Movie-Lens Recommendation System

Cpe 695: Applied Machine Learning–Team 7

Fan Luo
10442682
fluo4@stevens.edu

Yiran Lyu
10441886
ylyu3@stevens.edu

Jiarong Xia
10433223
jxia10@stevens.edu

***Abstract*—This report is based on Movie-Lens 20M data set to build a movies recommendation system using three different algorithm.**

I. Introduction

Movie, as a creative industry of politics, economy and culture, has been an important part of people's life. With the development of Machine Learning, many different movie recommendation systems appeared. They make recommendations to people based on the data of both users' history record and movies' information. Users could find more movies they may be interested in automatically, and movie makers could using such a system to attract people watching more movies. In this project, our team aim to build a movie recommendation system based on three different methods – Decision Tree, K-means clustering, K- nearest neighbor. The system could determines whether a movie will be recommended to the user or not, and for a certain movie, recommend 10 or more similar movies.

II. Related Work

A recommender system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.[1]

Recommender systems are utilized in a variety of areas, it could be a content recommenders for social media platform like Facebook, a playlist generator for videro services like YouTube or a product recommender for Amazon. The famous movie recommendation system Netflix, provides users with the recommendations of the movies that are similar to the ones that have been watched in the past or searched, and also with a collaborative content filtering, provides users with the recommendations in respect with the other users who might have a similar viewing history or preferences.

The recommendation system is classified into two type, content-based and collaborative based recommendation system.

Souvik Debnath and his team proposed a hybridization of collaborative filtering and content based recommendation system assigning weight estimated from a set of linear regression equations obtained from social network graph to attributes used for content based recommendation system.[2] Vimala et al propose fuzzy C-means clustering based on Kullback-Leibler divergence [3] Akter Hossain et al develop a neural engine based-recommendation system applying neural network.[4]

III. Algorithm

In this project we are trying to use three different methods:

Decision Tree –Jiarong Xia
K-means clustering – Yiran Lyu
K-Nearest Neighbor – Fan Luo

A. Data description and processing

This data set describes 5-star rating and free-text tagging activity from Movie-Lens, a movie recommendation service. It contains 20000263 ratings across 27278 movies created by 138493 users between January 09, 1995 and March 31, 2015. This data set provides users' ID and ratings for watched movies in ratings.csv and movies' ID, title and genres in movies.csv.

The data set we used is the Movie-Lens 20M data set, and we choose to use "movies.csv" and "ratings.csv" in our project. For convenience, we merge these two files together. Also for each movie, there are more than one users gave the rating score, so we use the mean rating to represent.

	movieId	title	genres	(rating, size)	(rating, mean)
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	49695.0	3.921240
1	2	Jumanji (1995)	Adventure|Children|Fantasy	22243.0	3.211977
2	3	Grumpier Old Men (1995)	Comedy|Romance	12735.0	3.151040
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2756.0	2.861393
4	5	Father of the Bride Part II (1995)	Comedy	12161.0	3.064592

Fig. 1. Movies' rating means .

B. Decision Tree

- The first machine learning algorithm our team used is the Decision Tree method.A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but

are also a popular tool in machine learning.A decision tree is a flowchart-like structure in which each internal node represents a ”test” on an attribute each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). [5] The paths from root to leaf represent classification rules.In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values of competing alternatives are calculated. So in this movies recommendation problem, we can use the Decision Tree algorithm to get a clearly result.

- Implementation process:
  To build the Decision Tree, first we need to decided how many attributes we will use in this method. According to the “movies.csv” file, I count the number of times each of the genre keyword appear, and using the Word Cloud to show it.

Fig. 2. Word cloud of the movie’s genres .

We could find out that there are 19 genres in totals, and some of the genres have a very small appear size, so I choose the top 9 genres to be the Decision Tree’s attributes. Each genre has a own column to represent that whether this movie has this genre or not (1: the movie has this genre; 0: not has).
The rating size of each movie is also very important in build the recommendation system. If the rating size is small, which means few of the user gives the ratings to this movie, then it won’t has useful value in this problem. To get the useful data, I only use the data that rating size greater than 150.

```
haha1=data[data['rating_size']>=150].sort_values([('rating_mean')],ascending=False)
```

Fig. 3. Rating size greater than 150 .

