# Movie Recommendation System

Siddharth Devendra Gupta, Dhruvi Lalit Jain
Rutgers University
siddharth.gupta1@rutgers.edu,dhruvilalit.jain@rutgers.edu

## ABSTRACT

**Abstract:** Nowadays, recommendation systems are commonly used in various online platforms. The goal of our project is to create a movie recommendation system using the "MovieLens" dataset. Our objective is to personalize recommendations based on user preferences while considering features like genre and popularity of movies. We have built several models including a popularity model, content-based model, collaborative filtering model, and latent factor based model. We carefully tuned hyperparameters, tested accuracy, and evaluated each model's recommendations. By combining the predictions of the latent factor based and collaborative filtering models, we were able to create a linear model that showed an improvement in predicted rating accuracy. We then created a hybrid model by combining all the methods to generate a final recommendation list for each user, which performed better in terms of quality and diversity of recommendations compared to individual models. We provide a detailed analysis of each model in our report.

## KEYWORDS

**Keywords:** movie recommendation, content-base, collaborative filtering, latent factor, SVD, hybrid recommendation

## 1 INTRODUCTION

With the large amount of growth in data, and capturing the consumer behaviour, one of the most important aspects of today's technologies is providing personalized services in the form of recommendation systems.

## 2 RELATED WORK

Although there are multiple approaches to solve the problem of recommendation systems, the two most commonly used approaches are Collaborative Filtering and Content Based Filtering. However, each has its shortcomings and hence in year 2005, Adomavicius came up with a hybrid approach, merging both methods therefore reducing the limitations of them individually. Today, hybrid approach is the most commonly used method for recommendation systems. Singular Value Decomposition is one such commonly used algorithm that is used to perform Collaborative Filtering. SVD performs Matrix Factorization on the user- movie matrix to generate the corresponding decomposition vectors showing similarity between movies. A brief review of the related work. (Here is an example of how to include citation.


*Conference'17, Washington, DC, USA*

DOI: 10.1145/nnnnnnn.nnnnnnn

## 3 RATING EVALUATION PARAMETERS

### 3.1 Rating Evaluation parameters

*3.1.1 Root Mean Square Error (RMSE):.* Root Mean Square Error is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data, that is, how close the observed data points are to the model's predicted values. It is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed. The smaller an RMSE value, the closer the predicted and observed values are or the closer you are to finding the line of best fit. RMSE is the square root of the average of squared errors. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Formula to calculate RMSE:

$$RMSE = \sqrt{(1/n)\sum_{i=1}^{n}(y_i - x_i)^2}$$

where,
n is the number of data points,
$y_i$ is the predicted value,
$x_i$ is the actual/observed value

*3.1.2 Mean Absolute Error:* The Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. The MAE is a linear score which means that all the individual differences are weighted equally in the average. It tells us how big of an error we can expect from the prediction model on average. The MAE will always be lesser than or equal to MAE. If all the errors have equal magnitude, then MAE = RMSE. The lower the value of MAE, the higher is the accuracy of the prediction model. Formula to calculate MAE:

$$MAE = 1/n\sum_{i=1}^{n}(y_i - x_i)$$

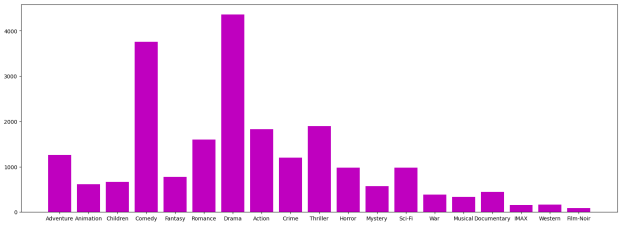
## 4 DATA

The primary dataset used for this project is the grouplens movie review dataset. The smaller dataset (ml-latest- small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. The dataset has a collection of 100,000 reviews over 9000 different movies by 600 users. Each rating being a value between 0-5. Some features of the dataset include the timestamp of the review, genre of the movie and the IMDB and TMDB id for the corresponding movie

