Movie Recommendation System

Introduction

"What movie should I watch tonight?" I have always asked myself this question. From Netflix to Hulu, the need to build a robust movie recommendation system is extremely important given the huge demand for personalized content of modern consumers. In this project, I am going to use the knowledge I have learned to build a movie recommendation system to recommend users movies that they might be interested in watching.

My client could be movie companies like Netflix, Hulu, or Disney+, where they have a strong need for a good movie recommendation system. These companies need to rely on a good movie recommendation system to attract users and retain users.

Dataset

The dataset that I am going to use is from the MovieLens. The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. This dataset is one of the most common datasets that are available on the internet for building a Recommender System. There are several size choices for synthetic datasets: 1B, 100K (100,000 ratings from 1000 users on 1700 movies. Released 4/1998.), 1M (1 million ratings from 6000 users on 4000 movies. Released 2/2003.), 10M (10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released 1/2009.), 20M (20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags. Released 4/2015; updated 10/2016)

In this project, I am going to use the 1M dataset.

Reference: https://grouplens.org/datasets/movielens/

Data Cleaning and Wrangling

There are 3 files (ratings, users, and movies) in the dataset. The three files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

In the users dataset, all demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided some demographic information are included in this data set.

The movies dataset includes movie title, movie id, and the corresponding movie genres.

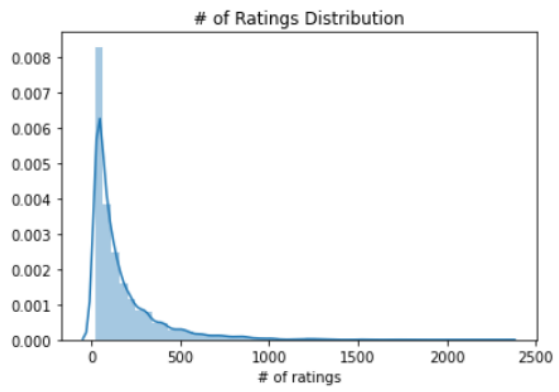
The rating datasets includes user id, movie id, ratings (made on a 5-star scale), and the rating timestamp. In this dataset, each user has at least 20 ratings.

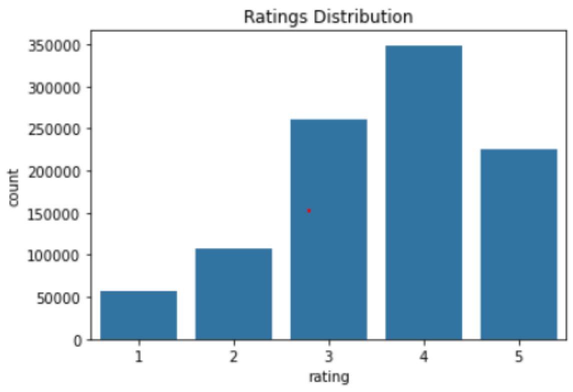
Since the dataset has already been cleaned by MovieLens, there are not duplicated rows or missing values. By checking over the dataset, there is nothing seems wrong. In the users dataset, Age and Occupation columns are already encoded. In order to have a more human readable exploratory data analysis later, I decoded these two columns. I create a column called age_desc to represent a human readable age range and a column called occ_desc to represent a human readable occupation.

EDA

In this EDA, we analyzed 3 sets of data (users, movies, and ratings). Through data analysis, I found some insightful information.

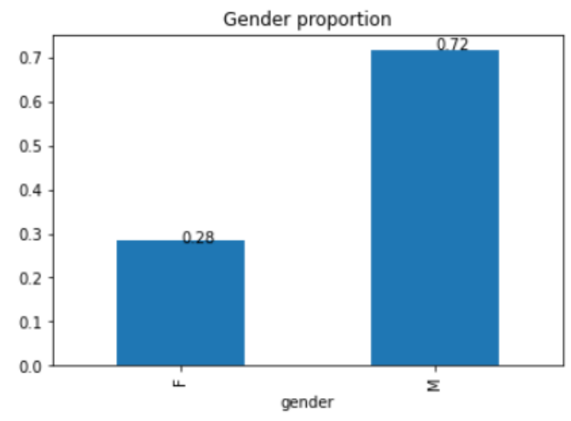
Distribution of ratings

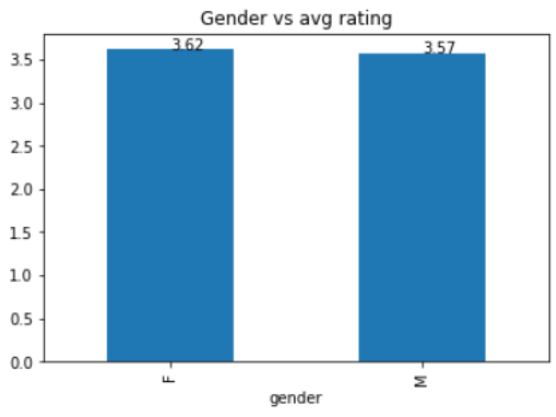




On average, a user rates 165 movies in MovieLens. Most users rate less than 500 movies. Most ratings are located on 3, 4, 5, meaning the users on MovieLens are pretty gentle to the ratings. However, the same ratings might mean differently for each user. For example, the rate 4 for user A could mean "highly recommended", but 4 for user B could be "not very impressive". We need to be careful about the different meanings for ratings for different users.

