Simple Recommender System for Movies

CS 504- Principles of Data Management and Mining

Final Project Report

Venkata Jayandra Kumar Lade

G01046700

Contents

Problem Statement: 3

Project Goal: 3

Dataset: 3

Preprocessing: 3

Technical Approach: 4

K Nearest Neighbor (KNN): 4

1) KNN- Unweighted Prediction Model: 4

2) KNN- Weighted Prediction Model: 5

Collaborative Filtering(CF): 5

Sarwar/Karypis method: 6

Results: 7

i) K-Nearest Neighbors: 7

ii) Collaborative Filtering: 8

iii) Dataset divided into 90% train and 10% test: 10

iv) ALL Models: 10

Conclusion: 11

Problem Statement:

Movielens is a non-commercial, personalized movie recommendations. It helps in finding the movies which you may like. Initially, the user should build the profile based on his tastes. Then it recommends other movies to watch. The purpose of the project is to predict the ratings of unseen movies by a user and recommend the movies based on prediction.

Project Goal:

The goal of this project is to use two methods called K-nearest neighbors(KNN) and collaborating filtering. Using these two-methods recommender systems will be build.

Dataset:

This data belongs to Movielens website which is obtained from https://grouplens.org/datasets/movielens/

The dataset contains 100000 ratings which are for 9216 movies with 671 users. The dataset contains user id, movie id, ratings, and timestamp.

Preprocessing:

The data set is modified by normalizing the userIds and movieIds. They were normalized because initially movieIds were not assigned continuously and as a result they were more than 9216 movieId's. By normalizing the userids and movieids were in range of 0 to n-1.

The dataset is divided into trainset and test set. There were divided in a way that if a userid is in test set then there must be at least on example in the trainset. Same for movieid, if a movieid is referred in test set, then it must have at least one other user who have rated same movie in trainset.

Technical Approach:

K Nearest Neighbor (KNN):

KNN is a method which is non-parametric used for both classification and regression. In this project KNN regression is used which means the output would a value. And the value is average of K nearest neighbors.

KNN is user-user based recommendation system. Euclidean distance is calculated to find the nearest users (And there is no need to calculate distance to every users). Here the process is to find similar users, they are who have rated the similar movies which user is already rated. So, how to calculate Euclidean distance?

Mathematically, Euclidean distance is the straight-line distance two points. The formula for Euclidean distance is:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Where

- ➔ p and q are two users.
- ➔ $p_1, p_2, \ldots . p_n$ are the ratings of movies 1,2…n by user p
- ➔ $q_1, q_2, \ldots . q_n$ are the ratings of movies 1,2…n by user q

In KNN there two types of averaging, they are:

1) KNN- Unweighted Prediction Model:

   In this model it takes the average of K-nearest neighbors without considering any weights. The formula for this model is:

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

- $r^*_{u,m}$ is the predicted rating for user u on movie m
- $r_{i,m}$ is the rating for user i on movie m
- k is the value how many nearest neighbors to be considered.

2) KNN- Weighted Prediction Model:

In this model it takes the weighted average of K-nearest neighbors. The formula for this model is:

$$r^*_{u,m} = \frac{\sum_{i=1}^{k} r_{i,m} w_i}{\sum_{i=1}^{k} w_i}$$

- $r^*_{u,m}$ is the predicted rating for user u on movie m
- $r_{i,m}$ is the rating for user i on movie m
- k is the value how many nearest neighbors to be considered.
- For w_i :

  If i=1 then $w_i = 1$

  Otherwise $w_i = (d(x', x^{NN_k}) - d(x', x^{NN_i})) / (d(x', x^{NN_k}) - d(x', x^{NN_1}))$

- d(x,y) is the Euclidean distance of user x and y
- x^{NN_i} is the ratings for the i^{th} nearest neighbor.

Collaborative Filtering(CF):

Collaborative filtering is a method for making predictions of a user based on collecting the information about many other users. In general, there would be two approaches, they are:

1) **User-user Collaborating Filtering:** In this method, similar users should be identified who have rated same item that should predicted for the user. Using the ratings of similar user on the same item, the rating is predicted or calculated by their averages.
2) **Item-Item Collaborating Filtering:** In this method, ratings are estimated or predicted based on the similar items.

