Movie Recommendation System

Capstone Project by Brian Ho

# Introduction

Over the past centuries, movies have been affecting many generations of people. Movies such as Godfather, Star Wars, Star Trek, Lord of the Ring, Titanic, Fast and Furious and more have showcased many story to the public and not only entertain audience, but also educate audience. Some people’s ambition are actually affected by movies, such as people who watch Star Trek are interested in astronomy and people who watch Willy Wonka and the Chocolate Factory are interested in creation and invention. The preference of the types of movies reflect the person’s interest and the kind of subject that draw his attention.

For business like Netflix, Amazon, Youtube or movie renting businesses, they would be interested to have users to re-use their service and it is necessary to attract them with the right kind of movie that are interested. They can use different marketing techniques like emails campaigns or Facebook ads to show new movies that this user will be interested. This becomes important to explore movie recommendation system as identifying which movie will the user be most interested is out of thousands of movies is not easy.

It is nature that all these companies are trying to understand more about the users and using all data they can capture to predict. This project is an attempt to predict how users will rate unseen movies based on users’ previous rating on different movies.

# Data

The data set has been obtained from GroupLens Search as they collected it from MovieLens website (https://grouplens.org/datasets/movielens/).

The dataset this project uses have 100,000 ratings from 943 users on 1682 movies. It has additional data about the genre of the movies, users’ age and users’ occupation. In the core dataset, each row of data is a movie rating from one user towards one movie at a date. The additional data information are all indexed by user id or movie id, which dataset can be joined to create one large dataset for different type of data analysis and machine learning modelling.

# **Data Dictionary**

User-Movie Ratings

|  |  |
| --- | --- |
| Column Name | Description |
| User\_id | Unique ID for the User |
| Movie\_id | Unique ID for the Movie |
| Rating | Ratings from user to the movie |
| Timestamp | The time the user give out the rating |

Movie Genre

|  |  |
| --- | --- |
| Column Name | Description |
| Genre\_id | Unique ID for the Genre |
| Genre | Genre Name |

User

|  |  |
| --- | --- |
| Column Name | Description |
| User\_id | Unique ID for the User |
| Gender | User’s gender |
| Occupation | User’s occupation |
| zipcode | User’s zipcode |

Movie

|  |  |
| --- | --- |
| Column Name | Description |
| Movie\_id | Unique ID for the Movie |
| Title | Movie Title |
| Release\_date | Movie Release Date |
| Video\_release\_date | Date movie available to be watched |
| Imdb\_url | IMDB official page for the movie |

# Tool Used

**Pandas**: Loading data, data wrangling and manipulation, feature engineering

**Scikitlearn**: libraries for test train split, metrics,

**Scipy**: Library for Singular Vector Decomposition (SVD)

**Matplotlib**: Data Visualization

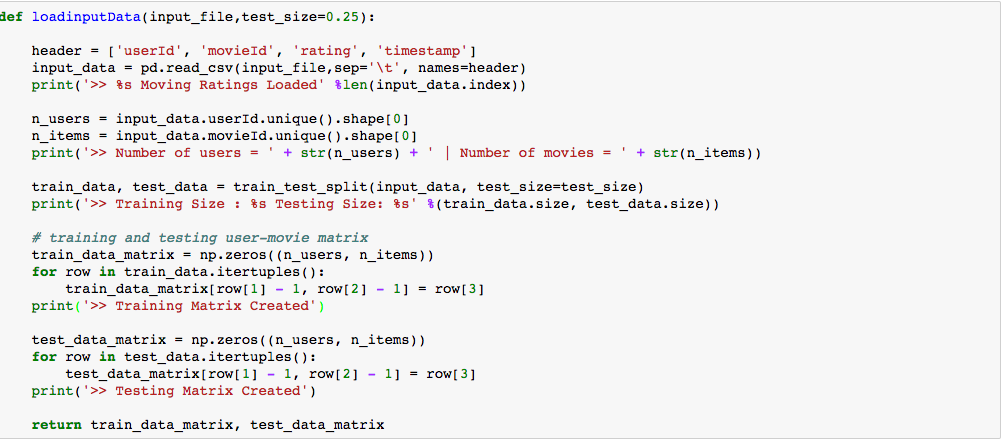
# Data Wrangling

Using pandas, movie rating dataset was imported and read. The dataset contains ratings that users have already given. There are no missing values in any columns.

