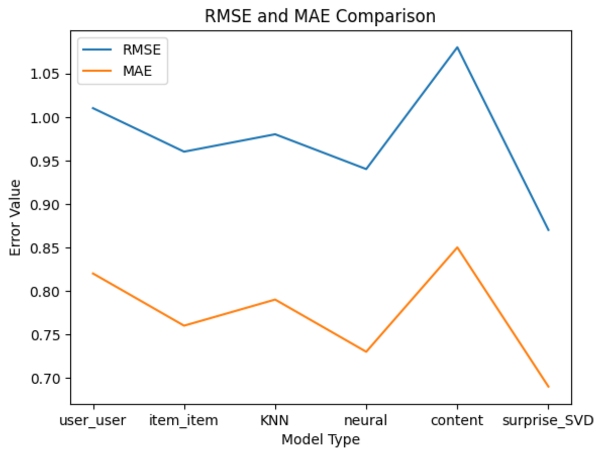


Figure 10: Comparison of RMSE and MAE values for each of the models

all relevant items:

$$Recall = \frac{tp}{tp + fn} \tag{9}$$

Here, tp = number of true positives, fp = number of false positives, fn = number of false negatives.

The F-measure combines precision and recall into a single value, it is the harmonic mean of precision and recall and gives a more balanced view of the performance.

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{10}$$

NDCG stands for Normalized Discounted Cumulative Gain. NDCG measures the quality of the ranked list of recommended items by computing the sum of relevance scores of the recommended items, discounted based on their position in the list. First we compute Discounted Cumulative Gain (DCG) as follows:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{11}$$

where p denotes the position up to which relevance is accumulated, rel_i returns the relevance of recommendation at position i. Next, we compute Idealized Discounted Cumulative Gain (IDCG) as follows:

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{12}$$

Finally,

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{13}$$

The values are summarized in the following tables:

Algorithm Name	Precision	Recall	F-measure
User-user	0.89	0.56	0.69
Item-item	0.92	0.60	0.72
KNN Based	0.90	0.58	0.70
Neural	0.94	0.64	0.77
Content Based	0.82	0.53	0.64
Surprise - SVD	0.88	0.65	0.66

Table 3: Precision, Recall, and F-measure values for the Models

Algorithm Name	NDCG
User-user	0.96
Item-item	0.97
KNN Based	0.97
Neural	0.99
Content Based	0.94
Surprise - SVD	0.99

Table 4: NDCG values for the Models

Figure 11 shows the graph comparing the values of these metrics for each of the models.

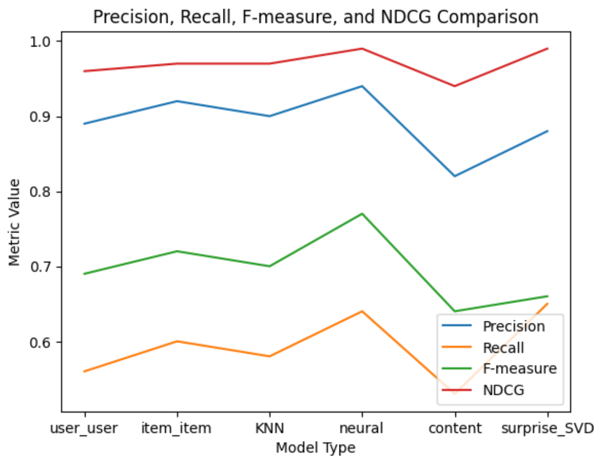


Figure 11: Comparison of Precision, Recall, F-measure, and NDCG values for each of the models

From Figure 10, we can see that the RMSE and MAE values are highest for the content-based recommendation system. The rest of the approaches, all of which are collaborative filtering-based approaches, have lower values for both of these errors. Among the collaborative filtering-based approaches, we can see that item-item collaborative filtering has lower values of both errors than user-user collaborative filtering, indicating better performance. The values for the KNN-based collaborative filtering approach are somewhere in between those of the item-item and user-user based approaches. We also observe that the SVD-based approach and the neural collaborative filtering approach have the lowest values of both the errors. Our observations in Figure 11 are consistent with this. The precision, recall, F-measure, and NDCG values are higher for the models having lower values of the RMSE and MAE, and vice versa.