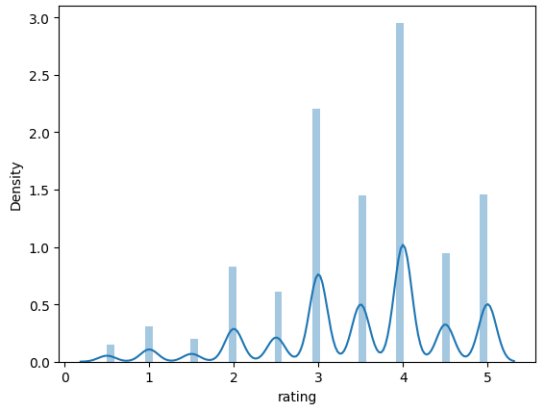
## 4.1 Data Analysis

A series of steps were performed to analyse the data and its properties.

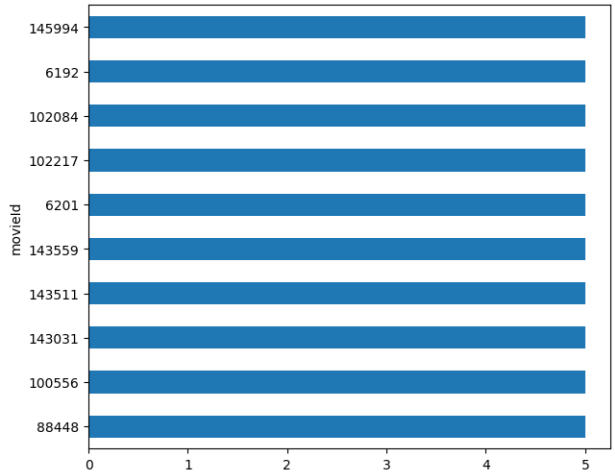
Below is a bar plot of the number of movies of each genre. An important point to note here is that a movie can have multiple genres and hence simply taking a summation of the number of movies in each genre does not give us the total number of movies.

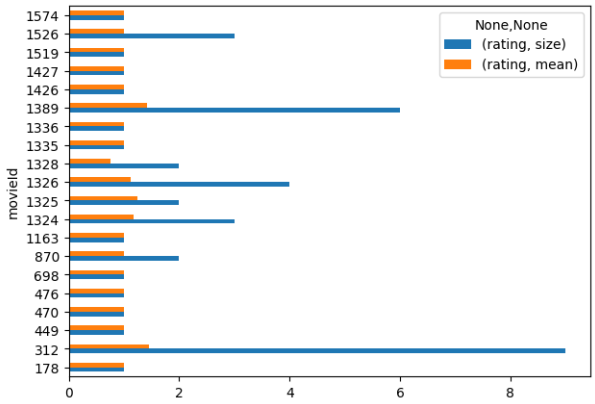



We also tried to observe the density distribution pattern of ratings. This helps us identify the most commonly rated value in the ratings. Below is the graph for the same.




We tried to find out a list of movies with high average rating and low average rating. Below are the graphs for the same.




## 4.2 Data Preprocessing

We split the dataset into test and train on the basis of UserID, such that 80 percent of the reviews by every single user is used for training the model and the remaining 20 percent of the reviews to validate the correctness of our model. We achieve the above using the train test split method in scikit-learn library. We are making use of the Surprise library to train the model which cannot interpret the train and test sets generate by the scikit-learn library. Hence we convert the train and test models into the Surprise format.

# 5 PROBLEM FORMALIZATION

Mathematically formalize your research problem.

# 6 MODEL

## 6.1 Matrix Factorization Using Pytorch

In PyTorch, we can implement matrix factorization using a neural network. The goal of matrix factorization is to decompose the user-item interaction matrix into two low-dimensional matrices, one for users and one for items. These low-dimensional matrices are called embeddings and are learned during training.

To implement matrix factorization using PyTorch, we can create a neural network with two embedding layers, one for users and one for items. Each embedding layer takes in a one-hot encoded input vector, which represents a user or an item, and outputs a low-dimensional embedding vector. These two embedding vectors are then multiplied together to generate a predicted rating for the user-item pair.