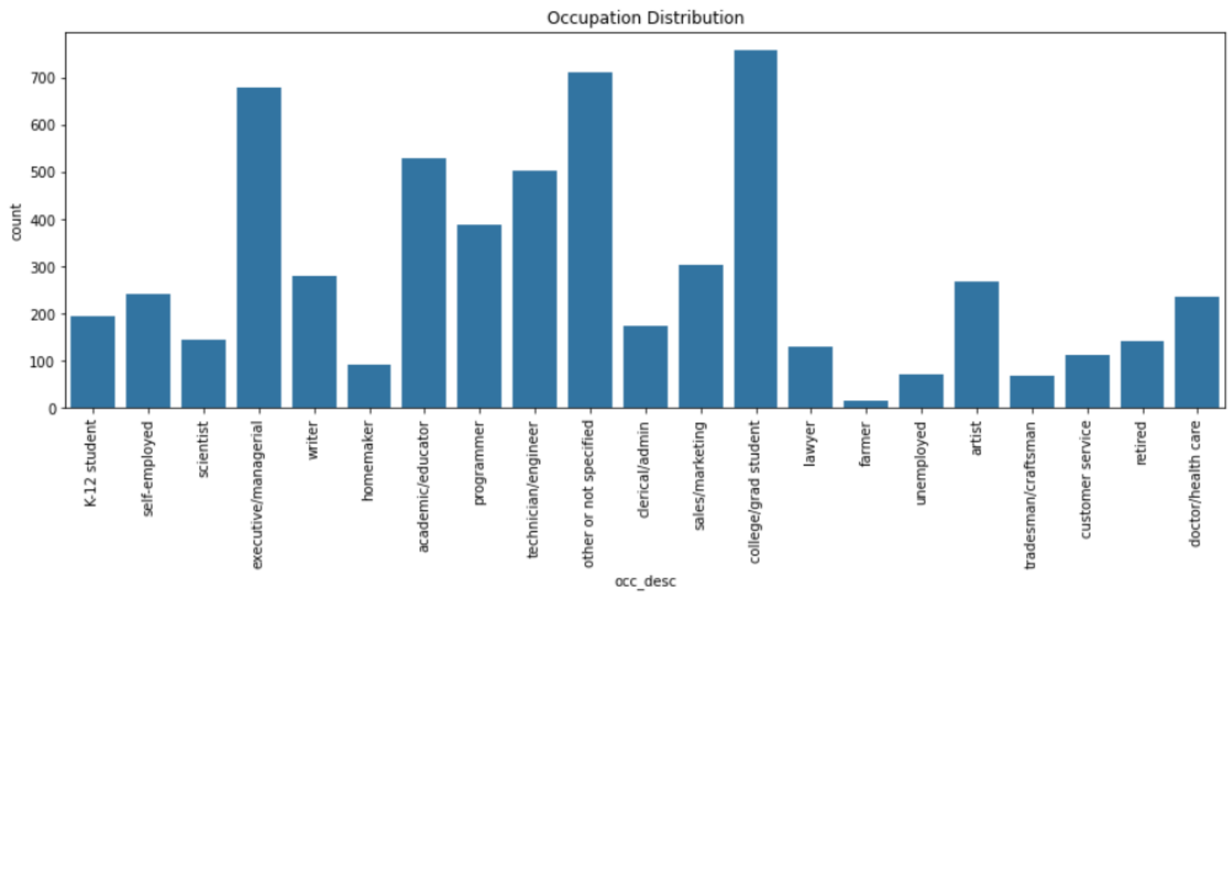
Gender

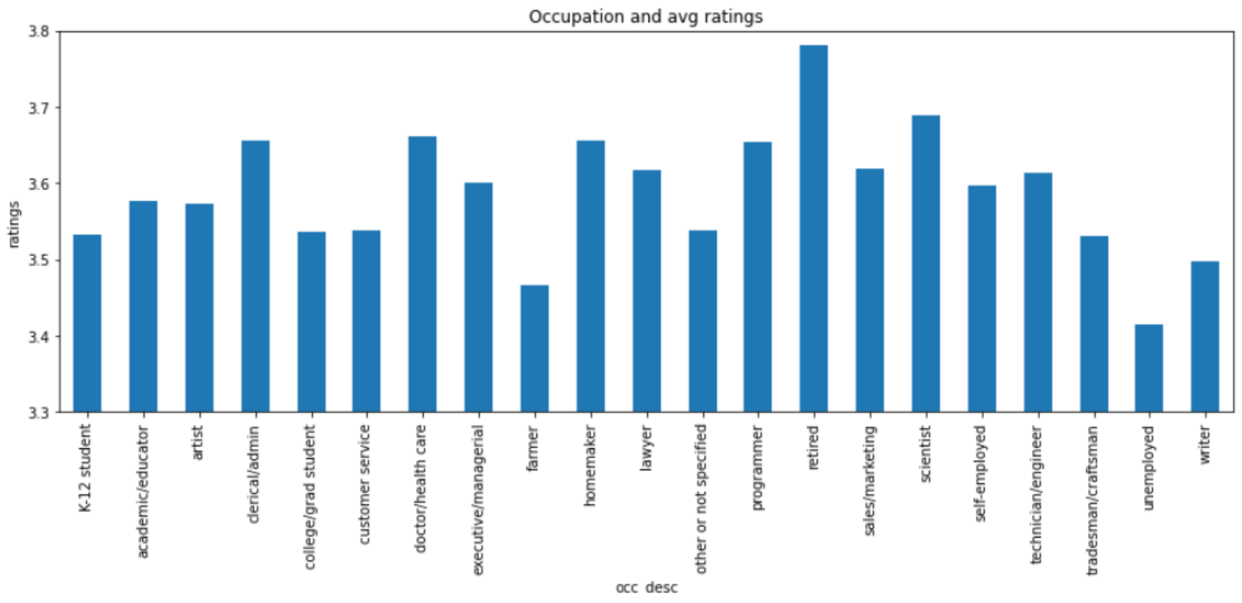




“Gender proportion” describes male and female users’ proportion. Among all users, 72% are male and 28% are female. There are significantly more male users than female users. The “Gender vs avg rating” shows the average ratings of male and female. On average, female users give higher ratings to movies than male users. Female users have an average rating of 3.62 which is slightly higher than male users’ 3.57.

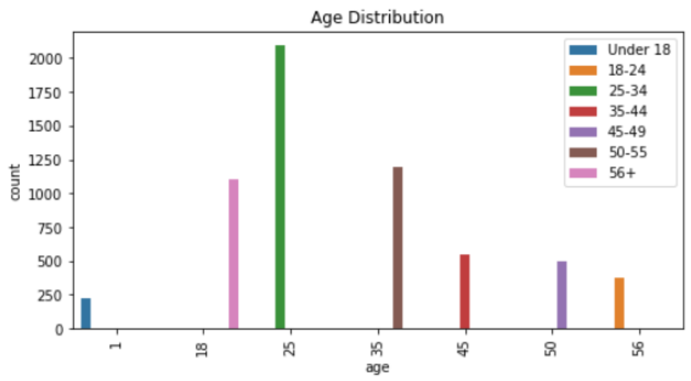
Occupation

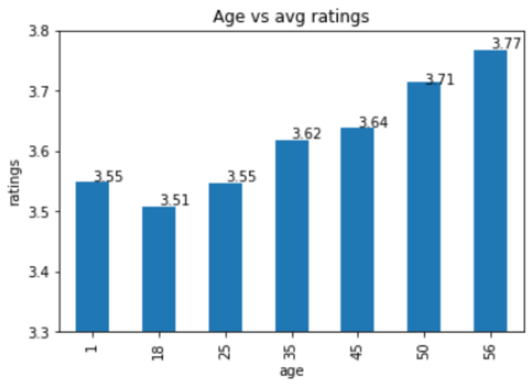




The “Occupation Distribution” shows the number of users that are certain occupations. Among all the users, college/grad student, executive/managerial and other are the biggest population. They provide the most ratings. The “Occupation and avg ratings” is the average ratings of different occupations. Retailed people gives the highest ratings on average. Unemployed and farmers tends to give the lowest ratings.

