Machine Learning (CS 6360)
PROJECT REPORT
Recommendation System using Collaborative FIltering

By
Diksha Chhabra

1. Abstract

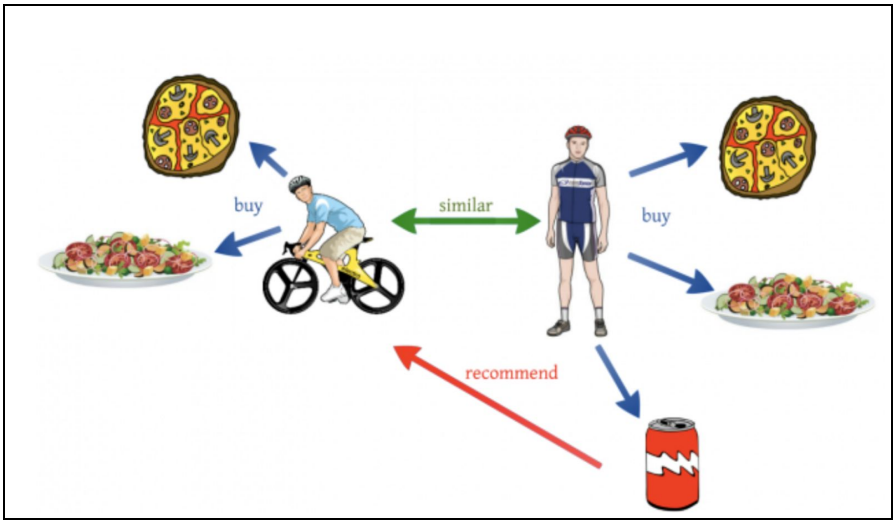
With the arrival of movie websites like Netflix, IMDb, etc, there is a huge collection of data about the movie ratings and the users who rated. This data could be used for finding different trends in the movie dataset like choices made by a user, top rated movies in all genre and also for suggesting movies to a user or designing a recommendation system. Collaborative filtering is one of the well known techniques in recommendation systems. The idea is to predict how a given user will predict an item given the ratings of other users. This is divided into two categories: Memory based and Model based.
In this project, we are comparing the two methods of Memory-based Collaborative filtering which are user-user and item-item based collaborative filtering. The former deals with finding the users similar to the target user and then taking the weighted average of their ratings for movie recommendation. The latter is finding those movies similar to the target movie based on the ratings given by the target user on those movies.These methods are combined with KNN for clustering K similar user/movies. We have used Pearson coefficient as a similarity measure and RMSE as an evaluation metric for choosing an optimal ‘K’ value for KNN.

2. Introduction

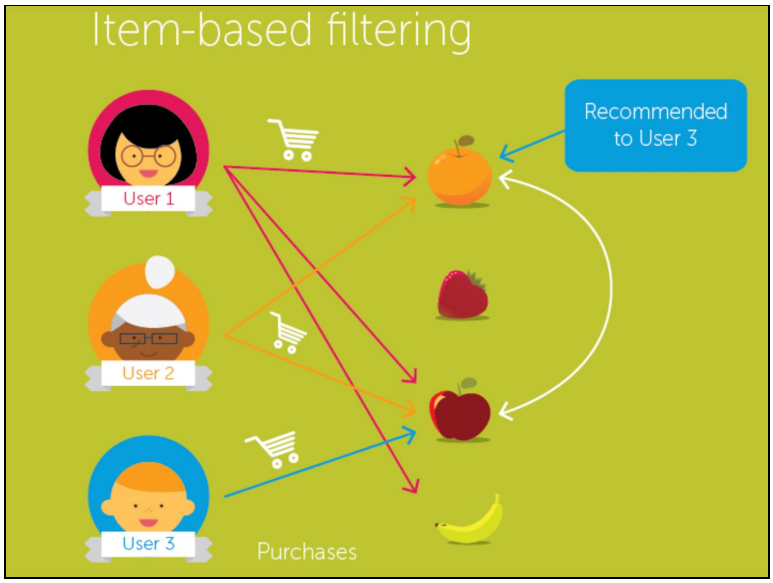
Recommendation system is finding and suggesting to the user something in which he might be interested. In movie recommendation system, we have used the method of collaborative filtering for finding the movies which the user might be interested to watch. Collaborative filtering is a method of making predictions about the interests of user by analysing the taste of users which are similar to the said user. The underlying assumption of the 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 as shown in the figure below:

We have compared two methods of collaborative filtering:

1. **User-user based** : Finding the users similar to the target user and then taking the weighted average of the ratings given by them on the target movie whose rating by the target user is to be predicted.