The following shows the command to load the data provided by GroupLens



3 data files are loaded into 3 pandas Dataframe and merges into 1 data frame. This will be easier to manage and explore the dataset. It makes split-apply-combine easier to utilize the built-in matplotlib plotting function in Pandas to visualize the data.



For the purpose of building the recommendation system, the data will be loaded and transformed in the different method. A user-movie rating matrix will be built .

Using a function that take an input csv file. It will split data into training data and testing data.

The main transformation is building the matrix with number of users by number of movies. The cell will be the rating a user give to a movie and this matrix store all the ratings. There will be a lot of 0 (blank) cells that indicate movies that users haven’t watched or rated.

This follow code look at the sparsity, meaning how many of the cells have value, of the training dataset.



# Exploratory Analysis

Figure

I started off the analysis by looking at

the number of ratings that the top 25 movies that get is plotted. It was seen that some of the movies get more ratings which reflect the popularity. Also, using the total number of rating that a movie receives as the metrics to decide which movie to recommend to user may not be ideal. This would result all users getting the same recommended movies. This is shown in figure 1 where the movies with most number of ratings are presented from highest to lowest

Star Wars, Contact and Fargo come up as the top 3. It is also important to know if these top movies also have received high rating as well. In figure 2, I have combined both average ratings of the movies and number of ratings of the top 25 movies. Close Shave, Schindler and Wrong Trousers have the highest average ratings but they don’t have a lot of ratings, especially Schindlers. In contrast, Star Wars, Silence of the Lambs and Godfathers have high ratings and high number of ratings. This indicates these movies are favored by users and can be good movies to recommend to users.

Figure

Age of users is also an important feature in understand what movies each user prefer.

By look at the age distribution from the 943 users, most users are between the age of 20 to 30. This means the young generation watch more movies and can also potentially indicate they are more likely to watch new movies than old movies.

From a recommendation system point of view, it is important to identify how to classify users, especially when applying k-nearest neighbor. Should we take more weight on similar users whose age is also closer to the user that we are predicting.

At the meantime, combining this with the previous chart, Star Wars and Godfather are movies from 1977 and 1972 respectively but they both have high average ratings and high number of ratings. We can also potentially classify some movies as ‘legendary movies’ that would-be blockbuster in any generations.

From a production recommendation system perspective, it can also consider whether this movie is a series of movies and is this series still in-progress. Godfather has 3 movies and the last one ended in 1990. Star Movies has 7 movies and new ones are still being filmed and played. It may be reasonable to recommend Star Wars movies to users when a new Star Wars movie is about to air in the cinema. At the same time, if users have watched any of the Star Wars movie recently, it may be reasonable to recommend the other Star Wars movies as they may be interested in those as well. However, this would need a balance because we don’t want to recommend all Star Wars Movies.

Another analysis done is to group user into age group and look at movie rating. Figure 3 shows the number ratings in each movie by age group. The movie E.T, the Extra- Terrestrial. And God Father are not being rated by users aged between 70-79.

This shows age can be one of the features that determine what movie user decides to watch.



Figure 3



Apart from age, gender would also be an important factor. Figure 4 shows the difference in rating between male and female for each movie. This would help to explore if certain genre of movies are favored by certain gender. It is not easy to why certain movies are more favored by certain gender, the reason can be due to the director, the actor, the storyline or other elements of the movie that would be driving factors for users to watch and give a high rating.

To predict users’ rating towards a movie, we should know the importance of each factor/feature to each user and the weight of features exist in each movie. Using this as the weight to predict users’ rating would include more features into the prediction and hopefully increase prediction accuracy.

Figure

While we can try to collect as many information about the user and the movie, it is extremely to find out the weight of certain features to the contribution of the ratings. An example would be user A likes horror movie, a big fan of Lord of the Ring and obsessed with actor Leonardo Dicaprio. Each user would have their own preference and each movie also has its own combination of features. We have to figure out the weight of each features that contribute to the user and movie respectively

# Predicting the user’s movie rating

In the last section, I explored some of the features that would determine user’s rating towards movie. This brings an insight into what machine learning algorithm I can apply to take those findings into the prediction. The analysis from above have raised 2 important points that form the foundation of the algorithm used in this project.