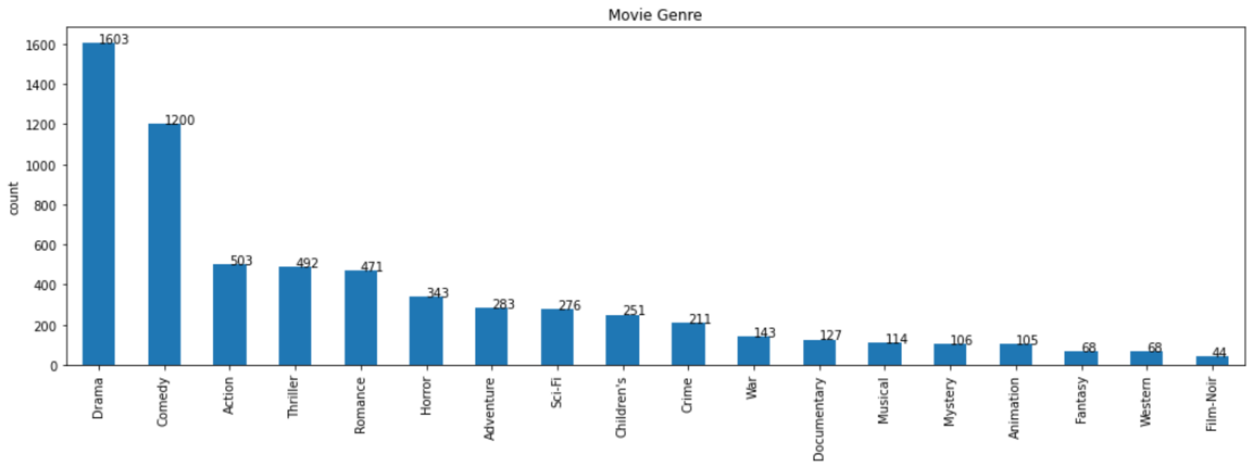
Age

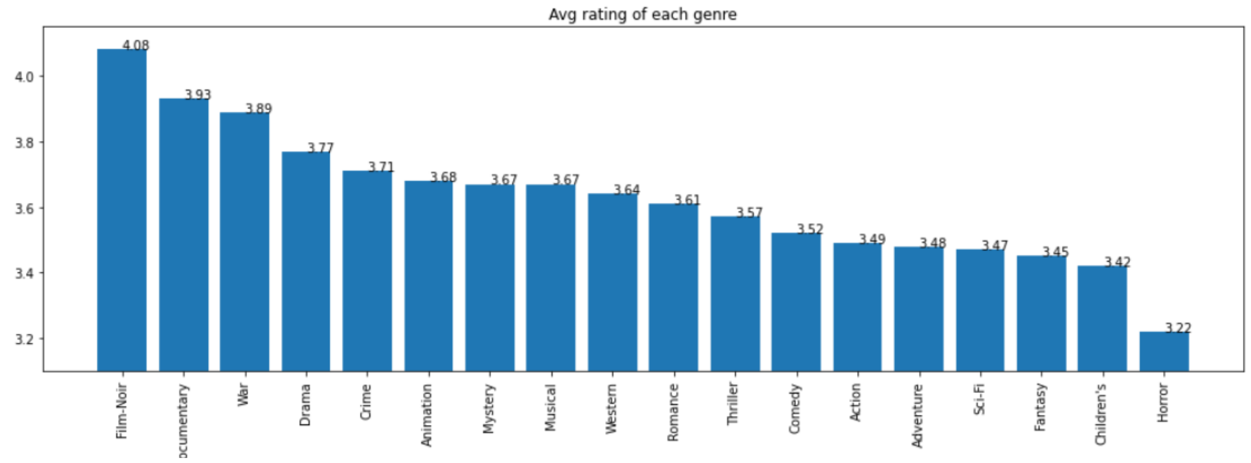




The “Age Distribution” shows the number of users in different age groups. Users between 25-34 are the largest population, 35-44 and 18-24 years old users are the next largest population. However, I notice that the age groups are not equally splatted. Maybe the website wants to keep the number of users in each group to be identical. The “Age vs avg ratings” graph show the average ratings of different age groups. Users between 18-24 tends to give the lowest ratings. Older people tend to give higher ratings.

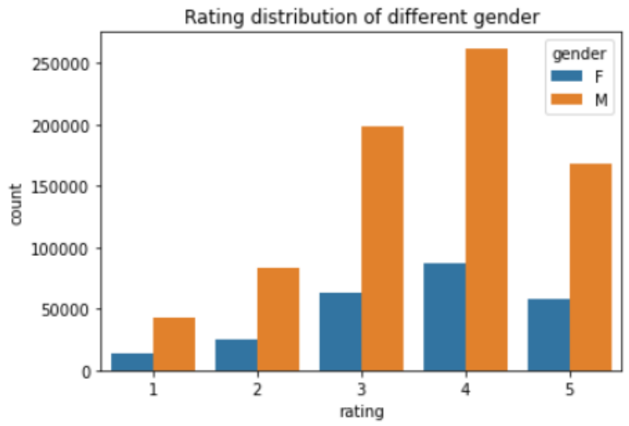
Movie Genre



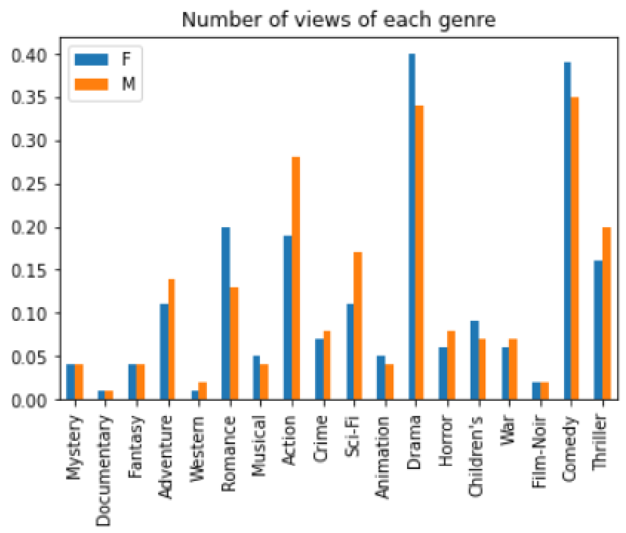


"Movie Genre" shows the number of movies in each genre. There are total of 18 different genres. The top 3 genres with the most movies are Drama, Comedy and Action. The Film-Noir has the least movies. However, the Film-Noir has the highest average ratings (4.08). Horror has the lowest average ratings (3.22).

Movie Genre and Gender



Distribution of ratings for male and female users has similar shapes. Most ratings are 3, 4 and 5.



From the "Number of views of each genre", we find"
The most popular genres for male are:

- Drama
- Comedy
- Action

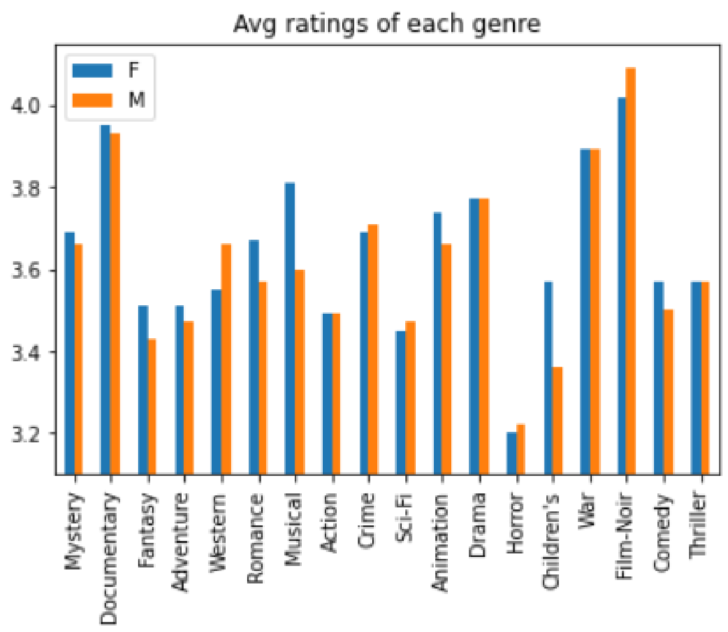
The most popular genres for female are:

- Drama
- Comedy
- Romance

Significantly more male users watched action movies, and there are more male users watched Crime, War, Adventure, Sci-Fi, Thriller and Horror.