The better performance of the collaborative filtering recommendation systems can be explained by the fact that the Movielens dataset has a large amount of user-item interaction data. The availability of a large amount of user-item interaction data allows the recommendation system to identify patterns in user ratings more accurately, and makes it easier for the system to make more precise recommendations. Also, in the content-based approach, it is often difficult to identify the appropriate features of the items to find the similarity between items. We have only utilized the 'genre' feature of the movies in order to make recommendations in the content-based approach. This may have resulted in lower performance of the content-based approach, as the system is limited to finding similarities between movies only based on the genre.

Among the collaborative filtering-based approaches, we see that item-item collaborative filtering has achieved better results than user-user collaborative filtering. Generally, item-item collaborative filtering works better than user-user collaborative filtering when the number of items in the dataset is greater than the number of users. We can attribute this to the fact that when we have more items, it is easier to identify similarities between items. This is because, with a greater number of items, each user can interact with more items, which makes it easier to identify patterns in user preferences for determining the similarity between items. Also, with a greater number of items, the set of items is more diverse, which facilitates finding similarities between items. On the other hand, in user-user collaborative filtering, identifying similar users based on interactions with items is difficult when there are more items than users. This leads to less accurate recommendations. In our Movielens dataset, the number of movies(9K) is much greater than the number of users (600). Therefore, we observe better performance of the item-item based approach. The KNN-based approach to collaborative filtering is found to have comparable performance to both item-item and user-user collaborative filtering, and is somewhere in between them in terms of performance.

The SVD-based model built using the surprise library and the neural collaborative filtering model have the best performance. The high performance of the SVD-based model can be attributed to the fact that SVD captures the latent factors in the user-movie interaction data (such as user preferences in movie genre) even though the movie ratings data may not explicitly exhibit these preferences. This is due to the fact that SVD decomposes the user-item matrix into the user-factor, factor-factor and item-factor matrices. Here,for example, the factors (latent factors) could be the strengths of different genres. The user-factor matrix would exhibit the strength of correlation of each user with each genre, while the item-factor matrix would display the strength of correlation of each item with each genre. Since these latent factors are captured, it becomes easier to make more accurate recommendations that are better suited to users' preferences.

Similarly, the neural collaborative filtering model also learns the latent factors in the user-movie interaction data, the only difference is that while SVD achieves this through matrix factorization, neural collaborative filtering achieves this by learning the latent factors through a deep neural network. Therefore, this results in very good performance of neural collaborative filtering. We observe that the performance of neural collaborative filtering is even slightly better than that of SVD-based collaborative filtering. This may be due to the fact that neural collaborative filtering can capture non-linear relationships between users and items, which is not possible in SVD as it is a linear model.

9 Conclusion and Future Work

In this project, we implemented a number of movie recommendation approaches on the Movielens dataset and evaluated their performance. We found that the collaborative filtering-based approach generally performs better than the content-based approach, since we have a large amount of user-item interaction data. The content-based approach would work well in the case when we do not have much user-item interaction data, i.e. it helps to overcome the cold start problem. We implemented the content-based approach using only genre as a feature for checking similarity.
A good point for future scope is to try the content-based approach with other features as well, such as taglines, in order to improve the performance. Among the collaborative filtering-based approaches, we see that the item-item collaborative filtering model performs better than the user-user model, as there are more items (movies) in the Movielens dataset than there are users. We used a threshold rating of 3.5 to check if the recommended movies are relevant. Varying this threshold value to find the optimal values of the evaluation metrics is another good point for future scope. The performance of the KNN-based approach is somewhere between that of the item-item and user-user approaches. The SVD and neural collaborative filtering based approaches perform the best, due to their ability to identify latent factors in the user-item interaction data.

10 Acknowledgement

We would like to thank Prof. Yongfeng Zhang and our two TAs, Sam Lin and Zejun Xie for giving us the opportunity to work on this project and also for the guidance and all the necessary support that we needed in order to complete it.

References

1. AGGARWAL, CHARUC (2018) Recommender systems: The textbook. SPRINGER.

2. AL-Ghuribi, S.M. and Noah, S.A.M. (2021) A comprehensive overview of recommender system and sentiment analysis, arXiv.org. Available at: https://arxiv.org/abs/2109.08794 (Accessed: May 1, 2023).