Sarwar/Karypis method is used for this project.

Sarwar/Karypis method:

This method works on the item-item based collaborative filtering. Here the similarity matrix is calculated based on the formula shown below.

$$sim(i,j) = \frac{\sum_{u \in U}\left((r_{u,i} - \overline{R_u})(r_{u,j} - \overline{R_u})\right)}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{R_u})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{R_u})^2}}$$

Where

- -1.0 <= sim(i,j) <= 1
- i refer to movie i and j refers to move j
- $r_{u,I}$ is the rating for user u on movie i
- U is the set of users who have rated both movie i and movie j.
- $R_u$bar is the average ratings assigned by user u.

This method uses weighted average technique to calculate the rating. And the formula is listed below:

$$r^*_{u,i} = \frac{\sum_{j \in N}(sim(i,j) * r_{u,j})}{\sum_{j \in N} sim(i,j)}$$

- $r^*_{u,I}$ = prediction for user u on movie i
- sim(i,j) is the similarity between movies i and j
- N is the set of movies that are similar to move I and which has sim(i,j) >0

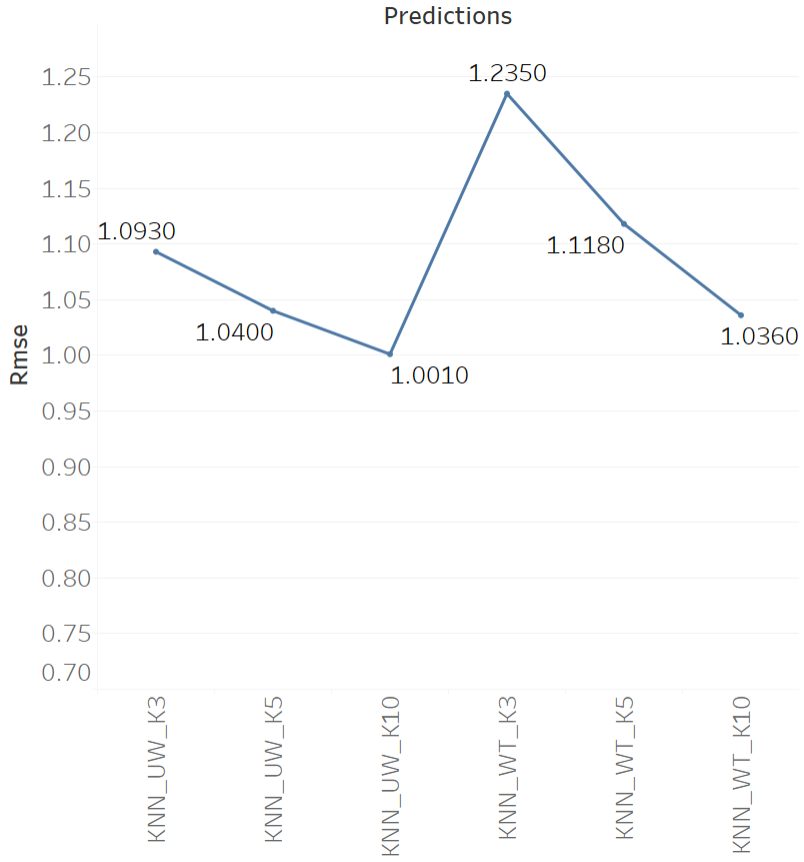
Results:

i) K-Nearest Neighbors:

Initially we will discuss about the results of K-nearest neighbors. Accuracy is observed based on root mean square value of predicted ratings with original ratings. The low rmse means it has best accuracy. Here, KNN is performed on dataset which is divided into 90% train set and 10% test set. For this dataset KNN is performed for 3,5 and 10 nearest neighbors and averages are calculates using both weighted and unweighted. The time to taken for each model and rmse values are shown in below table.