Significantly more female users watched romance movies, and there are more female users watched musical, drama, comedy and children's.

Documentary has the least watches for both genders.



For most genres, female rates higher than male. However, male rate higher than female in Film-Noir, Western, Crime, Horror and Sci-Fi. I also noticed that female favors Children's movie much more than male. Maybe because female spends more time with their children. Female also rate much higher than male for Musical movies.

Genre vs Age

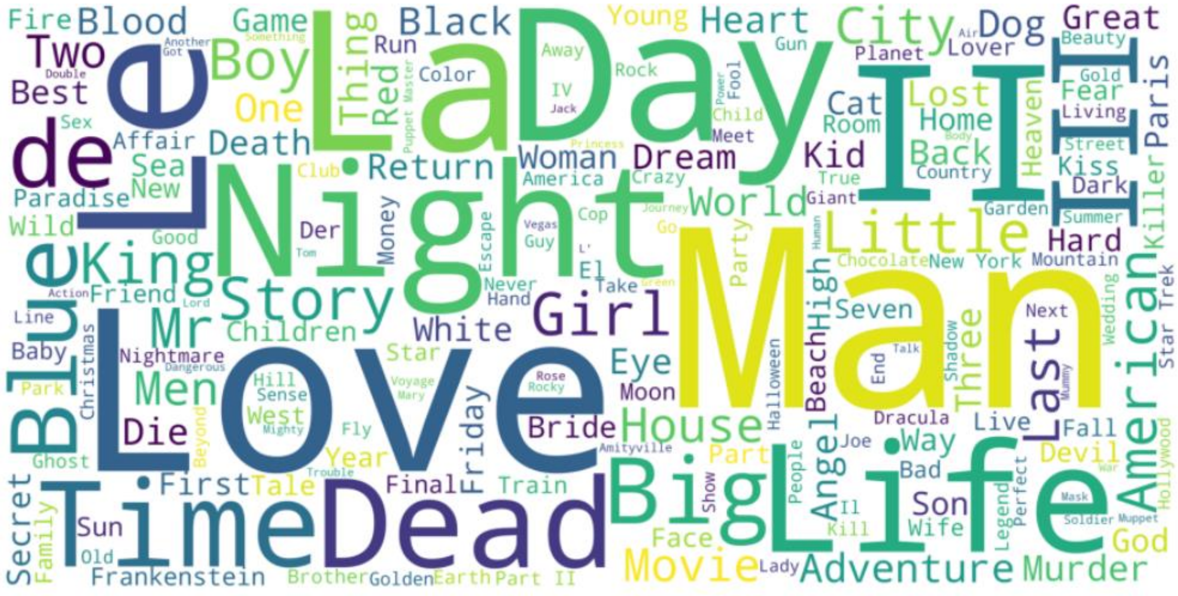
For most age groups, the popular genres are the identical. The most popular genre is Comedy and Drama and the least popular genre is Documentary. Action, Thrill, Sci-Fi, Adventure and Romance are also popular. Younger users watches more Children's movies and Animation. As age grows, users' tastes are more concentrated on Drama and Comedy.

And, for most age groups, the most high-rated movie is Film-Noir and the lowest rated genre is Horror. One thing that we need to notice is that users under 18 give lowest ratings to Children's movies.

Genre vs Occupation

Most occupation give highest ratings to Film-Noir and lowest for Horror. Doctor/health care and homeworker users favor War type movies. Farmer loves documentary. K-12 student give low ratings to Children's movies. Unemployed does not like Children's movies as well.

Movie Titles



I also built a word cloud on the titles of movies, since the title is also a very important factor for my to decide whether to watch the movie. I would like to check if it may have effects to the ratings. The most frequent words are ""Love", "Man", "Night", "Life", "Day", "Time", "Dead". Movies with "Love", "Life" and "Day" has higher ratings.

EDA Summary

Through this Exploratory Data Analysis, we find that people in different age group, with different occupation could affect their preference in genre. We could build several content-based or collaborative filter model to make recommendations to different types of users based on their user portrait.

Models

Since there are not many features, and most of feature are 0 and 1, we do not need to do many feature engineering processes.

In the modeling section, I tried 6 recommendation engines. 4 of them are relatively useful, and performed well, so I will introduce these 4 recommendation engines: Weak personalized Recommendation, Content based filtering, Memory-based collaborative filtering, Model-based collaborative filtering.

Weak Personalized Recommendation

In this model, I will recommend movies to users based on their demographic (gender, occupation, and age). I will find the most popular movie genres in within each user group and recommend the most popular movies in the genre.

I first merged all three data frames (users, movies, and ratings) together, and group the users by their gender, age, and occupation. In each group, I found out the most popular movie genre (based on the number of ratings, the movie genre is considered as most popular if it has the most ratings). Then based on the most popular genre, I build an engine to recommend the user the most popular movies in the genre.

The most popular movies are not based the average rating of the movie, because the movie could only have 5 ratings but all 5 are 5 stars. This kind of movie will have high ratings but could not considered as popular. So I calculate the score based on weighted rating metric to decide the most popular movies in each genre.

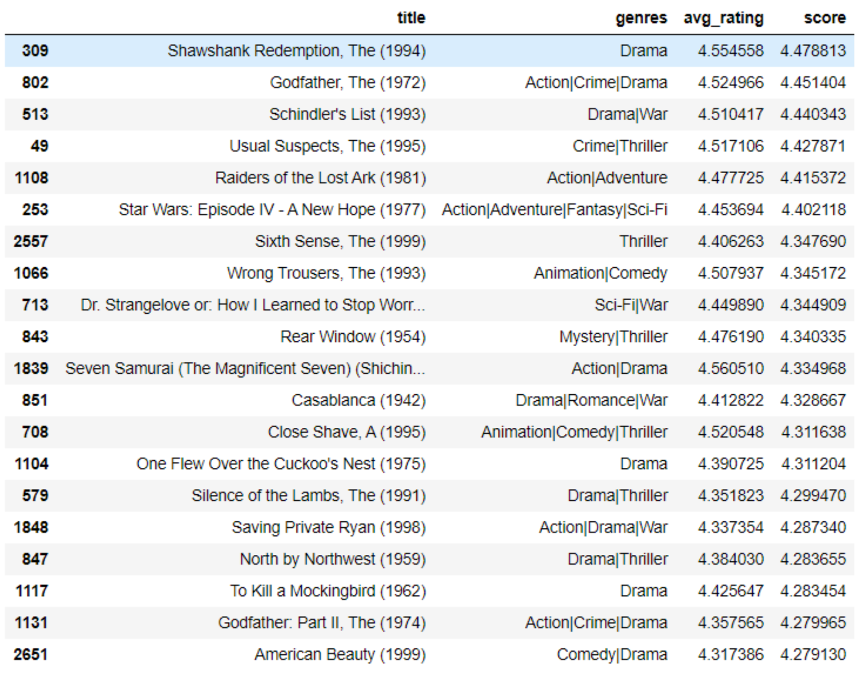
The weighted rating metric is,

$WR = ((v / (v + m)) R) + ((m / (v + m)) C)$

where,

- v is the number of votes for the movie
- m is the minimum votes required to be listed in the chart
- R is the average rating of the movie
- C is the mean vote across the whole report