Then according to the describe of the rating mean information, there are many kinds of them, such as count, mean, std, min, etc (Fig 4).
We need to choose one of them to be the boundary of the rating mean. If the rating mean is greater than or equal to the boundary, the movie recommendation system will automatically recommend this movie to the user. On the other hands, if the rating mean of the movie is less than the boundary, the movie recommendation system will not recommend it.

```
1  data['rating_mean'].describe()
```

```
count    26745.000000
mean         3.133343
std          0.664094
min          0.500000
25%          2.800000
50%          3.235529
75%          3.565217
max          5.000000
Name: rating_mean, dtype: float64
```

Fig. 4. Rating mean info .

For this part, I have tried all of them to be the boundary of whether recommend this movie. Finally it turns out that when the boundary is equal to the mean of the rating mean, the movie recommendation system will get the best accuracy. So I choose boundary = mean of rating mean = 3.13 in my further works.

```
haha1['recommend'] = '0'
for idx, row in haha1.iterrows():
    rating_mean = row['rating_mean']
    if rating_mean >= 3.13:
        haha1.loc[idx, 'recommend'] = '1'
```

Fig. 5. Recommend boundary .

In the following picture, we can see that each genre has a own column to represent, and it also has a “recommend” column. The Fig 5 is the final file I will use for the Decision Tree algorithm. For the Decision Tree algorithm, I split the data set into two parts – training and test. Using the training data to train the model and the testing data to test the tree’s accuracy.

genres	rating_size	rating_mean	drama	comedy	thriller	romance	action	crime	horror	documentary	adventure	recommend
Crime|Drama	63366.0	4.446990	1	0	0	0	0	1	0	0	0	1
Crime|Drama	41355.0	4.364732	1	0	0	0	0	1	0	0	0	1
Crime|Mystery|Thriller	47006.0	4.334372	0	0	1	0	0	1	0	0	0	1
Drama|War	50054.0	4.310175	1	0	0	0	0	0	0	0	0	1
Crime|Drama	27398.0	4.275641	1	0	0	0	0	1	0	0	0	1
Action|Adventure|Drama	11611.0	4.274180	1	0	0	0	1	0	0	0	1	1
Mystery|Thriller	17449.0	4.271334	0	0	1	0	0	0	0	0	0	1
Action|Drama|War	4305.0	4.263182	1	0	0	0	1	0	0	0	0	1
Drama|Romance	24349.0	4.258327	1	0	0	1	0	0	0	0	0	1

Fig. 6. Final File .

To build the Decision Tree for the movie recommendation system, I compute the “Entropy” “Confuse matrix” and “Accuracy”. Then draw the full tree (Fig.7). We can see that the full tree of this problem is pretty complicated. If the movie recommendation system is based on the full tree it will cost lots of times to determine whether a movie will be recommended or not, so we need to prune

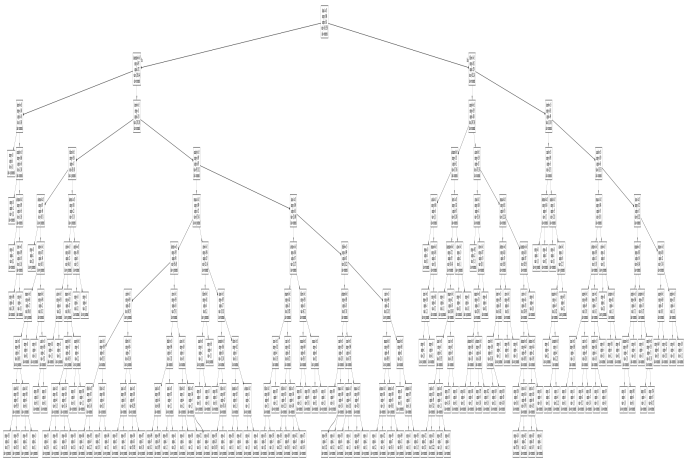

Fig. 7. Full Decision Tree .

the tree.

Then I used the GridSearchCV function to search cv to find the best leaf node number and with the best leaf node to prune the tree. Finally we get the new pruned tree as the figure 8 shows.