Model	**RMSE**	**Time Taken**
KNN_10_UW_K3	1.093	17 minutes 39 seconds
KNN_10_UW_K5	1.04	17 minutes 45 seconds
KNN_10_UW_K10	1.001	17 minutes 50 seconds
KNN_10_WT_K3	1.233	16 minutes 15 seconds
KNN_10_WT_K5	1.114	16 minutes 20 seconds
KNN_10_WT_K10	1.033	16 minutes 22 seconds

KNN



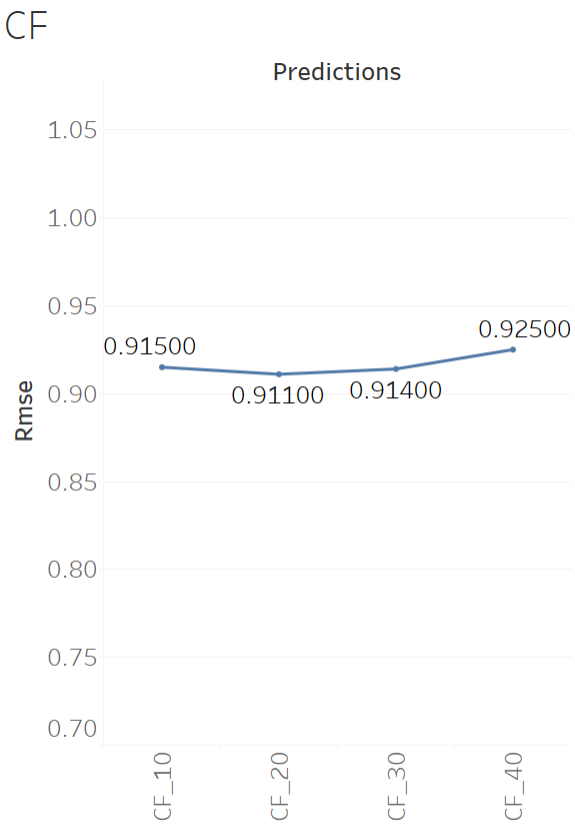
By observing the graph, KNN with unweighted average of 10 nearest neighbors has low RMSE value. Which means the best among KNN is KNN with unweighted average of 10 nearest neighbors. By observing the graphs there is increase in RMSE values when changing the unweighted average into weighted average. So, changing averages into weighted will not give you a better result. And again, observing separately for both unweighted and weighted average, increasing K value decreases rmse value which means increasing in accuracy.

ii) Collaborative Filtering:

For collaborative filtering also, the accuracy is calculated based on RMSE values. Here the collaborative filtering is done four type of data sets. First one is 90% of data used for training and remaining 10 % for testing. Second one is 80% of data used for training and remaining

20 % for testing. Third one is 70% of data used for training and remaining 30 % for testing. Fourth one is 60% of data used for training and remaining 40 % for testing. The below table shows the RMSE value and time taken for each dataset.

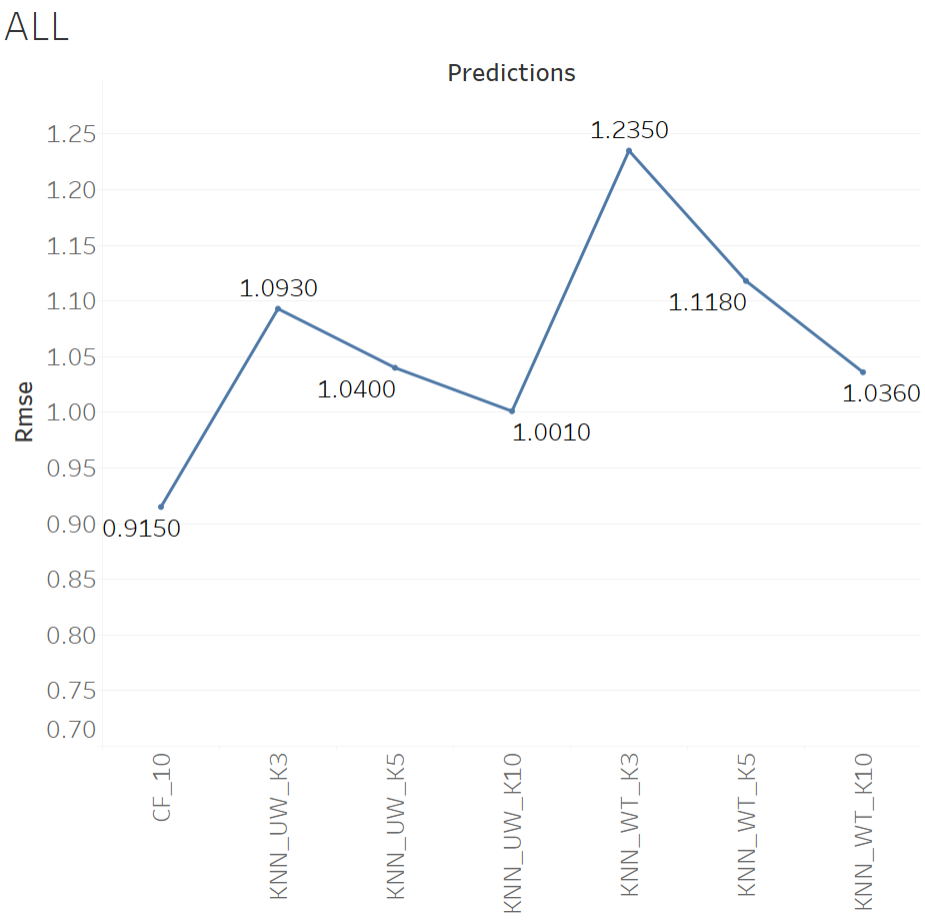
DATASET	**RMSE**	**TIME TAKEN**
CF_10	0.915	17 minutes 46 seconds
CF_20	0.911	27 minutes 52 seconds
CF_30	0.914	49 minutes 45 seconds
CF_40	0.925	47 minutes 8 seconds