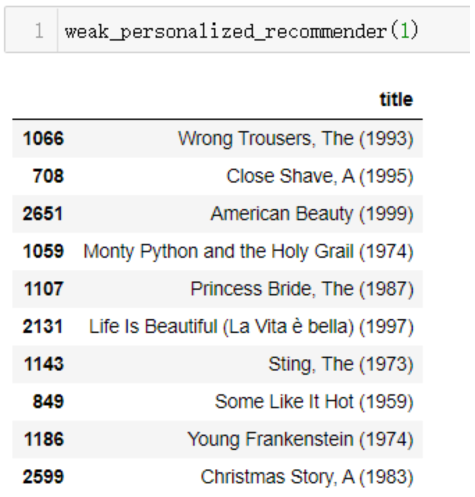
Based on this metric, we can find out the most popular movies on the website (Below is the top 20 most popular movies on MovieLens):



	title	genres	avg_rating	score
309	Shawshank Redemption, The (1994)	Drama	4.554558	4.478813
802	Godfather, The (1972)	Action|Crime|Drama	4.524966	4.451404
513	Schindler's List (1993)	Drama|War	4.510417	4.440343
49	Usual Suspects, The (1995)	Crime|Thriller	4.517106	4.427871
1108	Raiders of the Lost Ark (1981)	Action|Adventure	4.477725	4.415372
253	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	4.453694	4.402118
2557	Sixth Sense, The (1999)	Thriller	4.406263	4.347690
1066	Wrong Trousers, The (1993)	Animation|Comedy	4.507937	4.345172
713	Dr. Strangelove or: How I Learned to Stop Worr...	Sci-Fi|War	4.449890	4.344909
843	Rear Window (1954)	Mystery|Thriller	4.476190	4.340335
1839	Seven Samurai (The Magnificent Seven) (Shichin...	Action|Drama	4.560510	4.334968
851	Casablanca (1942)	Drama|Romance|War	4.412822	4.328667
708	Close Shave, A (1995)	Animation|Comedy|Thriller	4.520548	4.311638
1104	One Flew Over the Cuckoo's Nest (1975)	Drama	4.390725	4.311204
579	Silence of the Lambs, The (1991)	Drama|Thriller	4.351823	4.299470
1848	Saving Private Ryan (1998)	Action|Drama|War	4.337354	4.287340
847	North by Northwest (1959)	Drama|Thriller	4.384030	4.283655
1117	To Kill a Mockingbird (1962)	Drama	4.425647	4.283454
1131	Godfather: Part II, The (1974)	Action|Crime|Drama	4.357565	4.279965
2651	American Beauty (1999)	Comedy|Drama	4.317386	4.279130

Then, I built the weak personalized recommendation engine to recommend the 10 most popular movies to each user.

Here is an example. we can type in the user id (user id = 1), and based on the user's gender, age and occupation to find the most popular genre. And based on the most popular genre to find the most popular movies, then recommend the 10 most popular movies to the user.



```
weak_personalized_recommender(1)
```

	title
1066	Wrong Trousers, The (1993)
708	Close Shave, A (1995)
2651	American Beauty (1999)
1059	Monty Python and the Holy Grail (1974)
1107	Princess Bride, The (1987)
2131	Life Is Beautiful (La Vita è bella) (1997)
1143	Sting, The (1973)
849	Some Like It Hot (1959)
1186	Young Frankenstein (1974)
2599	Christmas Story, A (1983)

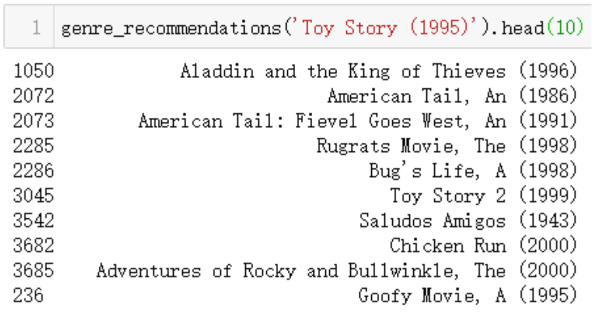
This engine could be used to recommend movies to new users. When a new user just registers the website, we do not know the user's preference. We can user this engine to safely recommend some most popular movies based on the user's demographic.

Content Based Recommendation System Based on Movie Genres

When I personally search for movies that I might be interested in, genre is an important factor. I would tend to watch movies in genres that I like. So, I am going to build a Content-Based Recommendation Engine that computes similarity between movies based on movie genres. It will suggest movies that are most similar to a particular movie based on its genre. To do so, I will make use of the file movies.csv.

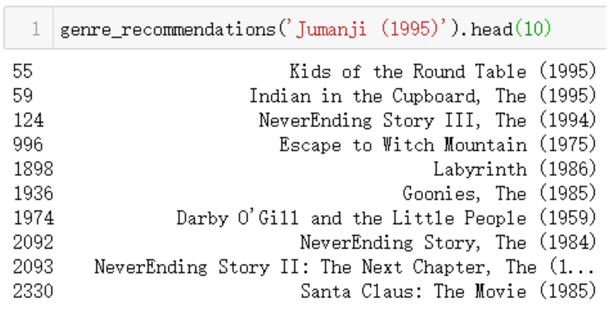
I first used the Cosine Similarity to calculate a numeric quantity that denotes the similarity between two movies. Since we have used the TF-IDF Vectorizer, calculating the Dot Product will directly give us the Cosine Similarity Score. Therefore, we will use sklearn's linear_kernel instead of cosine_similarities since it is much faster.

To check the performance of this content-based filtering, I tried 3 movies "Toy Story", "Jumanji", and "Gone in 60 Seconds".