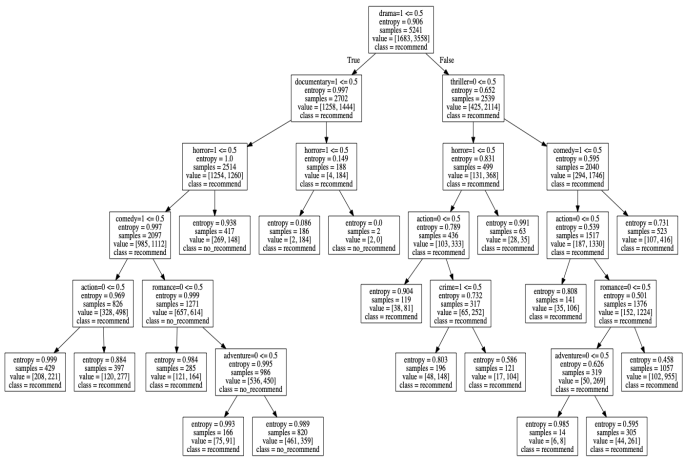

Fig. 8. Final Decision Tree .

How the Decision Tree recommendation system work, we only need to do is input the movie's name and its genres. Then the movie recommendation system will automatically tell recommend or not.

Example:

```
1 movie_name = input('Please input the movie name(or keywords): ')
2 movie_genres = input('Please input the movie genres: ')
```

```
Please input the movie name(or keywords): North by Northwest (1959)
Please input the movie genres: Action|Adventure|Mystery|Romance|Thriller
```

```
117
R
```

Recommend this film !

Fig. 9. Example of Decision Tree .

C. K-Means Clustering

- One of the goals of this project is recommending 10 or more movies for a certain movie, which seems more like a part of search system. A good search system would not only match the correct answer to the system input, such like a movie title, but also give another results related to the input, which part as a small recommendation system may be called "Guess you like it". At that time, system judge the input as what the user are interested in. Here I have to find similar movies with the input one. Certainly there are many classification standards of movies, besides the actor or the director, movie's type may be the most common one. However, each movie may belongs to more than one genre. Then I determine to use K-means clustering, an algorithm of machine learning, to reclassify all the movies and recommend which are in the same new classification with the input one.

  K-means clustering partitions n observations(each observation is a d-dimensional real vector) into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.[6] It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different and far as possible, which makes this algorithm reasonable to realize our project objective.

- Implementation process:

  To apply K-means clustering for reclassifying all the movies, each movie as a observation need to be expressed with a d-dimensional vector. After reading the processed data, I make a new data frame with new added columns indexed by genre name appeared in genres of movies data set. If one movie belongs to one genre, it would display 1 in that cross table, otherwise, it would display 0. Dropping movies whose rating size value is less than 150 and dropping genres whose value of appearing times is less than 1000, then we can get a 16 dimensions vector set for last movie data. (Part of the vector set has been shown as figure.10) For use K-means, it is necessary to change it as a spreadsheet-style pivot table with the title as index.

	title	drama	comedy	thriller	romance	action	crime	horror	documentary	adventure	sci-fi	mystery	fantasy	war	children	musical	animation
0	Shawshank Redemption, The (1994)	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	Godfather, The (1972)	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	Usual Suspects, The (1995)	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0
3	Schindler's List (1993)	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	Godfather: Part II, The (1974)	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fig. 10. Movie genre matrix .

It is critical to specify a great number of clusters k to

get a good model. I try to use elbow method to find such an optimal number. Varing k from 2 to 100 incremented by 5 and compute clustering algorithm for different value of k. For each k, calculate the sum of squared error and plot the curve of SSE according to the number of cluster k(shown as figure.11) According to the elbow method, the location of a bend in the plot is generally considered as an indicator of the appropriate number of clusters. That is to say, after this location, the sum of squared error would not descend and vary very obviously. Here we can think the bend location is on point where number of clusters equals to 20.

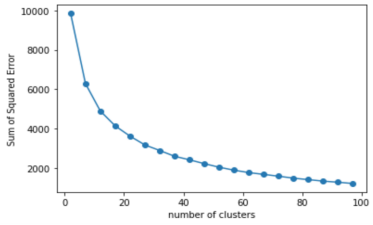

Fig. 11. Curve of SSE according to the number of cluster K .