* Find similar users who are similar in certain ways, e.g. age, preference, occupation, gender, etc. that reflect their ratings of the movie can be used for prediction.
* Finding the underlying features that contribute to the user and movie to use

There are many methodologies to compute similarity among users or movies as well as compute relative importance or ‘weight’ of certain feature. I used collaborative filtering and singular vector decomposition respectively to implement the two models. To get a threshold of the result of the algorithms, I use a mean rating of all user’s previous rating (user-based) to predict the next rating or use rating that a movie received from other users to predict the next rating it will receive(item-based). If the model that I built is below this threshold, it is not a good model.

## Evaluation Method

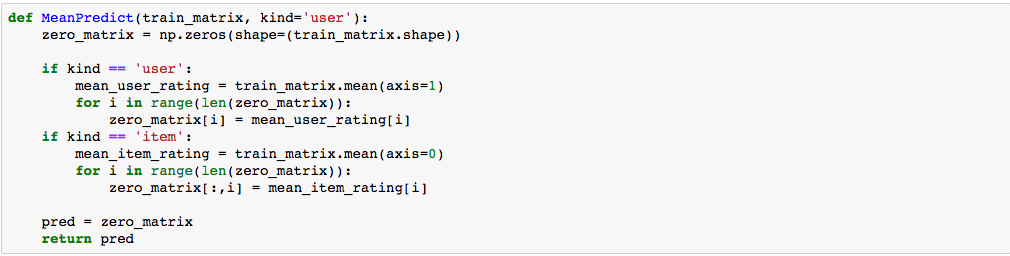
I use Root Mean Squared Error (RMSE) as the evaluation method. The dataset is split into training set and dataset as shown. RMSE will be used for the model the predict what rating user will give to a movie from the test set and compare the predicted value with the true value. The goal is to get as close as 0, meaning there’s predicted value equals to the true value.

## Predicting threshold – Mean Rating

Before discussing about the model I anticipate the implement, I want to find a baseline prediction method. This is useful for establishing non-personalized baselines against which

personalized algorithms can be compared, as well as for pre-processing and normalizing data for use with more sophisticated algorithms. Baseline algorithms that do not depend on the user’s ratings can also be useful for providing predictions for new users. The intuition is if the prediction model that I built give a higher score than the mean rating model, that means the prediction has less prediction accuracy and use a mean model can get better prediction in general. Mean model become the threshold of the project.

The mean model has been implemented as user-based mean rating and item-based mean item.



Figure

Figure 5 shows that a mean prediction function takes in a parameter of ‘user’ or ‘item to determine which model to use. As the training data is a matrix with item as column and user as rows, the line *train\_matrix.mean(axis=1)* and *train\_matrix.mean(axis=0)* determine how to get the mean value the user or the items. It takes the global mean of the user or movie as prediction. Baseline predictors effectively capture effects of user bias, item popularity, and can be applied to more exotic but increasingly-important factors such as time.

../../../../Desktop/Screen%20Shot%202017-04-17%20at%209.54.02%2

Figure

This returns a score of 3.42 and 3.23 respectively of user-based and item-based model. We seek our model to return a lower RMSE score than this.

## Model 1 - Collaborative Filtering

Our first model is collaborative filtering. It is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behavior of other users in the system. The fundamental assumption behind this method is that other users’ opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active user’s preference (user-user collaborative filtering). Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items. There is another method

for performing recommendation, such as finding items similar to the items liked by a user using textual similarity in metadata (item-item collaborative filtering).

