User Based Collaborative Filtering Recommender System

Definition

Project Overview

Recommender systems have become a well studied problem now, and heavily integrated in many websites and apps we use daily,these systems tend to help users in their everyday decision making process from buying items to music listening by recommendations tailored differently to each user taste.
They are implemented in famous e-commerce websites like Amazon's famous "customers who bought this item also bought" section, in music we find Spotify's "Discover now" playlist which recommends a playlist curated differently to each user based on his taste, and even in movies where companies like Netflix recommend new movies you may like to watch,recommender systems have proven to be very important for these companies as they are an essential part for increasing their revenues and retaining customers.
There are different types for Recommender systems,this project is about implementing User Based Collaborative filtering recommender systems for movies, this type of systems is based on a simple idea which is people tend to ask their similar in taste friends for movies to watch, same idea implemented here but on a large scale of users, we find similar users in taste, we see what other items they like then recommend these items.

Problem Statement

Given a dataset (MovieLens dataset)of movie ratings and the ID of each user our goal is to:

1. Identify similar users(nearest neighbours) that had similar preferences to the user we want to recommend movies to.
2. For every movie the user has not yet seen, a prediction is computed based on the ratings for these nearest neighbour users

There are two assumptions here

1. If users had similar tastes in the past they will have similar tastes in the future
2. User preferences remain stable and consistent over time.

Metrics

There are two approaches for metrics in this problem

1. Since this is a mainly a classification problem (Rating from 1 to 5), Accuracy was used to score the results of the classifier, but after observing the results and score, the score was too low, but the results were pretty close to true ones (High ratings are predicted high, low ratings are predicted low).
2. Due to the problem with the first metric and after some research, i used Root mean squared error, it produced good results because as i stated earlier the ratings were pretty close to true ones (if true ratings were high predictions were high and vice versa)

Analysis

Data Exploration & Exploratory Visualization

Here the dataset used is MovieLens dataset.

	user_id	age	gender	occupation	zipcode	movie_id	title	release_date	video_release_date	IMDb URL	...	Horror	Musica
0	1	24	M	technician	85711	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	...	0	0
1	1	24	M	technician	85711	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	...	0	0
2	1	24	M	technician	85711	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	...	0	0
3	1	24	M	technician	85711	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	...	0	0
4	1	24	M	technician	85711	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	...	0	0

The above figure is a sample from the dataset, after reading three files(user,movie,ratings) and merging them, there are some features about user like ID,age and occupation, other features about the movie like ID, title and there is a categorical feature (genre) which is distributed across different columns where each column name is a category, there are some other irrelevant features for movies and finally there are the ratings.

First we start with some statistics about the data to understand it better:
Total number of users 943
Total number of features 30 (genre is distributed across columns,where each one is a dummy feature)
Total number of features 12 (if genres were combined into one column)
Number of Male users 670
Number of female users 273
Mean rating 3.52986
Standard deviation of ratings is 1.12567359914
Ratings can be an integer from 1 to 5

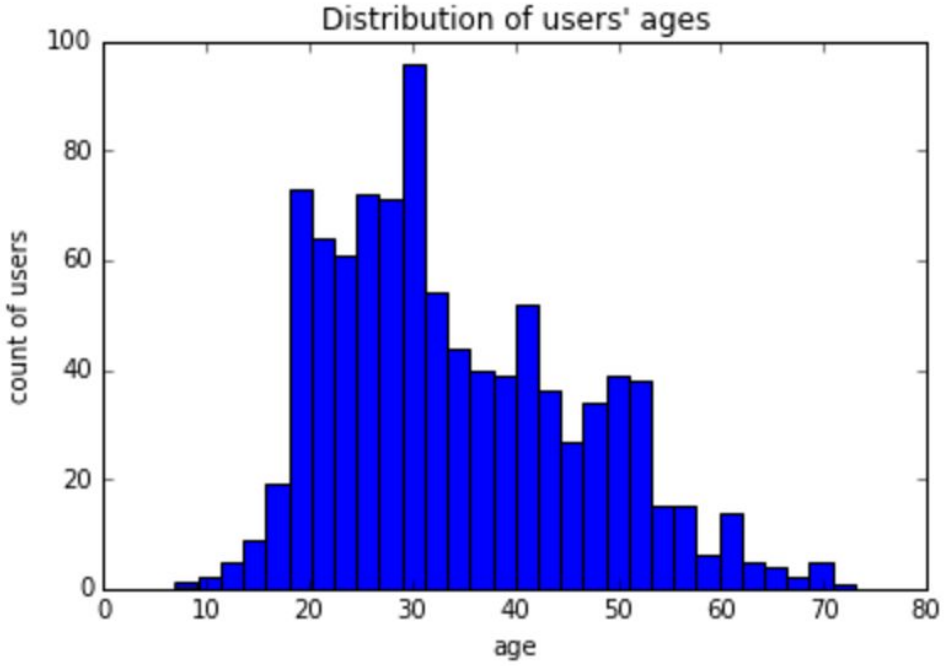
Then we calculate the Most rated movies and Top Rated movies

```
dtype: int64
                                  rating
                                    size      mean
title
Close Shave, A (1995)                112  4.491071
Schindler's List (1993)              298  4.466443
Wrong Trousers, The (1993)           118  4.466102
Casablanca (1942)                    243  4.456790
Shawshank Redemption, The (1994)     283  4.445230
Rear Window (1954)                   209  4.387560
Usual Suspects, The (1995)           267  4.385768
Star Wars (1977)                     583  4.358491
12 Angry Men (1957)                  125  4.344000
Citizen Kane (1941)                  198  4.292929
```