Observing the graph, collaborative filtering works or achieve less RMSE for dataset which is divided into 80% for training and 20% testing.

iii) Dataset divided into 90% train and 10% test:

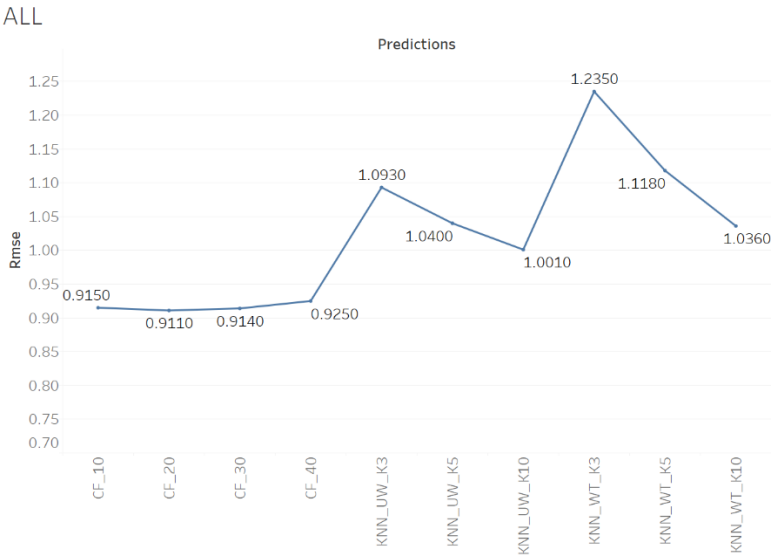
For dataset which is divided into 90% for training and 10% test is used for collaborative filtering and KNN for 3,5,10 nearest neighbors with both weighted and unweighted average. Observing the graph below, Collaborative filtering have less RMSE value which means it has high accuracy.



iv) ALL Models:

By observing the RMSE values for all the models generated on different percentage split of dataset. Collaborative filtering done on data which is divided into 80% training and 20% test

set has low RMSE value which it has high accuracy. And data which is divided into 90% and 70% training set also have similar RMSE values with little difference.



Based on the run time and RMSE values, collaboration filtering for data divided into 90% training data and 10% as test data is best model. Even tough collaboration filtering for data divided into 80% train and 20% test set has less RMSE value. The CF on 90%training and 10%testing is best because the difference of RMSE between them is 0.003 which can be neglected when comparing it with run time also. Since run time difference between the models is approximately 10 minutes which is a high difference.

Conclusion:

From the results shown above there some interesting insights can be taken. They are as k-value increases the accuracy of the model also increases. And for collaborative filtering also as training set increases accuracy increases. Combining the RMSE values and run time, collaborative filtering for data which is divided into 90% training and 10% testing set is the best model.