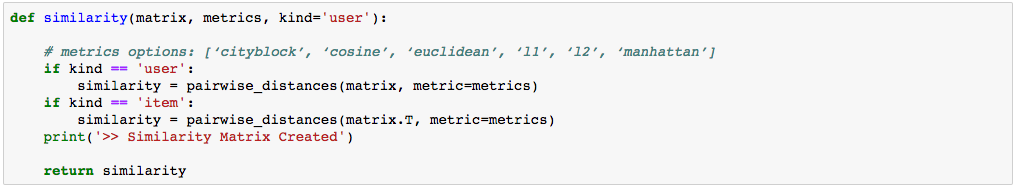
User–user collaborative filtering, also known as k-NN collaborative filtering, was the first of the automated CF methods-

User–user CF is a straightforward algorithmic interpretation of the core premise of collaborative filtering: find other users whose past rating behavior is similar to that of the current user and use their ratings on other items to predict what the current user will like. An example is to predict Mary’s preference for an item she has not rated, user–user CF looks for other users who have high agreement with Mary on the items they have both rated. These users’ ratings for the item in question are then weighted by their level of agreement with Mary’s ratings to predict Mary’s preference

Item–item collaborative filtering is another interpretation of collaborative filtering. Rather than using similarities between users’ rating behavior to predict preferences, item– item CF uses similarities between the rating patterns of items. If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items. In its overall structure, therefore, this method is similar to content-based approaches to recommendation and personalization, but item similarity is deduced from user preference patterns rather than extracted from item data. In a system that has more users than items, it allows the neighborhood-finding to be amongst the smaller of the two dimensions, but this is a small gain. It provides major performance gains by lending itself well to pre-computing the similarity matrix.

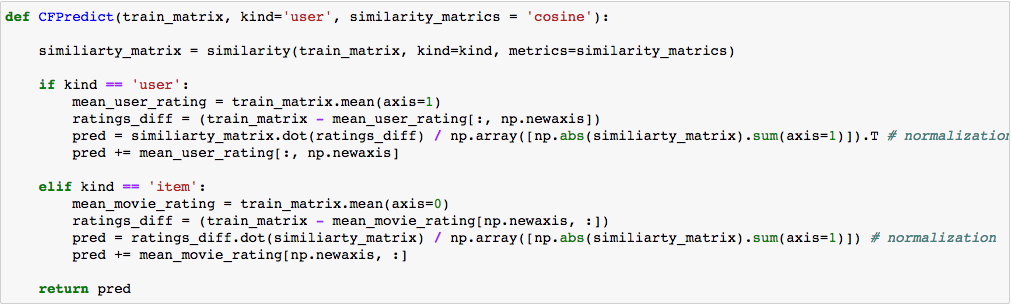
Collaborative filtering requires a similarity function to compute the similarity between two users and a method for using similarities and ratings to generate predictions. There are several popular similarity functions and are available in scikitlearn metrics

* Pearson correlation
  + This method computes the statistical correlation (Pearson’s r) between two user’s common ratings to determine their similarity.
* Spearman rank correlation
  + the items a user has rated are ranked such that their highest-rated item is at rank 1 and lowerrated items have higher ranks. Items with the same rating are assigned the average rank for their position
* Cosine similarity
  + it is a vector-space approach based on linear algebra rather than a statistical approach. Users are represented as N-dimensional vectors and similarity is measured by the cosine distance between two rating vectors



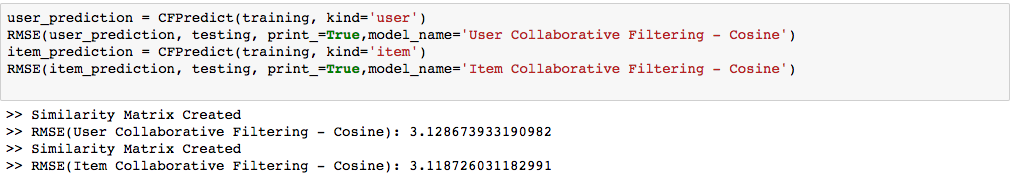
Figure

Figure 7. shows the implementation of similarity function that accept an input of matrix and return a matrix with the similarity score between users or similarity score between items. As pairwise distance matrix already accepts different metrics options which are the similarity function we discussed, I can get the similarity score.



Figure

Figure 8 shows the function of a simple collaborative filtering model. It first calculate and get a similarity matrix. it will then find the dot product of the training and get the similarity matrix. With our similarity matrix in hand, I can predict the ratings that were not included with the data. Using these predictions, I can then compare them with the test data to attempt to validate the quality of our recommender model. I also normalize it by the number of ratings.