K-means algorithm initialize centroids by first shuffling the data set and then randomly selecting k data points for the centroids. In each iteration, K-means computes the sum of the squared distance between data points and all centroids, assigns each data point to the closest cluster, then computes the centroids for the clusters by taking the average of the all data point that belong to each cluster. Keep iterating until there is no change to the centroids. Different initialization of centroids may lead to different clusters since K-means algorithm may stuck in a local optimum and may not converge to global optimum. Therefore, using different initialization and pick the result with least sum of squared error is the best choice. K-means method is in the module sklearn.cluster. I assign the number of clusters equals to 20, max iterations equals to 500, number of initialization equals to 10 and pass the pivot table to fit method to learn the clusters on train data. After iterations, it returns an array of integer labels corresponding to the different clusters. For making this fitted model visualized, I use principal component analysis(PCA) to reduce data dimension and draw this figure with label attribute shown as figure.12. Here are 20 clusters with their centroids marked with white cross.

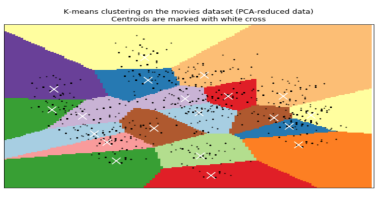

Fig. 12. K-means Clustering on PCA-reduced data .

I create a dictionary to store the label result of each movie according to the returned array after model fitting and save in a new file merged with other data set of other movie information shown as figure.13. Each movie's title, genre, mean rating and cluster label has been shown in this file.

	movieId	title	genres	rating_mean	cluster
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3.921240	18
1	2	Jumanji (1995)	Adventure|Children|Fantasy	3.211977	6
2	3	Grumpier Old Men (1995)	Comedy|Romance	3.151040	15
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2.861393	14
4	5	Father of the Bride Part II (1995)	Comedy	3.064592	2

Fig. 13. Movie data set with cluster label .

Final step is for recommending movies, which can take movie "Avatar" as an example. When the input movie is "Avatar", the system will match certain movie and return the 16 dimension vector of it then use predict method to find which cluster it belongs. After that, finding out other movies with same cluster label and sorting them in descending order of mean ratings. In the end, recommend the top 10 movies of that, which can be shown as figure.14.

	title	genres	rating_mean
0	Seven Samurai (Shichinin no samurai) (1954)	Action|Adventure|Drama	4.274180
1	Raiders of the Lost Ark (Indiana Jones and the...	Action|Adventure	4.219009
2	Yojimbo (1961)	Action|Adventure	4.211717
3	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi	4.190672
4	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi	4.188202
5	Treasure of the Sierra Madre, The (1948)	Action|Adventure|Drama|Western	4.161397
6	Good, the Bad and the Ugly, The (Buono, il bru...	Action|Adventure|Western	4.123241
7	Sanjuro (Tsubaki Sanjûrô) (1962)	Action|Adventure|Drama	4.117039
8	Cowboy Bebop (1998)	Action|Adventure|Animation|Crime|Sci-Fi	4.042373
9	Indiana Jones and the Last Crusade (1989)	Action|Adventure	4.007593

Fig. 14. Recommendation of movie Avatar .

D. K-Nearest Neighbor

- K-Nearest Neighbor is classified by measuring the distance between different characteristic values. [7] The idea is: if most of the samples of a sample in the feature space of k most similar samples (that is, the samples closest to each other in the feature space) belong to a certain category, then the sample also belongs to this category. When the data and labels in the training set are known, input the test data, compare the characteristics of the test data with those of the training set, and find the first K data that are most similar to them. Then the category corresponding to the test data is the one that appears most frequently in the K data. In the movie recommendation system, if we want to recommend a current movie's similar movie, we need to compare the distance between current movie vector and other movie vectors vector space to find the closest movies.

- Implementation process:
  First, our data is from Movie-Lens data set. The user review data contains three tables, movies, ratings and links. We will only use two of them, movies and ratings. Since score data is more, we will feed all the data to our recommendation algorithm, it will lead to memory leaks, there will be a ”memory error”, so, we just focus public attention on high film, is also more than the number of evaluation and grading is high also, because the fewer evaluation and evaluation of the film is not necessary to recommend.
  To find out which movies have the highest public interest, we need to integrate the movies table and the rating file. We found that there are a large number of records with empty movie titles, so we should first filter out those records without movie titles. Then we are going to count the total number of reviews for each movie. With the total number of reviews per movie, we can figure out which movies are the most watched and which are the most popular.
  First, let's look at the distribution of the total number of reviews:

```
In [23]: pd.set_option('display.float_format', lambda x: '%.3f' % x)
         print(movie_rating_count['totalRatingCount'].describe())

count   26744.000
mean      747.841
std      3085.818
min         1.000
25%         3.000
50%        18.000
75%       205.000
max     67310.000
Name: totalRatingCount, dtype: float64
```

Fig. 15. Distribution of the total numbers .