2. Item-Item based : Finding the movies similar to the target movie based on the ratings given by the target user on those movies snf then taking the weighted average to find the rating on given movies.



For finding the similarity, we have used the Pearson's coefficient. After the coefficients are calculated we have taken the top K users(in the case of user-user) or top K movies (in the case of item-item) and perform a weighted average to calculate the predicted rating on a given movie.
Since we don't have a training set of the form <x,y>, this is an unsupervised learning algorithm.

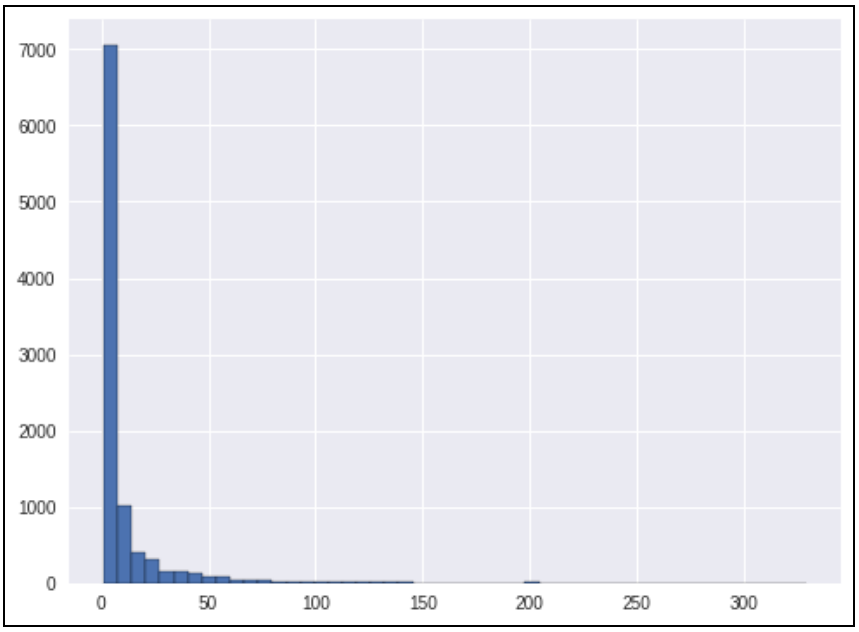
3. Data Set:

We used the data from the MovieLens website for education and development. The data contains those users who have rated at least 20 movies:

No. of Users	610
No. of Movies	9742
Total Ratings	1,00,836

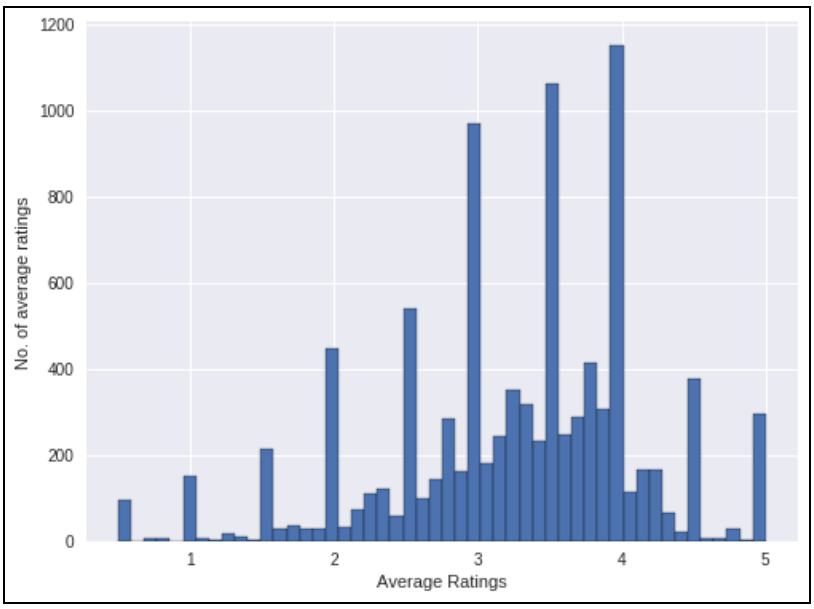
➔ Data analysis:

a. Histogram of number movies rated by the number of ratings.