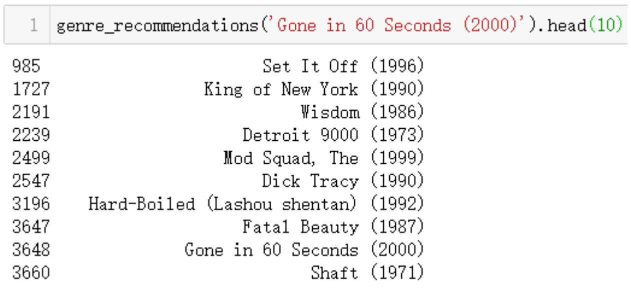
```
genre_recommendations('Toy Story (1995)').head(10)
```

```
1050            Aladdin and the King of Thieves (1996)
2072                        American Tail, An (1986)
2073          American Tail: Fievel Goes West, An (1991)
2285                        Rugrats Movie, The (1998)
2286                             Bug's Life, A (1998)
3045                              Toy Story 2 (1999)
3542                           Saludos Amigos (1943)
3682                             Chicken Run (2000)
3685    Adventures of Rocky and Bullwinkle, The (2000)
236                             Goofy Movie, A (1995)
```



```
genre_recommendations('Jumanji (1995)').head(10)
```

```
55                         Kids of the Round Table (1995)
59                      Indian in the Cupboard, The (1995)
124                      NeverEnding Story III, The (1994)
996                        Escape to Witch Mountain (1975)
1898                                     Labyrinth (1986)
1936                                  Goonies, The (1985)
1974            Darby O'Gill and the Little People (1959)
2092                         NeverEnding Story, The (1984)
2093   NeverEnding Story II: The Next Chapter, The (1...
2330                         Santa Claus: The Movie (1985)
```



```
genre_recommendations('Gone in 60 Seconds (2000)').head(10)
```

```
985                       Set It Off (1996)
1727                 King of New York (1990)
2191                           Wisdom (1986)
2239                     Detroit 9000 (1973)
2499                    Mod Squad, The (1999)
2547                       Dick Tracy (1990)
3196    Hard-Boiled (Lashou shentan) (1992)
3647                     Fatal Beauty (1987)
3648               Gone in 60 Seconds (2000)
3660                            Shaft (1971)
```

The genre recommendation engine gives me movies in similar genres. By checking the results, the recommended movies are acceptable.

This recommendation system does not need data on other users. We can recommend to users based on their unique tastes. However, this recommendation engine will not recommend movies outside a user's content profile.

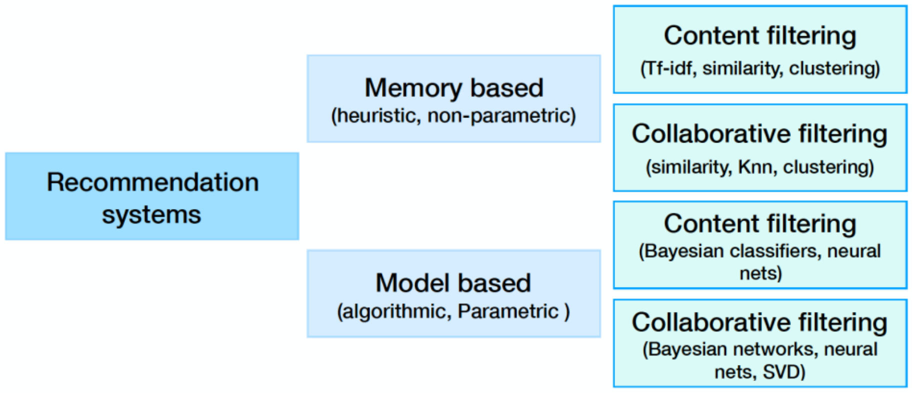
So, this recommendation engine could be used under the following circumstances:

1. A new user registers the movie website. The website showed a list of movies and let the user choose the movies they like. Then we use this recommendation system to recommend similar movies based on what the user chose.
2. A user rates high rating for a certain movie, then the website can use this recommendation engine to recommend similar movies.

Collaborative Filtering

There are a couple different types of collaborative filtering:

- Memory based
  - User-Item Filtering
  - Item-Item Filtering
- Model based
  - Matrix Factorization
  - Clustering
  - Deep Learning



Recommendation systems
Memory based
(heuristic, non-parametric)
Model based
(algorithmic, Parametric)
Content filtering
(Tf-idf, similarity, clustering)
Collaborative filtering
(similarity, Knn, clustering)
Content filtering
(Bayesian classifiers, neural nets)
Collaborative filtering
(Bayesian networks, neural nets, SVD)

Memory-Based Collaborative Filtering

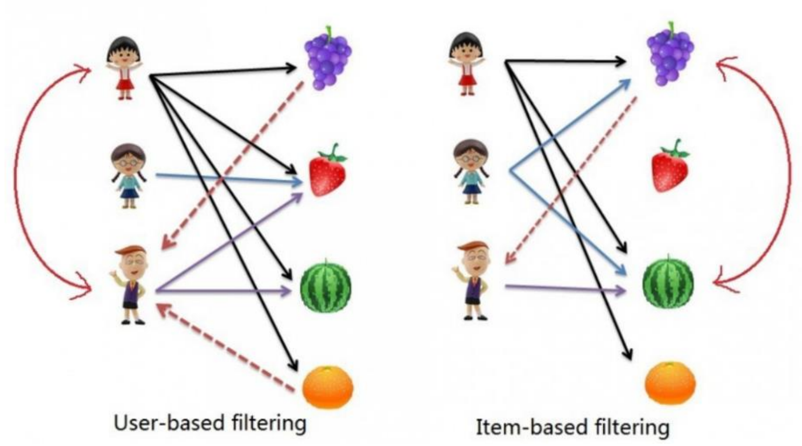
User-Item Collaborative Filtering considers similarities between user consumption history. The underlying concept behind this technique is as follows:

Assume Person A likes Oranges, and Person B likes Oranges. Assume Person A likes Apples. Person B is likely to have similar opinions on Apples as A than some other random person.

Item-Item Collaborative Filtering takes the similarities between items' consumption history. Earlier collaborative filtering systems based on rating similarity between users (known as user based collaborative filtering) had several problems:

- systems performed poorly when they had many items but comparatively few ratings
- computing similarities between all pairs of users was expensive
- user profiles changed quickly, and the entire system model had to be recomputed

Item-item models resolve these problems in systems that have more users than items. Item-item models use rating distributions per item, not per user. With more users than items, each item tends to have more ratings than each user, so an item's average rating usually does not change quickly. This leads to more stable rating distributions in the model, so the model does not have to be rebuilt as often. When users consume and then rate an item, that item's similar items are picked from the existing system model and added to the user's recommendations.



User-Item Collaborative Filtering

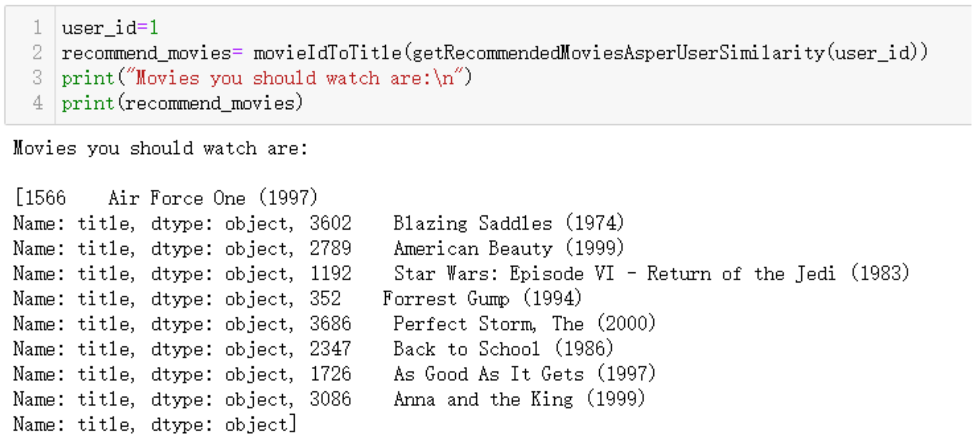
The underlying assumption of the user-item collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

Here I will find look alike users based on similarity and recommend movies which first user's look-alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every user pair information.