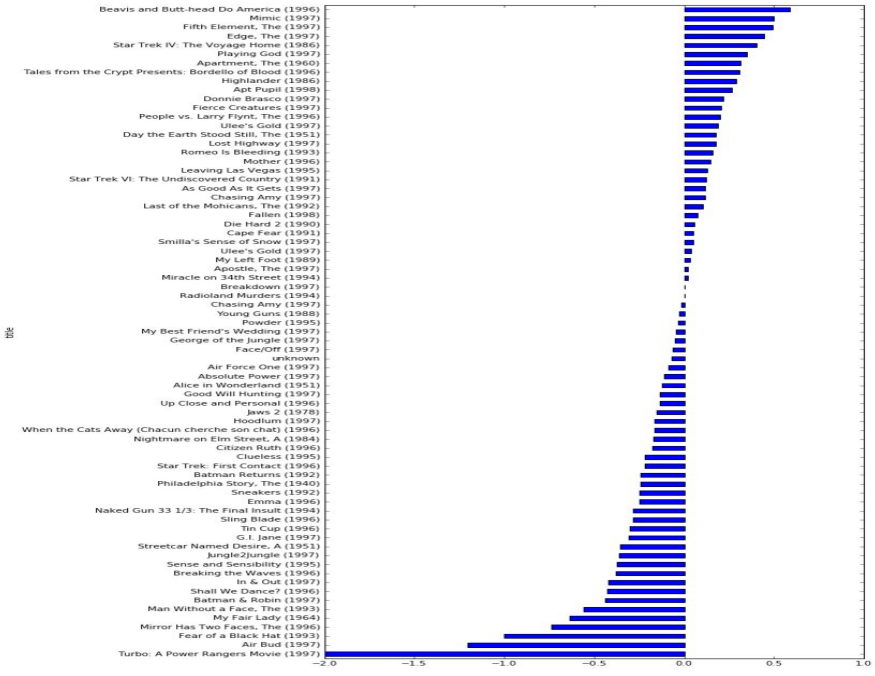
Top 10 rated movies in the dataset.

Most and top rated movies are useful they are represented as an option for how the user want to browse movies on their website for example he can browse them by seeing top rated movies like in IMDB top 250 movies list

