1. INTRODUCTION:

Recommender systems (RS) are used to help users find new items or services. They usually use machine learning algorithms to predict the probability of the preference of a user to a specific product or service and give them appropriate recommendation according to that predicted preference. Different algorithms are available for this purpose, and choosing the best one for a specific problem needs evaluating different ones. This paper evaluates the capability of two different algorithms that widely used in recommender systems, k-nearest neighbors (KNN) (unweighted and weighted) and UIIman collaborating filtering (CF), for analyzing and making predictions for different movies based on one of the MovieLens datasets. The data set consist of the rates for more than 9000 unique movies from 670 users.

2. METHODS:

2.1. KNN:

KNN is one of the most famous algorithms for both classification and regression. It uses k closest training examples in the feature space for this purpose. It is known as a type of lazy learning, because no model is built during its process. In this study Euclidian distance metric is used for finding k closest neighbors for each piece of data. Figure 1 shows how each neighbor is defined in this method. In this study, if k example were not available for a specific data, the k was reduced to the actual number of examples. Moreover, two different methods are considered, unweighted and weighted method.

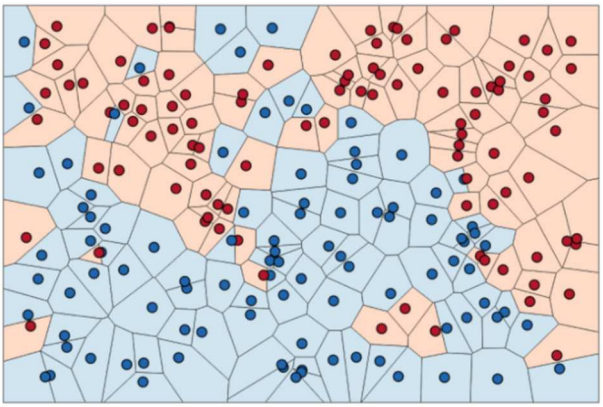

Figure 1 Kevin Zakka's blog (kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

2.1.1. Unweighted:

In this method, as it can be seen in Eq.1, the average of the k nearest neighbors is considered as the predicted rating.

$$r^*_{u,m} = \sum_{i=1}^{k} \frac{r_{i,m}}{k} \qquad \text{Eq.1}$$

In this study, in the case that the k^{th} neighbor had a distance similar to the $k+1^{th}$ neighbor (and the k+2, etc), the k expanded such that it included all the ties.

2.1.2. Weighted:

Assign weight to the contributions of the neighbors, can improve the results. In this method the nearer neighbors contribute more to the final rating than the more distant ones (Eq.2 and Eq.3).

$$r^*_{u,m} = \frac{\sum_{i=1}^{k} r_{i,m} * w_i}{\sum_{i=1}^{k} w_i} \qquad \text{Eq.2}$$

if i = 1	$w_i = 1$
otherwise	$w_i = \frac{d(x', x_k^{NN}) - d(x', x_i^{NN})}{d(x', x_k^{NN}) - d(x', x_1^{NN})}$

Eq.3

$r^*_{u,m}$, $r_{i,m}$, x', d(x,y), x_i^{NN} are the predicted value for user *u* on movie *m*, rating for user *i* and movie *m*, the vector of ratings for user *u* (the user we are currently predicting), the Euclidian distance between two vectors x and y, the ratings vector for the i^{th} nearest neighbor, respectively.

2.2. Collaborative Filtering (CF)

Collaborative filtering is the process of filtering for likelihood of two or several items and give a recommendation to a specific user from the information gleaned from many other users. In this study, UIIman method is utilized. In this method, the similarity between two items/movies (i,j) is defined in Eq.4

$$Sim_{i,j} = \frac{\sum((r_{u,i}-R_m)(r_{u,j}-R_m))}{\sqrt{\sum(r_{u,i}-R_m)^2}\sqrt{\sum(r_{u,j}-R_m)^2}} \quad \text{Eq.4}$$

where R_m is the average of the movies.

This study considers two different cases when the user did not have any other movies that also have a similarity score. First, predicted a score of "0.0"; second, predicted the average score "2.5".

3. RESULTS

3.1. KNN:

Three different "k"s were investigated (k=3, k=5, and k=10). The best choice of k depends on the data. By adopting larger values of k, the effect of noise on the classification will be reduced. However, larger k decreases the distinction of the boundaries between classes.

3.1.1. Unweighted:

The run time and RMSE for unweighted KNN for each run is presented in table 1.

Table 1 Run time and RMSE for unweighted KNN

	K=3	K=5	K=10
Run time (sec.)	82	82	84
RMSE	1.02	1.04	1.00

As it can be seen, higher k does not always produce better results. In this study, k=3 was slightly better than k=5. However, k=10 was the best by RMSE=1.

3.1.2. Weighted:

The run time and RMSE for weighted KNN for each run is presented in table 2.

Figure 2 compares the RMSE for both weighted and unweighted KNN for different Ks. For weighted KNN, as it's obvious in figure 2, larger k resulted better predictions for our data set.