To train this model, we would pass in batches of user-item pairs and their corresponding ratings, compute the loss using MSE, and update the embedding vectors using SGD. During training, we use mean squared error (MSE) as the loss function to measure the difference between the predicted ratings and the actual ratings. We can then use stochastic gradient descent (SGD) or another optimization algorithm to update the embedding vectors and minimize the loss.

The evaluated metrics for the above model was as follows:

| Metric | Matrix Factorization (Pytorch) |
|---|---|
| RMSE | 0.36 |
| MAE | 0.45 |
| Precision | 0.69 |
| Recall | 0.46 |
| F1-score | 0.49 |
| NDCG | 0.57 |

## 6.2 Content Based Recommendation

The basic idea of Content based recommendation system is to find similar items based on the different attributes present in the dataset. For example, in the Movielens dataset, similar movies can be found by studying the list of genres for each movie.

*6.2.1 Term Frequency (TF) and Inverse Document Frequency (IDF).* We use the TF-IDF vectorizer and cosine similarity to recommend movies based on genres. We first initialize a TfidfVectorizer object, which tokenizes the genre information for each movie and computes the TF-IDF values for each token. The resulting TF-IDF matrix can be then used to compute cosine similarity between movies based on their genre information.
We created a function that takes a movie title as input, finds the corresponding movie in the dataset, and calculates the cosine similarity between that movie and all other movies based on their genre information. The function then returns the top 10 movies with the highest similarity scores.
The model is evaluated by computing the precision, recall, F1-score and NDCG.

| Metric | Content Based (Genre) |
|---|---|
| Precision | 0.95 |
| Recall | 0.99 |
| F1-score | 0.96 |
| NDCG | 0.95 |

*6.2.2 User-Item.* User-Item Filtering is another common approach used in recommendation systems. It involves building a model that predicts the rating a user would give to an item based on their past ratings and other factors, such as item attributes or user demographics. The model can then be used to generate personalized recommendations for the user.
We implement User-Item Filtering by constructing a movie vector and a user vector. The length of these vectors is equal to the number of unique genres. The movie vector is a one hot encoding for each movie representing 1 against the genre it belongs to and rest all as 0. The user vector indicates the average rating given by the user to the particular genre of movies. To predict a rating, the corresponding movie vector is multiplied by the user vector.

| Metric | Matrix Factorization (Pytorch) |
|---|---|
| RMSE | 0.92 |
| MAE | 0.71 |
| Precision | 0.80 |
| Recall | 0.50 |
| F1-score | 0.61 |
| NDCG | 0.95 |

## 6.3 Collabarative Filtering

Collaborative filtering uses the ratings to form a neighbourhood N of a user, say X, whose ratings (likes and dislikes) are similar to X's ratings. The neighborhood of X is defined using nearest neighbor approach with a notion of similarity defined. There can be two types of collaborative filtering user-user and item-item. This method is different from content based as the neighborhood or similarity is defined only on the basis of rating behaviour. The following are the experiments done for collaborative filtering algorithm.

*6.3.1 Item-Item.* Item-Item filtering is a common approach used in recommendation systems that aims to recommend items (in this case movies) based on similarities between them. The approach involves calculating the similarity between items based on the ratings that users have given them, and then recommending items that are similar to the ones that the user has already liked. We have used cosine similarity as a metric to calculate the similarity measure.
We create a pivot table from the ratings dataframe with userId as the row index, movieId as the column index, and rating as the cell value. This creates a matrix where each row represents a user and each column represents a movie, with the user's ratings for each movie in the corresponding cell. Based on the rating's matrix, we further compute the cosine similarity of different movies.

*6.3.2 User-User.* User-User Collaborative Filtering is a technique used in recommendation systems to suggest items or products to users based on the similarity between their preferences and the preferences of other users. In this approach, a similarity metric is used to identify users who have similar tastes and preferences. Then, the system recommends items that the similar users have liked or consumed in the past.