We can see that the total number of movies is 26744, and 50 percent of them have less than 18 reviews, which means that 50percent of them have more than 18 ratings. However, if a movie has only about 18 times of evaluation, it is not a movie that is worthy of attention. Therefore, it should not be recommended, let alone ”fed” to the recommendation algorithm for calculation. Because the recommended algorithm will consume a large amount of system resources during the execution, the worthless data should not be involved in the operation, otherwise it will cause system memory overflow and ”memory error” problem. Next, let's look at the top 10 percent of the percentile table:

```
In [24]: print(movie_rating_count['totalRatingCount'].quantile(np.arange(.9,1,.01)))

0.900    1305.700
0.910    1543.000
0.920    1848.000
0.930    2285.980
0.940    2847.000
0.950    3612.950
0.960    4700.560
0.970    6219.970
0.980    8835.780
0.990   14388.690
Name: totalRatingCount, dtype: float64
```

Fig. 16. Top 10 percent of the percentile table .

We can see that 90 percent of the movies were rated less than 1305 times, which means that 10 percent of the movies were rated more than 1305 times. And 9 percent of the movies had more than 1543 reviews, 8 percent had more than 1848 reviews, and 7 percent had more than 2285 reviews. I think if a movie gets more than 1305 reviews, it should be a movie that gets a lot of attention. Let's take 1305 times as an indicator of popular movies. (you may have different ideas, but you can try other indicators.) the total number of movies is 26744.
To implement K-Nearest Neighbor method, we now construct a matrix of user ratings of movies, where each row represents a movie, each column represents a user, and each value in the matrix represents a user's rating of a movie. If the user doesn't comment on a movie, set it to 0. We then convert the value (rating) of the matrix data frame to a sparse matrix so that more efficient calculations can be made. Then we use the sklearn.neighbors algorithm. Specify parameters (metric='cosine', algorithm='brute') to compute the cosine similarity between the rating vectors. Finally, fit our model. In this step, the K-Nearest Neighbor algorithm starts to calculate the distance between the movie and all the other movies, and finds the K movies closest to the current movie.
Example:

```
movieId=29
distances, indices = model_nn_binary.kneighbors(ratings_pivot.query('movieId == 29').values, n_neighbors = 11)

for i in range(0, len(distances.flatten())):
    likelymovieId=ratings_pivot.index[indices.flatten()[i]]
    if i == 0:
        print('Current movie:',movies[movies.movieId==movieId]['title'].values[0])
    else:
        print('Recommand movie{0}: {1}, Distance is:{2}'.format(i, movies[movies.movieId==likelymovieId]['title'].value
                                                                distances.flatten()[i]))

Current movie: City of Lost Children, The (Cité des enfants perdus, La) (1995)
Recommand movie1: Delicatessen (1991), Distance is:0.5609612915294158
Recommand movie2: Brazil (1985), Distance is:0.6153414366133902
Recommand movie3: Blade Runner (1982), Distance is:0.661584742255307
Recommand movie4: Dark City (1998), Distance is:0.6697961262326608
Recommand movie5: Twelve Monkeys (a.k.a. 12 Monkeys) (1995), Distance is:0.6713548084356827
Recommand movie6: Clockwork Orange, A (1971), Distance is:0.6762167519110898
Recommand movie7: Trainspotting (1996), Distance is:0.6864511942296707
Recommand movie8: Pi (1998), Distance is:0.6898976955380305
Recommand movie9: Ghost in the Shell (Kôkaku kidôtai) (1995), Distance is:0.7006300194646904
Recommand movie10: 2001: A Space Odyssey (1968), Distance is:0.703187155348789
```

Fig. 17. Example of K-Nearest Neighbor .

IV. Comparison

- Decision Tree:
  The advantages of Decision Tree method are that this algorithm has a high accuracy for the recommend predict. The decision tree can show the prediction process and the prediction result more directly.
  The disadvantage is that the accuracy of the not recommend predict is a little bit low. Also, it could only predict one movie at a time, so the efficiency is low.

```
The in-sample recommend accuracy is:  85.7504215851602 %
The in-sample no_recommend accuracy is:  43.49376114081996 %
The out-sample recommend accuracy is:  84.9673202614379 %
The out-sample no_recommend accuracy is:  43.793584379358435 %
```

Fig. 18. Accuracy of the Decision Tree .