I first create a matrix of similarities of each user to every other users. Since we have 6040 users, the matrix is 6040x6040. In the matrix, we have similarity of users in columns with respective users in row. So, if we find maximum value in a column, we will get the user with highest similarity. So now we can have a pair of users which are similar.

Then we built a recommendation engine to recommend movies which user has not watched as per User Similarity. We will find a user which is like our target user, and then filter the movies which are highly rated by the similar user to recommend them to the target user.

Here is the result. We input the target user id (user id =1), then recommend movies that his/her similar user likes the most.



```
user_id=1
recommend_movies= movieIdToTitle(getRecommendedMoviesAsperUserSimilarity(user_id))
print("Movies you should watch are:\n")
print(recommend_movies)
```

```
Movies you should watch are:

[1566    Air Force One (1997)
Name: title, dtype: object, 3602    Blazing Saddles (1974)
Name: title, dtype: object, 2789    American Beauty (1999)
Name: title, dtype: object, 1192    Star Wars: Episode VI - Return of the Jedi (1983)
Name: title, dtype: object, 352    Forrest Gump (1994)
Name: title, dtype: object, 3686    Perfect Storm, The (2000)
Name: title, dtype: object, 2347    Back to School (1986)
Name: title, dtype: object, 1726    As Good As It Gets (1997)
Name: title, dtype: object, 3086    Anna and the King (1999)
Name: title, dtype: object]
```

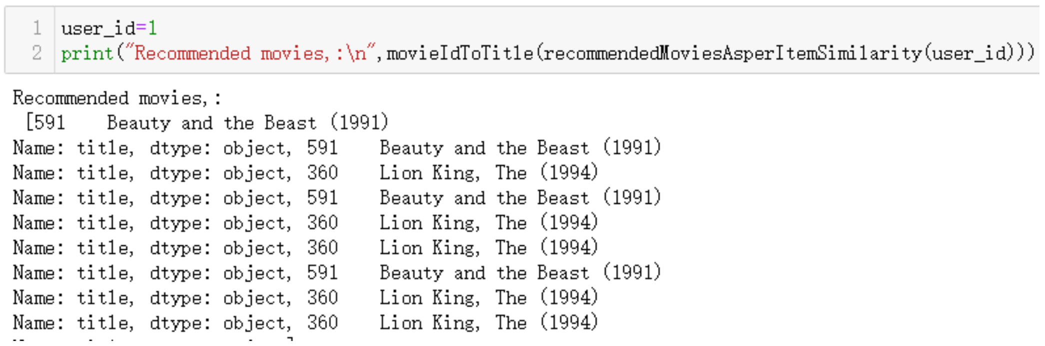
Item-Item Collaborative Filtering

Item-Item collaborative filtering look for items that are similar to the movies that user has already rated and recommend most similar movies. But what does that mean when we say item-item similarity? In this case we do not mean whether two items are the same by attribute like Fountain pen and pilot pen are similar because both are pen. Instead, what similarity means is how people treat two items the same in terms of like and dislike. we try finding movie's look-alike. Once we have movie's look-alike matrix, we can easily recommend alike movies to user who have rated any movie from the dataset. This algorithm is far less resource consuming than user-user collaborative filtering. Hence, for a new user, the algorithm takes far lesser time than user-user collaborate as we don't need all similarity scores between users. And with fixed number of movies, movie-movie look alike matrix is fixed over time.

In this engine, I will use movies.csv and ratings.csv since. First, I created a user-movie matrix. We have each user to be a column and each movie to be a row. Ratings from users to each movie is filled into the matrix. Based on the ratings from users, I used pairwise-distances to calculate the movie similarity matrix.

Then I built an engine which takes a user id. The engine will find the most recent rated movies that are rated above 4 by the user. We are finding this because we know that in Item-Item similarity approach we recommended movies to the user based on the previous selection. Then, find movie similarities of the movie that the user is highly rated, and get the movies which are highly similar to movie that is highly rated by the user. From the movie list, I filtered the movies with the similarity greater than 0.45. Then, filter out the movies that the user has not seen and then recommended that movies to the user.

Here is the result. We input the target user id (user id =1), then recommend movies that similar to the one he/she recently liked.



```
user_id=1
print("Recommended movies,:\n",movieIdToTitle(recommendedMoviesAsperItemSimilarity(user_id)))
```

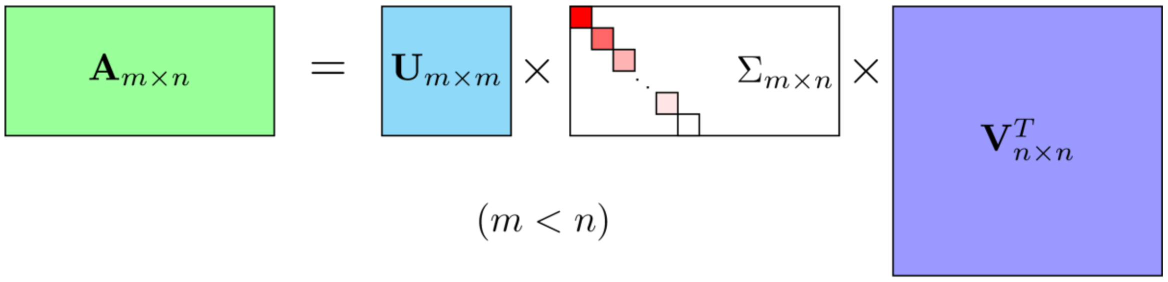
```
Recommended movies,:
 [591    Beauty and the Beast (1991)
Name: title, dtype: object, 591    Beauty and the Beast (1991)
Name: title, dtype: object, 360    Lion King, The (1994)
Name: title, dtype: object, 591    Beauty and the Beast (1991)
Name: title, dtype: object, 360    Lion King, The (1994)
Name: title, dtype: object, 360    Lion King, The (1994)
Name: title, dtype: object, 591    Beauty and the Beast (1991)
Name: title, dtype: object, 360    Lion King, The (1994)
Name: title, dtype: object, 360    Lion King, The (1994)
```

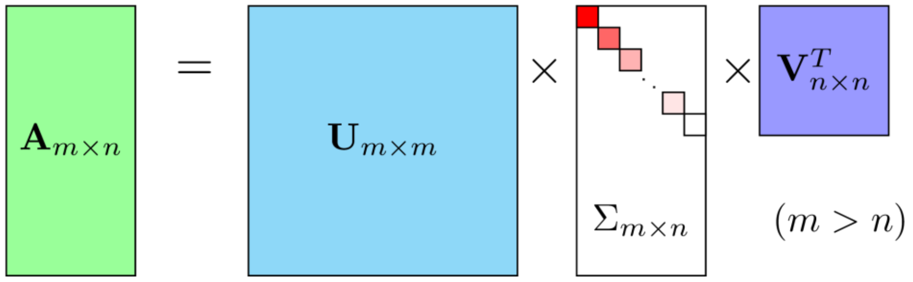
Model-Based Collaborative Filtering – SVD

In this model, I used model based collaborative filtering to build the recommendation engine. Model-Based Collaborative Filtering algorithm uses RS information to create a model that generates the recommendations. Unlike Memory-Based Collaborative Filtering, Model-based Collaborative Filtering does not use the whole dataset to compute predictions for real data. There are various model-based CF algorithms including Bayesian Networks, Clustering Models, and Latent Semantic Models such as Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and Probabilistic Matrix Factorization for dimensionality reduction of rating matrix. The goal of this approach is to uncover latent factors that explains observed ratings. In this attempt, I used SVD to build the model.