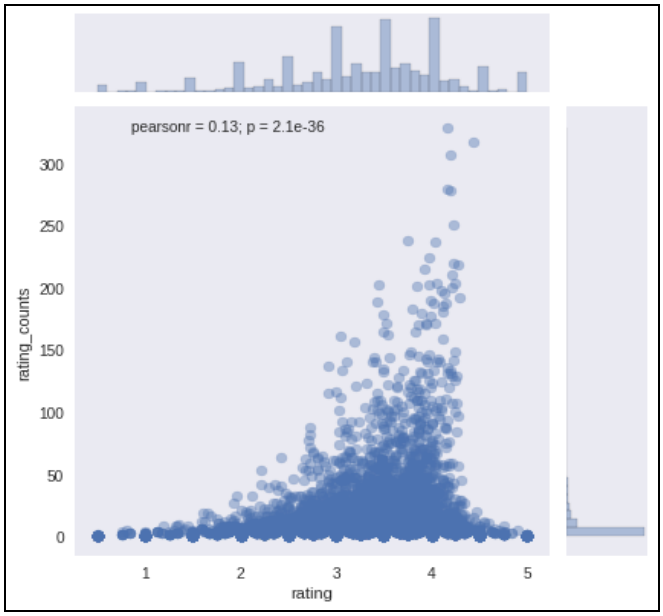


- From the output, we can see that most of the movies have received less than 50 ratings. While the number of movies having more than 100 ratings is very low.

b. No. of avg ratings vs no. of ratings.



- We can see that the integer values have taller bars than the floating values since most of the users assign rating as integer value i.e. 1, 2, 3, 4 or 5.
- It is evident that the data has a weak normal distribution with the mean of around 3.7. There are a few outliers in the data.
- The graph shows that, in general, movies with higher average ratings actually have more number of ratings, compared to movies that have lower average ratings as can be seen in the figure below.



4. Approach

The following is the approach that we took for comparing the two methods of building a recommendation system.

- We start by selecting a target user to whom we want to recommend the movies.
- Once the target user has been selected we take the top 100 movies(based on the highest average rating) which are not seen by the target user.
- From these top 100 highest average rated movies, we suggest the similar ones to the target user.
- Then we perform the following two algorithms on the dataset for predicting the target user's rating on the movies he has not seen.

1) User-User based algorithm:

The user based collaborative filtering finds a cluster of k nearest neighbors who are similar to target user using the pearson correlation coefficient. The pearson coefficient finds the similarity using the set of movies which are rated by target user and the set of movies which are not rated by target user.

Pearson coefficient(sim(u,v)) for user-user algorithm is given as

$$sim(u,v) = \frac{\sum_{i \in com(u,v)} \left(r_{u,i} - \bar{r_u}\right)\left(r_{v,i} - \bar{r_v}\right)}{\sqrt{\sum_{i \in com(u,v)} \left(r_{u,i} - \bar{r_u}\right)^2} \sqrt{\sum_{i \in com(u,v)} \left(r_{v,i} - \bar{r_v}\right)^2}}$$

Where $r_{u,i}$ be the rating of the i^{th} item under user u, r_u be the average rating of user u, and $com(u,v)$ be the set of items rated by both user u and user v.

The top K users with max similarity with the target user are taken.
The predicted rating for the movie which is not seen using the rating given by k-nearest neighbors can be given as

$$p_{u,i} = \bar{r_u} + \frac{\sum_{v \in nn(u)} sim(u,v) \cdot \left(r_{v,i} - \bar{r_v}\right)}{\sum_{v \in nn(u)} |sim(u,v)|}$$

Where $nn(u)$denote the set of k-NN to u.

2) Item-Item based algorithm:

The item based collaborative filtering finds a cluster of k nearest items which are similar to movie not seen by the target user using the pearson correlation coefficient. The pearson

coefficient finds the similarity using the set of movies which are rated by target user and the set of movies which are not rated by target user.

Pearson coefficient(sim(u,v)) for item-item algorithm is given as

$$sim(i,j) = \frac{\sum_{u \in com(i,j)} (r_{u,i} - \bar{r_u})(r_{u,j} - \bar{r_u})}{\sqrt{\sum_{u \in com(i,j)} (r_{u,i} - \bar{r_u})^2} \sqrt{\sum_{u \in com(i,j)} (r_{u,j} - \bar{r_u})^2}}$$

Where i is the movie rated by target user, j is the movie not rated by target, $r_{u,i}$ be the rating of the i^{th} item under user u, r_u be the average rating of user u, and com(i,j) be the set of items similar to i and j.

Predicted rating for the ith item of user u using item-item similarity is given by

$$p_{u,i} = \frac{\sum_{j \in nn(i)} sim(i,j) \cdot (r_{u,j})}{\sum_{j \in nn(i)} |sim(i,j)|}$$

Where $nn(u)$denote the set of k-NN to u.

3) Evaluation Metric : RMSE(Root Mean Squared Error)

To find the efficiency of the model, we have used RMSE as a measure to find the difference between the predicted and original value.