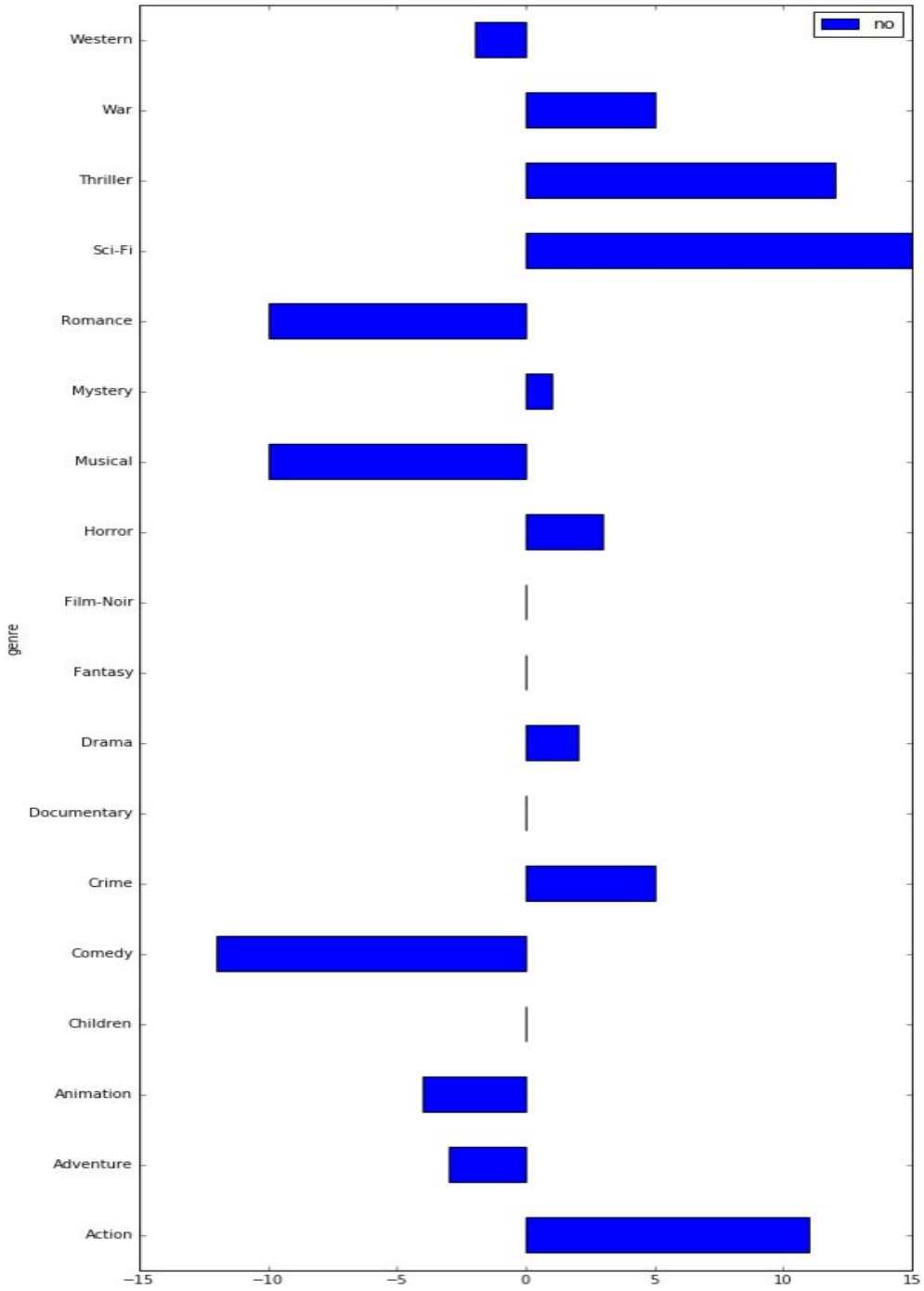
In order to understand the audience better i then plotted a histogram which shows a distribution of the users age, most users are about 30, after that it appears that much of the users are in their 20s, users older than 30 or younger than 20 are much less.



Then i took a look at male vs female different average ratings on top 100 movies, from this i decided to take it up a notch and determine the genre of each movie and conclude which genre is better based on whether the user was male or a female.



Male vs. Female Avg. Ratings for top 100 movies



Male vs. Female Movie Genres

The above figure shows movie genres and which genre is better for males(right) and females(left), so for example females tend to love comedy,musical movies but males tend to like action,sci-fi movies more than females do.This results are not used in this project but it's very important, it can be used in so many different ways if you hold user data you can already

recommend movies based on the above graph, or you can use the recommender system implemented here and give movies with genres that males like for males more weight and give genres that female like for females more weight

Algorithms and Techniques

First i took a subset of the dataset discussed above, i only used in this recommender system 3 features, user ID , movie ID , movie rating as they are what we need for the equations below.

There are two sub parts for the recommender system and that's how the algorithm works

1. Calculate similarity between current user and all the other users
2. Calculate prediction for a movie

Similarity

First we calculate a similarity matrix between the current user that u want to recommend movies for and all the other users, this similarity matrix shows how different users are similar/close/distant to the current user.
There are different ways to calculate similarity Euclidean distance , Pearson similarity score, Cosine similarity measure.
In this problem we use Pearson similarity score which results from 1 to -1, where 1 means highly correlated and -1 means the opposite
The symbol r_a corresponds to the average rating of user a, $r_{i,j}$ means rating for user i on movie j

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

After some research, analyses show that for user-based recommender systems – and at least for the best studied recommendation domains ,the Pearson coefficient outperforms other measures of comparing users,the Pearson measure considers the fact that users are different with respect to how they interpret the rating scale. Some users tend to give only high ratings, whereas others will never give a 5 to any item. The Pearson coefficient factors these averages out in the calculation to make users comparable.