Model-based Collaborative Filtering is based on matrix factorization (MF) which has received greater exposure, mainly as an unsupervised learning method for latent variable decomposition and dimensionality reduction. Matrix factorization is widely used for recommender systems where it can deal better with scalability and sparsity than Memory-based CF:

- The goal of MF is to learn the 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.
- When you have a very sparse matrix, with a lot of dimensions, by doing matrix factorization, you can restructure the user-item matrix into low-rank structure, and you can represent the matrix by the multiplication of two low-rank matrices, where the rows contain the latent vector.
- You fit this matrix to approximate your original matrix, as closely as possible, by multiplying the low-rank matrices together, which fills in the entries missing in the original matrix.





The Singular Value Decomposition (SVD) is the powerful technique of dimensionality reduction. The key issue in an SVD decomposition is to find a lower dimensional feature space. SVD of an m × n matrix A is of the form shown above.

Where,

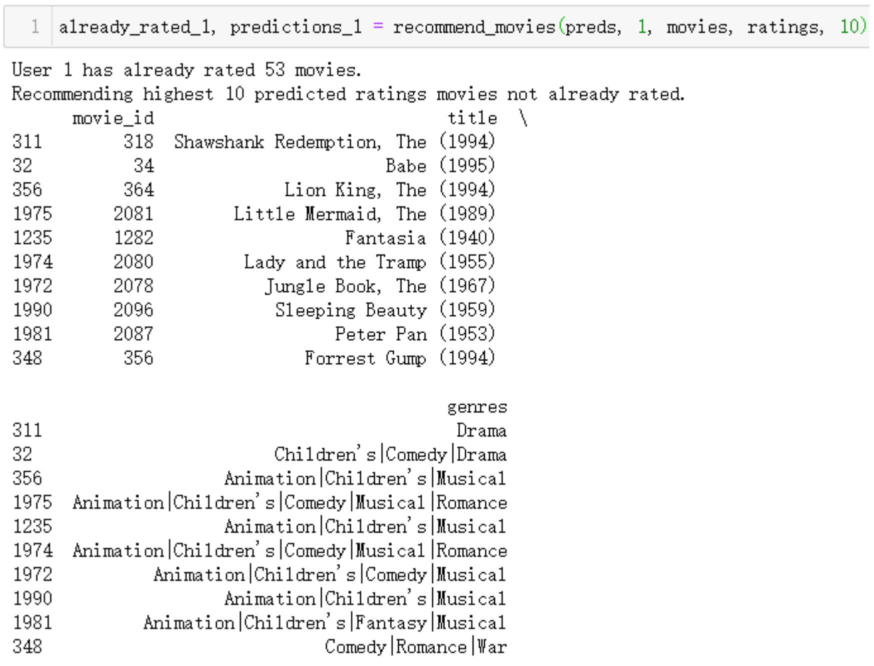
- U and V are m × m and n × n orthogonal matrices respectively
- Σ is the m × n singular orthogonal matrix with non-negative elements

An m × m matrix U is called orthogonal if equals to an m × m identity matrix. The diagonal elements in Σ (σ1, σ2, σ3, …… σn) are called the singular values of matrix A. Usually, the singular values are placed in the descending order in Σ. The column vectors of U and V are called the left singular vectors and the right singular vectors respectively。

In this model, I used svds function in Scipy because it let's me choose how many latent factors I want to use to approximate the original ratings matrix (instead of having to truncate it after).

After setting up the function and prepare the vectors, I can do movie prediction all at once by following the math and matrix multiply U , Σ , and VT back to get the rank k=50 approximation of A. With the predictions matrix for every user, I, then, built a function to return the movies with the highest predicted rating that the specified user hasn't already rated. The function takes 5 inputs: the predictions matrix, user id, movies (the data frame from movies.csv), original_ratings (the data frame from ratings.csv), and number of recommendations.

Here is a sample result, preds is the predictions matrix, user is is 1, movies the movies data frame, ratings is the ratings data frame, and 10 means returning 10 recommendations.



```
already_rated_1, predictions_1 = recommend_movies(preds, 1, movies, ratings, 10)
```

```
User 1 has already rated 53 movies.
Recommending highest 10 predicted ratings movies not already rated.
      movie_id                          title  \
311        318  Shawshank Redemption, The (1994)
32          34                     Babe (1995)
356        364            Lion King, The (1994)
1975      2081        Little Mermaid, The (1989)
1235      1282                 Fantasia (1940)
1974      2080         Lady and the Tramp (1955)
1972      2078          Jungle Book, The (1967)
1990      2096           Sleeping Beauty (1959)
1981      2087                 Peter Pan (1953)
348        356              Forrest Gump (1994)

                                          genres
311                                        Drama
32                         Children's|Comedy|Drama
356                    Animation|Children's|Musical
1975  Animation|Children's|Comedy|Musical|Romance
1235                   Animation|Children's|Musical
1974  Animation|Children's|Comedy|Musical|Romance
1972            Animation|Children's|Comedy|Musical
1990                   Animation|Children's|Musical
1981           Animation|Children's|Fantasy|Musical
348                             Comedy|Romance|War
```

By conducting the model evaluation, the RMSE is 0.866 meaning the performance of this model is pretty good.

Conclusion

In this project, I have built 4 different recommendation engines based on different ideas and algorithms. They are as follows:

1. **Weak Personalized Recommendation:**

This engine recommends movies to users based on their demographic (gender, occupation, and age). The engine finds the most popular movie genres in within each user group and recommend the most popular movies in the genre. This engine could be used when a new user just register, and we safely recommend some most popular movies based on the user’s demographic.

2. **Content Based Recommendation System Based on Movie Genres:**
   This content-based recommendation engine computes similarity between movies based on movie genres. It will recommend movies that are most similar to a particular movie based on its genre. This engine could be used when a new user chooses the movie they like and this engine could recommend similar movies based on their preference. The engine can be also used when a user rates highly for a certain movie, then our engine could recommend similar movies.

3. **Memory-Based Collaborative Filtering:**
   I built two types of memory-based collaborative filtering: User-item collaborative filtering and Item-item filtering. User-Item Collaborative Filtering considers similarities between user consumption history. The engine finds the most similar users B of the target user A, then recommend the top-rated movies by user B to user A.
   Item-Item Collaborative Filtering takes the similarities between items' consumption history. The engine find the most recent highly-rated movie (A) of the user, then find a list of most similar movies to movie A, then recommend the list of similar movies to the user.

4. **Model-Based Collaborative Filtering – SVD:**
   In this model-based collaborative filtering attempt, I user SVD to calculate the predictions matrix for every user. Then I built a function to return the movies with the highest predicted rating that the specified user hasn't already rated. The model is evaluated with a RMSE of 0.866, which mean the performance of this model is pretty acceptable.