3. Badrul Sarwar GroupLens Research Group/Army HPC Research Center et al. (2001) Item-based collaborative filtering recommendation algorithms: Proceedings of the 10th International Conference on World Wide Web, ACM Conferences. Available at: https://dl.acm.org/doi/10.1145/371920.372071 (Accessed: May 1, 2023).

4. S. Chawla, S. Gupta and R. Majumdar, ”Movie Recommendation Models Using Machine Learning,” 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-6, doi: 10.1109/ISCON52037.2021.9702472.

5. Mu Y, Wu Y. Multimodal Movie Recommendation System Using Deep Learning. Mathematics. 2023; 11(4):895. https://doi.org/10.3390/math11040895

6. Harper, F.M. and Konstan, J.A. (2016) The movielens datasets: History and context: ACM Transactions on Interactive Intelligent Systems: Vol 5, no 4, ACM Transactions on Interactive Intelligent Systems. Available at: https://dl.acm.org/doi/10.1145/2827872 (Accessed: May 1, 2023).

7. Hug, N. (2020) Surprise: A python library for Recommender Systems, Journal of Open Source Software. Available at: https://joss.theoj.org/papers/10.21105/joss.02174 (Accessed: May 1, 2023).

8. Koren, Y. et al. (2009) Matrix factorization techniques for Recommender Systems, Computer. Available at: https://dl.acm.org/doi/10.1109/MC.2009.263 (Accessed: May 1, 2023).

9. Leskovec, J., Rajaraman, A. and Ullman, J.D. (2022) Mining of Massive Datasets. Cambridge etc.: Cambridge University Press.

10. Alireza Abdolmaleki Mohammad Hossein Rezvani (2022) An optimal context-aware content-based movie recommender system using genetic algorithm: a case study on MovieLens dataset, Journal of Experimental Theoretical Artificial Intelligence, DOI: 10.1080/0952813X.2022.2153279