Figure

Figure 9 shows the RMSE score for the both item-based and user-based collaborative filtering model. Comparing this with the mean rating model, CF model’s RMSE is lower which show the model gives a better prediction than mean rating model.

This model is get other users’ rating and multiply by the similarity score as the weight. As mentioned, it is also important to determine which users’ rating should we use for the prediction.

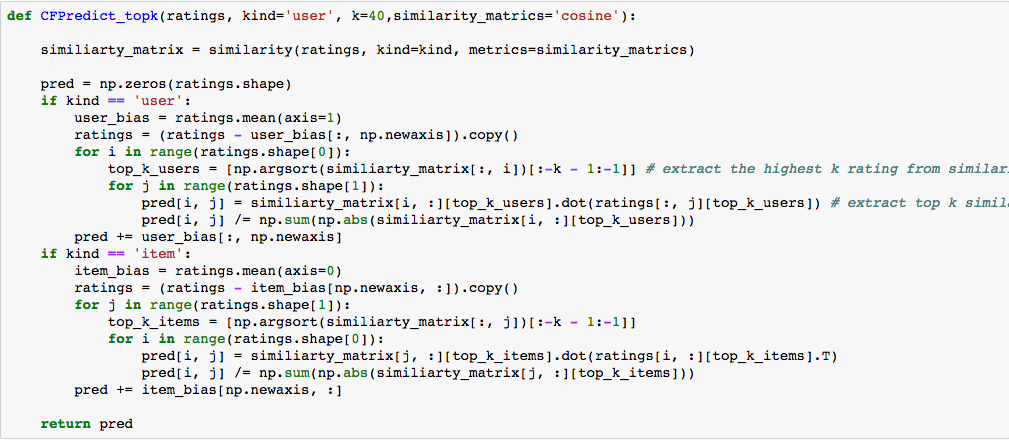
Incorporating gender, age, occupation and other features can be used to define ‘nearest neighbor’.

## Model 2 - Collaborative Filtering with K-nearest neighbor

Model 2 anticipates to improve model 1 – Collaborative Filtering – by including the kNN into the model. kNN is a classical classification algorithm that kNN that is a non-parametric lazy learning algorithm.  it does not make any assumptions on the underlying data distribution.

It does not use the training data points to do any generalization. Lack of generalization means that KNN keeps all the training data.  In other words, *there is*no explicit training phaseor it is very minimal. In this MovieLens data, kNN can be done in many ways as mentioned.

In this project I still rely on the existing similarity distance function and predict the rating only using the top-k similar users. I will talk about what can done beyond this beyond to improve it.



Figure

Figure 10 shows the source code and it is still using the item-based and user-base collaborative filtering. As view in the code that within the loop (*for i in range (ratings.shape[0]),* I first extract the index position of the highest k-number of users with highest similarity score. In the next loop (*for i in range (ratings.shape[1]),* it slices the dataset to only get the ratings from top k user and multiply that with similarity score. The prediction is also normalized and return as a prediction matrix.

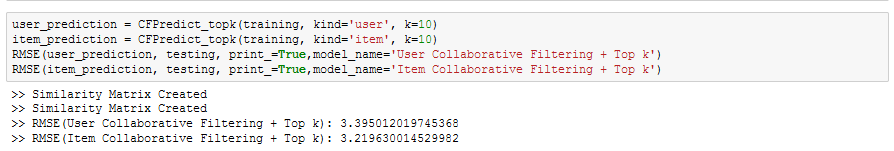


Figure 11

Figure 11 shows the RMSE score of the model and it is surprisingly higher than model 1 – normal collaborative filtering. This means using the current similarity function and find the top-k is not a good approach. We can incorporate other features as mentioned above such as age, genre, occupation and time to determine the similarity.

# Matrix Factorization

Matrix Factorization is a different approach to predicting ratings. It finds the set of features of items and users that determine the reaction of most users to most movies.

SVD has a popular approach of matrix factorization and it is an important property that makes it interesting for recommender systems. In the movie rating dataset, most users respond to a small number of features; they like certain genres, they may have certain famous actors or actresses that they like, and perhaps there are a few directors with a significant following. SVD tries to learn the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and items.