## 6.4 Singular Value Decomposition

We want our recommendation model to predict the unknown ratings. For this we use the latent factor method, SVD. SVD takes the lower dimensional representation of movies such that people who like similar movies together are mapped together. It discovers the features (hidden latent factors) such that movies can be mapped into the same space as user. We can use the SVD model to form an optimization problem to minimize the RMSE for unseen test data. Thus, using gradient descent algorithm and regularization allows rich model structure.
SVD stands for Singular Value Decomposition and is a mathematical technique used to decompose a matrix into three matrices - U, S, and V, where U and V are orthogonal matrices and S is a diagonal matrix with singular values. In the context of collaborative filtering, SVD can be used to reduce the dimensionality of the utility matrix, which is typically large and sparse, by identifying latent features that explain the variance in the ratings. By truncating the SVD matrices to include only the top K singular values, where K is a hyperparameter, we can create a low-rank approximation of the utility matrix that captures the most important latent features. This approximation can then be used to fill in missing values in the utility matrix and predict ratings for new items. SVD is a powerful tool for recommendation systems because it can handle missing data and reduce the dimensionality of the utility matrix, making it

easier to compute and faster to compute the recommendations.
The code starts by loading two datasets, ratings.csv and movies.csv, and merges them into a single dataset. This merged dataset is then split into training and testing sets, with each user's data randomly split into 80 percent training and 20 percent testing data. The code then obtains a list of unique users and movies from the merged dataset and creates a utility matrix, where each row represents a movie, each column represents a user, and the values are the corresponding ratings. Missing values in the utility matrix are then filled using singular value decomposition (SVD), which calculates the mean of the ratings for each user and replaces the missing values with the corresponding mean ratings. The filled matrix is then normalized and centered at zero. Finally, the code predicts the ratings for the testing set using the truncated SVD matrices obtained in the previous step and calculates the root mean square error (RMSE) to compare the predicted ratings with the actual ratings. Overall, this code provides a simple but effective way to build a movie recommendation system using collaborative filtering.
The RMSE and MAE for the model created using SVD is as below:

| Metric | Value |
|---|---|
| RMSE | 2.37 |
| MAE | 1.98 |

### 6.5 K-Nearest Neighbours

The basic idea behind KNN is to find the k closest data points in the training set to a new input data point, and then use the most common class or average value among those k neighbors to classify or predict the output of the new data point.

The "k" in KNN represents the number of nearest neighbors to consider. For example, if k=3, then the algorithm would consider the three nearest data points to the new input point, and the majority class or average value of those three neighbors would be used to classify or predict the output of the new data point.
The RMSE and MAE for the model created using KNN is as below:

| Metric | Value |
|---|---|
| RMSE | 0.85 |
| MAE | 0.71 |

## 7 CONCLUSIONS AND FUTURE WORK

The Content Based model is decent model for generating recommendations and similarity between items and users. However, they lack personalization for a given user. Collabarative Filtering and SVD improves the accuracy of the predicted ratings as seen in the combined model. A hybrid model of combining all the models discussed above would be beneficial. The individual weights for each model can be determined by implmenting a simple gradient descent and trying all possible combinations of weights which results in the best results.

## ACKNOWLEDGEMENT

We would like to thank the Professor Mr. Yongfeng Zhang and the teaching assistant Mr. Sam Lin for their guidance as well as for providing necessary information regarding the project and also for their support in completing the project.

## 8 REFERENCES

Herwijnen, O. V. (2019). Collaborative filtering for movie recommendations using the MovieLens dataset. Retrieved from https://arxiv.org/abs/1908.09353
Shahzad, F., Javaid, A., and Haider, M. A. (2020). A hybrid movie recommendation system using collaborative filtering and content-based filtering techniques. International Journal of Advanced Computer Science and Applications
Gomathi, R., and Vaidyanathan, S. (2020). A comparative analysis of movie recommendation systems using the MovieLens dataset. International Journal of Scientific Research in Computer Science, Engineering and Information Technology.
Deng, C., Liu, C., Li, J. (2021). A hybrid approach for movie recommendation using matrix factorization and deep neural networks. In Proceedings of the International Conference on Computing, Networking and Communications.