- From the movies rated by the target user, 10% of the movies are taken as not rated and we used this as the test set .
- Remaining 90% of the movies are taken as training set which we used for both our algorithms.
- RMSE is given by the formula as

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2}$$

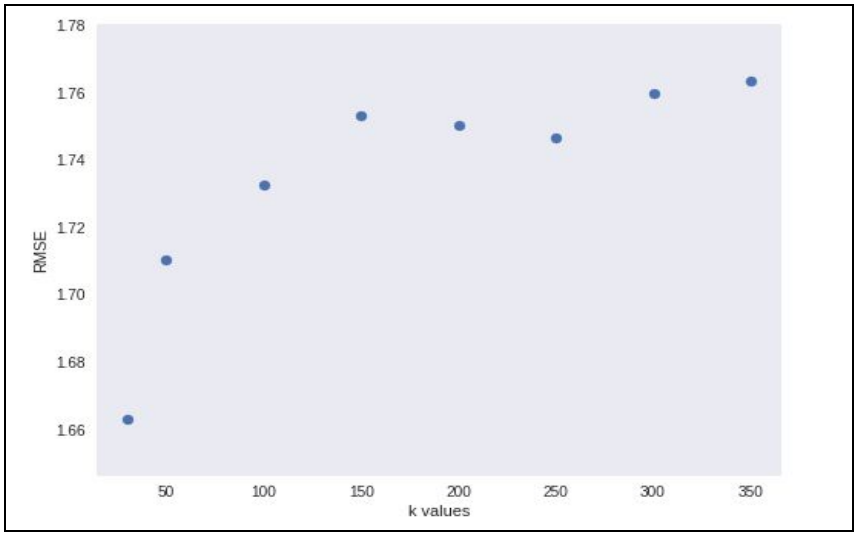
Where y_j is the original value and y_j is the predicted value and n is the number of observations. We ran this experiment for multiple times and found out the optimal value of k (k-nearest neighbors) which gives the least RMSE.

- We calculated RMSE values using different K values in both the algorithms.
- The K-Values are taken as [30.50,100,150,200,250].
- The value of K which gives the least RMSE is taken for prediction on the movies not seen by the target user.
- At last, we show a list of top 10 movies which should be recommended to the target user.

5. Experimental Results and Observations:

1) **User-user algorithm**→ The following is the K vs RMSE table.
    - The least value of RMSE is at K=30, i.e, the weighted average of ratings of 30 users gives similar results as that of our target user

	k1	RMSE1	
0	30	1.69930863	
1	50	1.710397681	
2	100	1.732754967	
3	150	1.753284681	
4	200	1.75046292	
5	250	1.746642826	
6	300	1.760054289	
7	350	1.763771191	



RMSE vs K plot (user-user)

→ Highest rated predicted movies at <u>smallest</u> RMSE: k = 30

	Unnamed: 0	movieId	predictedRating	title	genres
0	0	131098	4.805598	Saving Santa (2013)	Animation|Children|Comedy
1	1	50	4.422825	Usual Suspects, The (1995)	Crime|Mystery|Thriller
2	2	131104	4.305598	The Brain (1969)	Comedy|Crime
3	3	31	4.181749	Dangerous Minds (1995)	Drama
4	4	16	3.742857	Casino (1995)	Crime|Drama
5	5	32	3.727687	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
6	6	1	3.648345	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
7	7	39	3.332003	Clueless (1995)	Comedy|Romance
8	8	2	3.197514	Jumanji (1995)	Adventure|Children|Fantasy
9	9	10	2.994588	GoldenEye (1995)	Action|Adventure|Thriller

→ Highest rated predicted movies at <u>highest</u> RMSE: k = 150

```
150
  movieId  predictedRating                                        title  \
0  131098         4.805598                                Saving Santa (2013)
1  131104         4.305598                                  The Brain (1969)
2      29         4.026120  City of Lost Children, The (Cité des enfants p...
3      50         3.809021                          Usual Suspects, The (1995)
4      28         3.679833                                   Persuasion (1995)
5       1         3.641221                                   Toy Story (1995)
6       6         3.591779                                        Heat (1995)
7      45         3.552965                                   To Die For (1995)
8      32         3.499852          Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
9      13         3.387458                                       Balto (1995)
```