The general SVD equation can be expressed as follows: X = U x S x VT

Given an m x n matrix X:

* U is an m x r orthogonal matrix
* S is an r x r diagonal matrix with non-negative real numbers on the diagonal
* VT is an r x n orthogonal matrix

Matrix X can be factorized to U, S and V. The U matrix represents the feature vectors corresponding to the users in the hidden feature space and the V matrix represents the feature vectors corresponding to the items in the hidden feature space. SVD has a hyper-parameter k, which is the latent factor. It is potentially between to filter the small singular values can be introduced as removing noise data in the matrix.

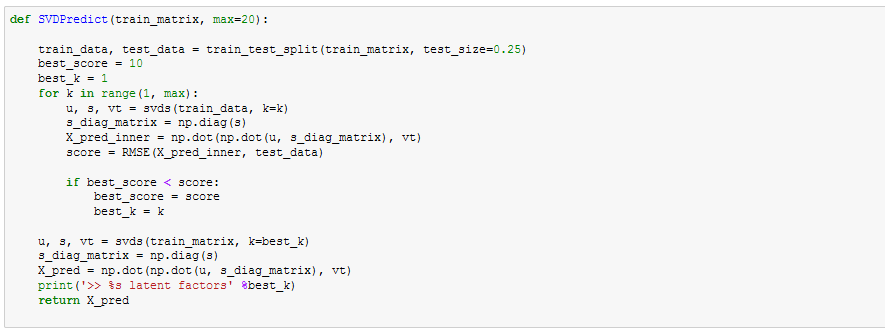
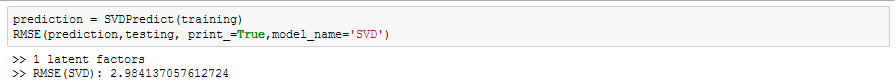


Figure 12

Figure shows show the implementation of SVDPredict function. I use the svds function from scipy to compute the basic SVD function. As there’s a hyper- parameter k. I have split the training data into training data and validation set and loop over different value of k. the prediction of the small training set is compared against the validation to calculate RMSE. The k value with lowest RMSE will be used for the real model. The key of SVD is it *X\_pred = np.dot(np.dot(u, s\_diag\_matrix), vt)* as it uses the 3 matrix from the SVD model with the k number of latent factor that we found as optimal. This line of code multiply the U, S & V matrix back to predict the rating.



The SVD has the best RMSE score out of the 3 models in this project. This means capturing the latent factor tends to give better prediction.

# Next Step

The performance of the 3 models vary differently and there are rooms to improve it. Collaborative Filtering has user-based model and item-based model which look at similarity between different entities. First, we can improve the similarity function which we can include additional features such as age, gender, movie genre, time, actor, IMBD information and etc. This becomes a clustering problem using different features. There are also content-based recommendation, demographic recommendation, utility-based recommendation and knowledge-based recommendation which some would need additional features and data in order to work.

Combining all these model will form a hybrid recommendation system which should be more accurate. The reasoning behind is we are using the similar users’ rating on similar movie and use the similarity to calculate the weighted average. An example would be in order to predict a new user’s prediction on “Alien”. We find users who are similar to the user (e.g age, frequency of movie watching, preference, previous movie ratings, etc) and their rating on movies similar to   
“Alien” (e.g. sic-fi, horror, year of production, movie length, etc). We can take the weight average of these ratings to predict.

In order to improve the SVD model, I can minimize the squared error by applying alternating least square or stochastic gradient descent and uses regularization terms to prevent overfitting in future. Using more advanced SVD models such as Asymmetric SVD. Another method is SVD-based recommendation. The new approach is to categorize items and users to SVD-based recommendation. We perform SVD on smaller matrices which consist of items in the same category and users in the same category. Furthermore, we can also use already included feature such as genre and data from IMBD into the model.

Another more advance and complicated method is to take time into account, some movie lover watch the movie on the first day and their rating will naturally be higher because of their enthusiasm. Some users just binge watch movies and tends to gives average rating to most movies during holiday.

Lastly, I have discussed many individual method and ensemble is always a way to predict the final rating by combing the predicted value from different models. Each model capture different component and features of the data and there’s no one perfect model. Ensemble model takes weight of different models’ predict and this weight can change depends on the context of the problems.