Prediction

To make a prediction for *an item*, we now have to decide which of the neighbors' ratings we shall take into account and how strongly we shall value their opinions, prediction is done according to the below equation with uses the similarity calculated from the previous section

$$pred(a, p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a, b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a, b)}$$

After calculating the prediction for movies which are not yet seen by our current users, then sorting them in descending order with high ratings first, we then can recommend the top predicted movies in this list.

Benchmarks

During prediction we exclude any user who has 0 to -1 similarity with the current user as they are not highly correlated with the user.
There is no certain benchmark in recommending top predicted movies if it's a movies website so it depends on how many movies are represented in the current page if we represent 10 movies per page then we show the first 10 top predicted movies then if the user enter the next page we recommend the second 10 and so on, so it's basically traversing in a sorted array in descending order

Methodology

Data Preprocessing

There is no preprocessing done on the dataset, feature selection is done manually as the algorithm uses only user id, movie id and movie ratings.
The genres feature is a categorical feature with about 18 categories, they are already read as dummy variables meaning that they are already preprocessed.

Implementation

First we read the data from three different files, these files represent user information, movie information, and ratings
Then these files were merged into one single variable.
After the analysis part of the data, we start with a small building block, by making a function which calculates pearson similarity between two users

```
def pearson_sim(user_a,user_b):
```

This function takes user a and user b and returns a similarity score between 1 and -1 where 1 means that they are very similar and -1 means they are not similar.
This is done first by calculating each user movie ratings mean ra , rb, then using these values and minusing them from each user's movie ratings we are able to calculate the pearson similarity according to this equation.

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

The pearson_sim function is used in the below predict function.
Predict function takes user ID and movie ID and returns the predicted movie rating for this user

```
def predict(user_a,movie_id):
```

It loops on all users and calculates a similarity matrix using our previous similarity function between our current user and all the other users.
Then it predicts a rating using this equation

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Both functions are commented in the code section in the iPython notebook for further information

Refinement

There has been some code refinement as the algorithm really is very slow like removing duplicates to loop on a much less data, but it's still slow that's also because i used the whole data to test the Accuracy and RMSE.
There has been some refinements in how i evaluated the model written in the next section.

Results

Model Evaluation and Validation

```
accuracy score 0.2
RMSE 0.95
```

In the first implementation i used Accuracy to measure the performance of the algorithm but it results low accuracy (0.2) but after logging and debugging, the predicted ratings and their corresponding true ratings were closely similar,but closely similar is not truly similar in accuracy terms
After some research on some sources online i found that mainly recommender systems are tested whether by A/B test some process with and without recommendation and observe changes in key metrics like clicks, purchases, views for example.
or RMSE , I switched to calculate root mean squared error from the actual predictions as accuracy was very strict and the prediction results were really close to actual ratings so i found it is not a good metric, RMSE results 0.95 here, i tested on 20 movies.
The model have been tested on various input sizes and for various number of movies but 20 was the largest as the algorithm is too slow

```
reduced = train[:] #this is used while developing to reduce the size of the dataset if needed
```

This algorithm is considered a nearest neighbour algorithm so there's no training/testing data split, i worked on the whole dataset.