- K-Means Clustering:
  K-means clustering is considered as an unsupervised learning method. In this project, we cannot process a testing data with true classification of each movie to compute the output accuracy of this algorithm. Generally, we can evaluate the clustering quality with silhouette score that displays a measure of how close each point in one cluster is to points in the neighboring clusters. This measure has a range of [-1,1], and the clustering quality would be better when the measure value is closer to +1. In this project, when the number of cluster is 20, the silhouette score is 0.44145, which seems not a very good result. Out of curiosity, I output silhouette score for k value from 2 to 1000 incremented by 50. The score ascend until the number is more than 602 measuring as 0.96925. However, when the number of cluster is too high, the classification would be too specific and the recommendation would be too simple to make a comprehensive recommendation. Although it is difficult to predict the number of clusters and initial seeds have a strong impact on the final results, K-Means clustering is still a great algorithm for easily implementing, fast computing and no requirement for ground truth, and it is measurable and efficient in large data collection.
- K-Nearest-Neighbor:
  Pros:1.The training time complexity is lower than algorithms such as support vector machines.2.Compared to algorithms like naive Bayes, which have no assumptions about the data, are more accurate, and are less sensitive to outliers.3.Because KNN method mainly relies on the limited neighboring samples around, rather than the method of class domain discrimination to determine the category, KNN method is more suitable than other methods for the sample set to be classified with more overlapping or overlapping class domains.
  Cons:1.It's a lot of work, especially if you have a lot of features.2.Compared with the decision tree model, KNN model is not strong in interpretability.3.When the samples are unbalanced, the prediction accuracy of rare categories is low Similarity: 66 percent.

V. Future Research Direction

There are two directions to improve our recommendation system. First, we can adopt more attributes of each movie such like the movie tags, actors, directors and so on. Since people's preference would be influenced by multiple factors, including more attributes will let the recommendation system consider more possibilities and information. After this improvement, decision tree algorithm can get higher accuracy, K-means clustering can use observations with more dimensions to fit a better model and K-Nearest Neighbor algorithm can have more information to compute the distance of two movies.
Second, we can try to change the input of recommendation system, that is to say, recommend movies to a certain user but not a certain movie. For the decision tree algorithm, we can use one user's history data as the data set to determine whether a movie should be recommended to such a user. For K-means clustering, the system can be improved to make multiple movies with higher ratings rated by one user as a mixed input and return the recommendation combining and sorting results from each signal high-rating movie input to this user. For K-nearest-neighbor algorithm, the system can also recognize movies that have have high rating from one user and find out the nearest movies to make a hybrid recommendation.

VI. Conclusion

In this final project our team build a movies recommendation system based on the Movie-Lens 20M data set. Jiarong uses the Decision Tree algorithm build a decision tree, determine whether a movie should be recommended based on the final new tree and get a good accuracy. Yiran uses the K-means Clustering to divide movies into different clusters by genre information and recommend 10 or more movies sorted by ratings in same cluster. Fan's work is based on the K-Nearest Neighbor algorithm, computing the distance between movies to find the closest movies and recommend the top 10 or more similar movies that have the nearest distance. Because each recommendation algorithm has advantages and disadvantages, in order to get a better recommendation system, we need to combine the advantages of the three algorithms to get a uniformly recommendation.

VII. References

[1] Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[2] Debnath, S., Ganguly, N. Mitra, P. (2008). Feature weighting in content based recommendation system using social network analysis. Proceedings of the 17th International Conference on World Wide Web, 2008: 1041–1042

[3] Vimala S V, Vivekanandan K. A Kullback–Leibler divergence-based fuzzy C-means clustering for enhancing the potential of an movie recommendation system[J]. SN Applied Sciences, 2019, 1(7): 698.

[4] Hossain M A, Uddin M N. A Neural Engine for Movie Recommendation System[C]//2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT). IEEE, 2018: 443-448.

[5] Wikipedia Decision Tree:
https://en.wikipedia.org/wiki/Decision- tree

[6] Wikipedia K-Means:
https://en.wikipedia.org/wiki/K-means-clustering

[7] Wikipedia K-Nearest-Neighbor:
https://en.wikipedia.org/wiki/K-nearest-neighbors-algorithm