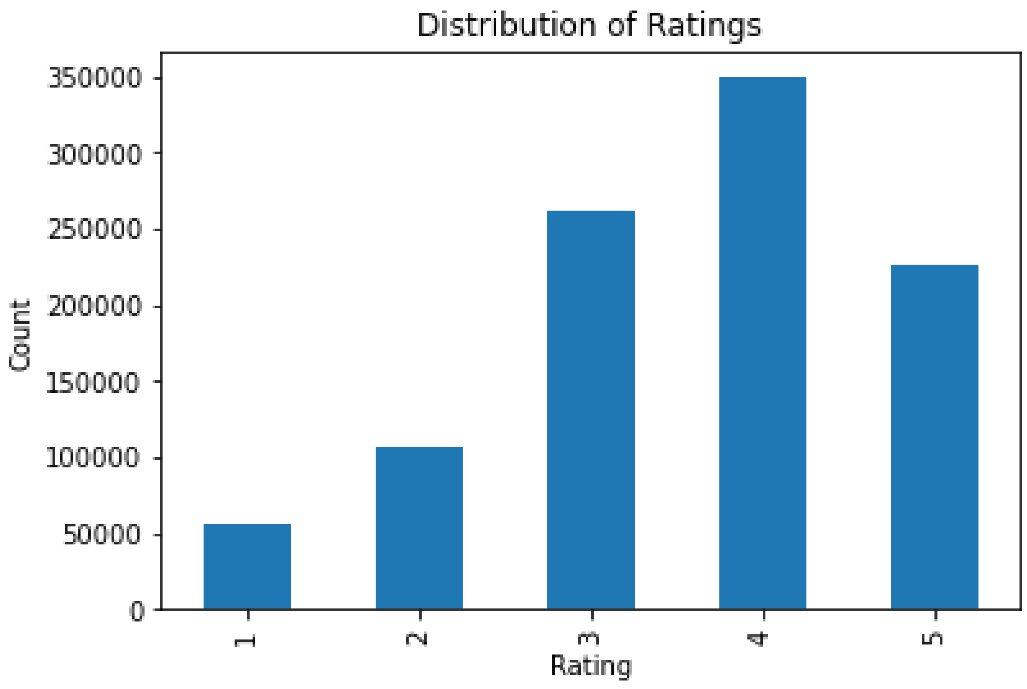


Figure 12: Distribution of Ratings

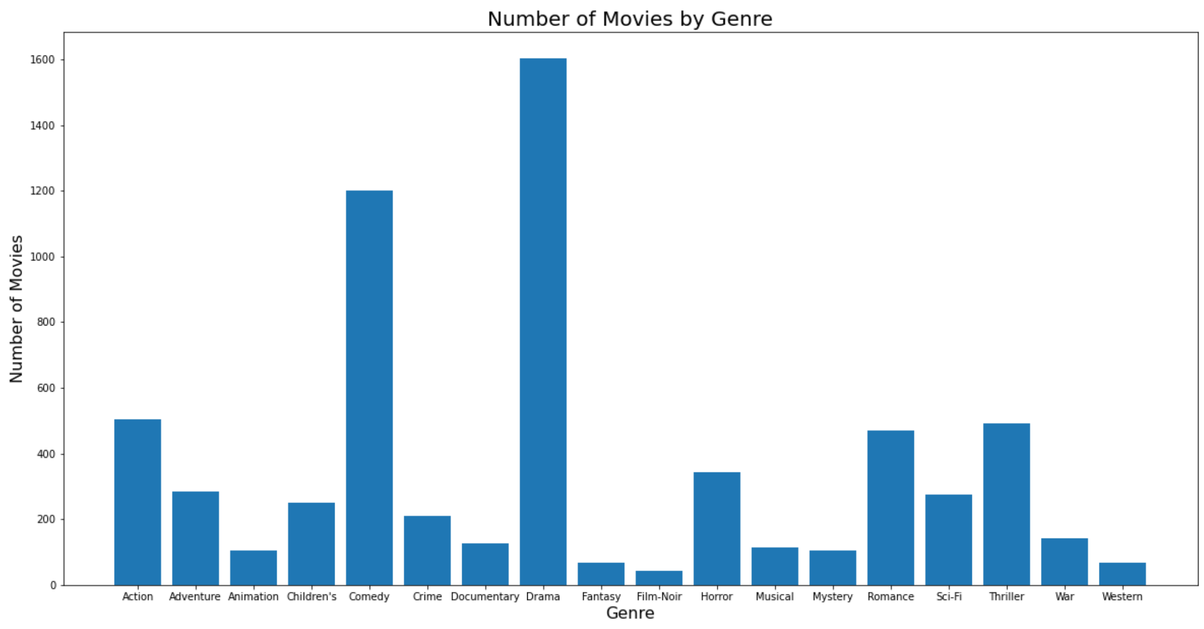


Figure 13: Number of movies by genres

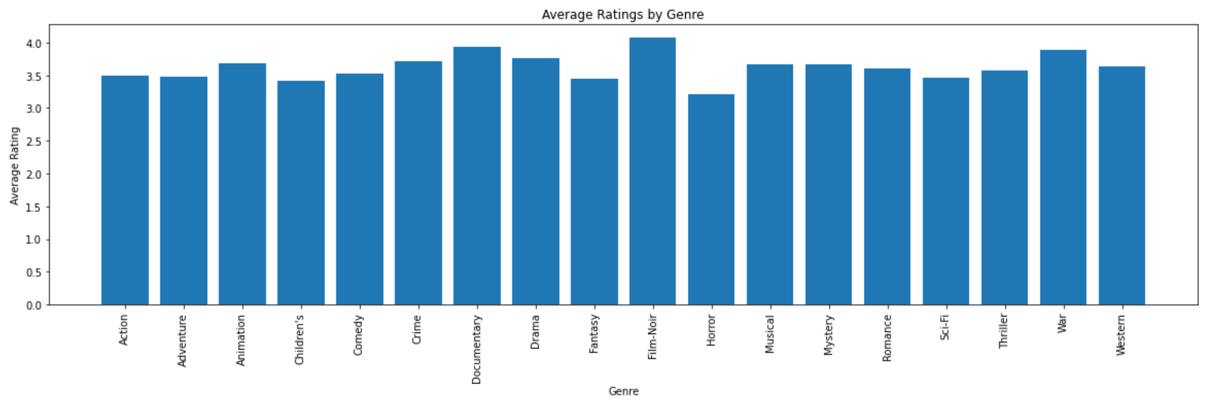


Figure 14: Average ratings by genres