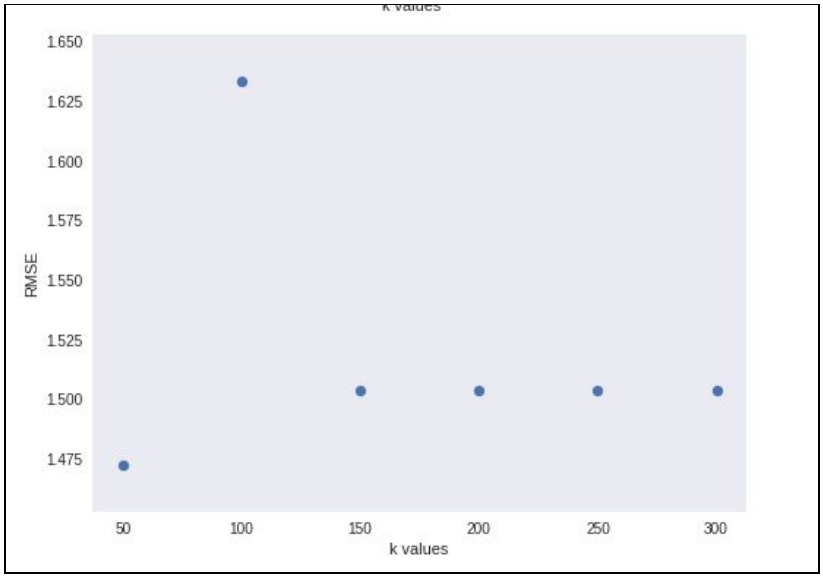
<u>Observation:</u>

- The movies in both highest and least RMSE values are almost similar and there's not much difference in the recommendation for the target user.
- As the value of k increases, the Root mean squared error increases. This means more **overfitting** occurs at higher K's.

2) Item item algorithm → The following is the K vs RMSE table.

- The least value of RMSE is at K=50, i.e, the weighted average of ratings on 50 movies gives similar results as that of our target user

	k1	RMSE1
0	50	1.47253
1	100	1.63322
2	150	1.50372
3	200	1.50372
4	250	1.50372
5	300	1.50372



RMSE vs K plot(item-item)

→ Predicted Movies on Least RMSE: k = 50

	movieId	predictedRating	title
0	46	4.157561	How to Make an American Quilt (1995)
1	22	3.773549	Copycat (1995)
2	24	3.722585	Powder (1995)
3	31	3.565487	Dangerous Minds (1995)
4	26	3.563641	Othello (1995)
5	44	3.519739	Mortal Kombat (1995)
6	27	3.497783	Now and Then (1995)
7	39	3.493073	Clueless (1995)
8	15	3.487994	Cutthroat Island (1995)
9	48	3.484763	Pocahontas (1995)

→ Predicted Movies on Highest RMSE: k =100

	Unnamed: 0	movieId	predictedRating	title	genres
0	0	1	3.382931	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	2	3.394268	Jumanji (1995)	Adventure|Children|Fantasy
2	2	3	3.876176	Grumpier Old Men (1995)	Comedy|Romance
3	3	4	3.258253	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	4	5	3.341076	Father of the Bride Part II (1995)	Comedy

Observation

- As k increases, the RMSE first increases then decreases.
- There are almost no common movies between least and highest RMSE values' prediction. This means here RMSE metric is effective in predicting the movie ratings.

6. Conclusion

By using the RMSE for both the algorithms we are able to find a 'good' K for KNN to use it for predicting the ratings on the movies.

- We find that more the value of K, i.e, more the number of users to be taken for calculating weighted average, more is the RMS Error. This means there are lots of outliers in the data which when are incorporated for calculating the prediction cause underfitting.
- Also, since the overall RMSE values of Item-Item algorithm are less than the overall RMSE values for User-User algorithm, the Item-Item approach seems to be a good choice over User-User for predicting ratings in this dataset.
- User-User method's result(recommended movies) doesn't vary much with different choices of K.
- Item-Item methods' results(recommended movies) vary a lot as we saw in the case of least and highest RMSE where none of the recommended movies were common between their results.

7. Future Work

- Modify the model to find the least RMSE by using different similarity measures like cosine similarity.
- Adding Neural Network based approach to solve the recommendation problem.
- To apply ensemble learning techniques (a mixture of user-user and item-item algorithms) for enhancing the recommendation performance.

8. References

1) Harper, F. Maxwell, and Joseph A. Konstan. "The movielens datasets: History and context." *Acm transactions on interactive intelligent systems (tiis)* 5.4 (2016): 19.

2) Movie Dataset from → https://grouplens.org/datasets/movielens/

3) Cui, Bei-Bei. "Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm." *ITM Web of Conferences*. Vol. 12. EDP Sciences, 2017.

4) Nikolov, Dimitar, and DongInn Kim. "Learning to Predict Movie Ratings from the Netflix Dataset."