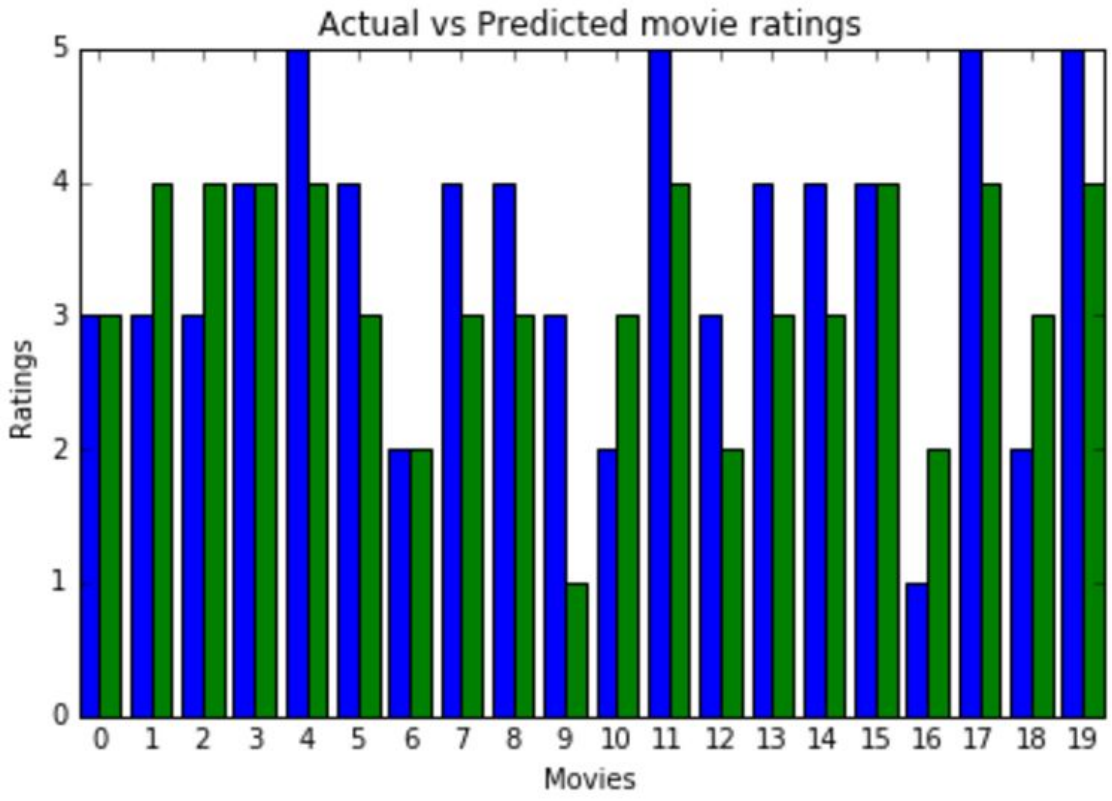
```
{880: (383, 1.0), 130: (450, 2.0), 171: (327, 3.0), 198: (237, 3.0), 230: (51, 3.0), 72: (493, 4.0), 522: (523, 4.0),
 299: (970, 3.0), 642: (581, 3.0), 874: (137, 4.0), 189: (276, 4.0), 848: (661, 4.0), 500: (619, 2.0), 405: (510, 2.
0), 246: (208, 3.0), 311: (778, 3.0), 664: (509, 4.0), 383: (137, 4.0), 234: (648, 3.0), 16: (76, 4.0)}
```

Final output user_id:(movie_id,rating)

The above figure show the final output for 20 movies(rmse = 0.95) we can easily sort them in descending order based on the rating and recommend them from top to bottom or recommend only movies of average or high ratings

Conclusion

Free-Form Visualization



Actual ratings are represented by blue lines and Predicted ratings are represented by green lines

The above figure shows the actual and predicted ratings for 20 movies, predicted results are pretty good, they may be slightly different by a scale of 1 rating but generally the flow of both graphs is the same where high ratings are predicted as high and vice versa,
There is only a large deviation in movie 9 where the actual rating was 3 but it was predict as 1

Reflection

In this Recommender system we calculated correlation between a user and each other user in the system, these weights represent similarity and how much should a user value other user's opinion, based on these weights we predict the estimated rating of a movie for a certain user. Opinions of users with high weights will be much more important than opinions of users of low weights and that was demonstrated in the prediction equation.
The goal of this project was to predict movie ratings for user, with these predictions many things can be accomplished and will be discussed in the improvements section
Other than the recommender system, user can query most rated items or top rated items, they sometimes are useful and needed in some applications
The algorithm is also slow and not real time, meaning that these predictions should be calculated and stored offline, presented for the user when needed.

Improvement

- Improve the time complexity of the algorithm to run faster.
- Integrate gender in the algorithm, create weights from the graph "Male vs. Female Movie Genres" and integrate these weights in the prediction function,where we will multiply by a weight based on user gender and movie genre as we discovered that some movies genres are strongly correlated with males while others are strongly correlated with females
- Implement item based collaborative filtering
    - Since we scan very large number of users, the need to scan this vast number of potential neighbors makes it impossible to compute predictions in real time
    - The main idea of item based CF is to calculate similarity between items not similarity between users, it's an interesting problem and similar to this one.

References

Recommender Systems: An introduction by Dietmar Jannach
http://www.slideshare.net/MrChrisJohnson/algorithmic-music-recommendations-at-spotify
http://shop.oreilly.com/product/9780596529321.do