Table 2 Run time and RMSE for weighted KNN

	K=3	K=5	K=10
Run time (sec.)	86	84	84
RMSE	1.23	1.12	1.04

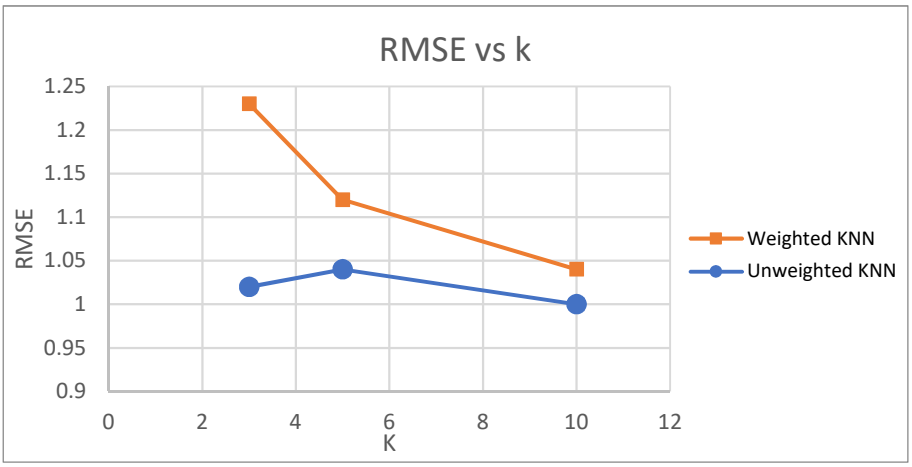

Figure 2 RMSE vs K for weighted and unweighted KNN

3.2. Collaborative Filtering (CF)

For CF, four test sets, 10%, 20%, 30% and 40% were selected. The run time and RMSE for each set is represented in table 3 for the case that a score of "0.0" was predicted when the user did not have any other movies that also have a similarity score.

It can be seen, as the test data become larger, more time was needed for the calculation. For example, there are twice piece of data in 20% for each piece of data in 10% data set. So, it needed twice time for calculation.

Table 3 Run time and RMSE for CF for the case that a score of "0.0" was predicted when the user did not have any other movies that also have a similarity score.

	10%	20%	30%	40%
Run time (sec.)	699	1306	2042	2838
RMSE	0.98	0.98	1.01	1.01

Table 4 shows the run time and RMSE for each set for the case that score of "2.5" was predicted when the user did not have any other movies that also have a similarity score.

Table 4 Run time and RMSE for CF for the case that a score of "2.5" was predicted when the user did not have any other movies that also have a similarity score.

	10%	20%	30%	40%
Run time (sec.)	795	1141	2084	2862
RMSE	0.938	0.933	0.940	0.948

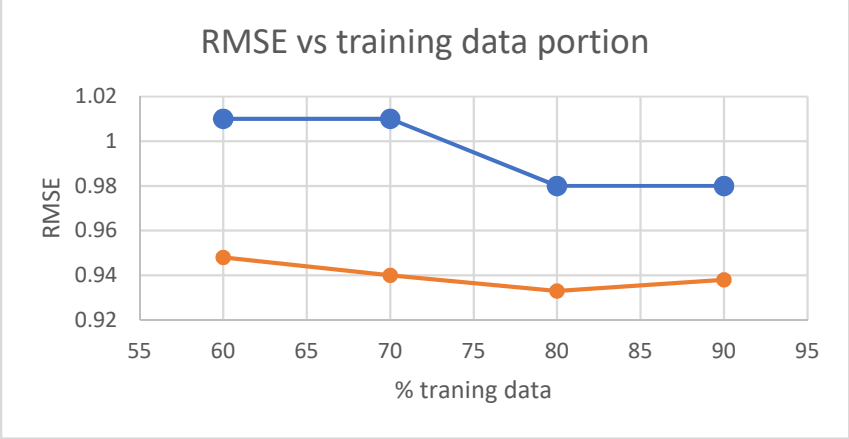

Figure 3 RMSE vs training data size

Figure 3 shows the RMSE for CF vs training data size. Moreover, it compares the cases that score of “0” and “2.5” was predicted when the user did not have any other movies that also have a similarity score. Larger training data size, resulted better rating for our data.

3.3. COMPARING THE RESULT

By comparing the results, it can be concluded that while CF had larger run time, it’s results were more accurate. This better performance might be because of difference between the number of users and the number of movies. While we predict according to the close distance between users in KNN, the distance between the movies is used for CF. larger data for movie increase the chance of finding more similar movies. On the other hand, this difference between the number of users and the number of movies increase the run time for CF. In other words, for finding rating for a similar user in KNN, the algorithm search through the other users, but, for finding the most similar movie to the current movie, the algorithm should search all other movies, which are about 15 times of the users.

4. CONCLUSION

While machine learning helps to improve recommendation systems, finding the best method and the appropriate parameters for that method needs a lot of investigation. As this study showed, higher values for “k” in KNN method, is not always effective. Moreover, choosing the best between weighted and unweighted KNN depends on the data. On the other hand, while CF produce much better results, it needs more run time. All in all, finding the best algorithm for a recommender system, is different for each data set, and, needs its own investigation.