# Exploring Movie Recommender System Using Machine Learning Techniques

Xiaoan Xu xiaoanxu413@gmail.com
Chenyuan Shen shency18@mails.tsinghua.edu.cn
Jinglei Chen chenjingl000817@gmail.com
Yixuan Liu 980117853@qq.com
Caiyue Chen caiyue_chen@163.com
Zhiyuan Wei 18961867913@163.con

## 1 Abstract

With the rapid exploration of technological applications, online information have caused confusion among customers. Hence, it seems that developing recommender systems has become increasingly inevitable. Though these recommending systems can predict users' preference, boosting the decision making skills of users as well as assisting corporations in a higher profit status, few truly conduct accurate recommendation to users. Focused on the problems of prediction inaccuracy which occurs in many websites, this paper will study the field of movie websites, firstly systematically introducing the main machine learning techniques which are often used in the movie recommendation systems. Then this paper will figure out the similarities and differences of the prediction results by comparing 3 machine learning techniques. The evaluation results finally indicate that the prediction accuracy in this movie recommender system has been efficiently enhanced by Latent Factor Model.

**Keywords: Movie recommender system; K-Means; Latent Factor Model; Cuckoo search algorithm**

## 2 Introduction

With the progress of technology and the rapid development of the Internet, massive amounts of information are flooding our lives, making users drowning in many noisy information. In order to improve this problem, the recommender systems consequently arise from these ever-expanding of considerable online information. The aim of recommender system is to actively perceive the needing of users, transferring from the passive way in absorbing information (Sun et al., 2013). However, while the widely application of recommender system the obstacles of data sparsity and low prediction accuracy often occur in many recommendation websites.

So far there have been many approaches showing great potential in improving the prediction accuracy of recommender systems. For example, collaborative filtering optimization has attracted great attention in many fields of recommend system, depending on the comments and rankings from customers. Moreover, according to Tahmasebi (2020), recent advances based on deep learning like natural language processing have also showed great performance especially on retailing industries.

Hence, this report will mainly focus on exploring the movie recommender systems by comparing of 3 machine learning techniques including K-means, latent factor models (LFM) and cuckoo search by adopting clustering analysis.

# 3 Literature Review

In this context, how to obtain valuable information efficiently and quickly has become a very important thing for us. As an efficient information filtering tool, recommendation system has been applied more and more widely in our life. In this paper, we mainly study the movie recommendation system and we have reviewed some papers in the field of it.

## 3.1 Recommender System

Many techniques can be applied to the recommender system, such as Collaborative Filtering, Content-Based Filtering , Item-Based Filtering, Hybrid Filtering, Clustering and so on.

### 3.1.1 Collaborative Filtering Method

Collaborative filtering is the technique to filter or calculate the items through the sentiments of other users.

The concept of collaborative filtering (CF) was first introduced by Goldberg et al. (1991), where systems enable customers to acquire messages about items with higher similarity with their past records.

Using Collaborative Filtering to Weave an Information Tapestry (1992) was among the earliest essays to introduce collaborative filtering into recommender system.It innovatively uses the behavior data of other users to make recommendation for certain user.

However, the integration of private message and public information may pose security threat.To further generalize collaborative filtering model, model-based collaborative filtering like latent factor models is proposed [Latent Semantic Models for Collaborative Filtering (2004)].It splits the interaction matrix into two matrix and both can be approximated using gradient descending algorithm.

In terms of model based CF, It appeared that constructing a factorized matrix becomes one of the most significant application in exploring user ratings and their preference to the movies (Takacs et al. ,2007).

Then about memory based CF, graph databases were firstly implemented by Yi et al.(2017), who found out that the rating system of movies can be impacted by the increasing radius of nodes and thicken edges.

### 3.1.2 Content-based Method and Item-based Method

Unlike CF method, in Content Based (CB) recommender system, there are two main ways for the system to capture data, either by observing customers rating or their clicking rate (Li et al., 2012).

However, problems of low accuracy, data sparsity and cold start often occurred among movie websites (Sun et al., 2013).Consequently, in order to improve the original CB method, Son and Kim (2017) built a network with more attributes in movie lens datasets, showing that problems like sparsity in movie datasets could be dealt with in this multi-attribute CB methods.

Item-Based Collaborative Filtering Recommendation Algorithms (2001) reviews the theory of user-based collaborative filtering and compare the result of that with item-based collaborative filtering.It turned out that item-based collaborative filtering algorithm has better accuracy.

Besides, item-based collaborative filtering is more suitable for large scale data sets due to the comparatively static item-based similarity.

### 3.1.3 Hybrid Method

However, both CF and CB methods have limited application, so to enhance filtering, hybrid filtering was proposed by Jain et al. (2018), which was a combination of both these two techniques.For instance, an experiment about exploring customers preference and similarity about the movie was conducted by Bharti and Gupta in 2019, where new users were applied in CB method, while CF method was for old users.

Since then, hive was generated and used in storing movie datasets.

### 3.1.4 Clustering

Clustering is also introduced to find the similarity of users in the recommender systems.And one of the most famous clustering algorithms is K-means clustering.

The aim of the K-means algorithm is to divide M points in N dimensions into K clusters so that the within-cluster sum of squares is minimized.The K-means clustering algorithm is described in detail by Hartigan (1975).

Aristidis Likasa, Nikos Vlassis, JakobJ. Verbeek(2002) presented the global k-means algorithm which was an incremental approach to clustering that dynamically added one cluster center at a time through a deterministic global search procedure consisting of N (with N being the size of the data set) executions of the k-means algorithm from suitable initial positions.They also proposed modi2cations of the method to reduce the computational load without signi2cantly a3ecting solution quality.The proposed clustering methods were tested on well-known data sets and they compared favorably to the k-means algorithm with random restarts.

J. A. HARTIGAN and M. A. WONG(2012) presented an efficient version of the K-means clustering algorithm.They sought instead "local" optima, solutions such that no movement of a point from one cluster to another would reduce the within-cluster sum of squares.

## 3.2 Research on Movie Recommender System

In Recommending and Evaluation Choices in a Virtual Community of Use (1995), the video recommender system is based on users rating rather than their information.

Panagiotis Symeonidis, Alexandros Nanopoulos and Yannis Manolopoulos2009designed a movie recommendation system with explanations called MoviExplain. It combines Collaborative Filtering with Content-Based filtering that provides both accurate and justifiable recommendations. Its Recommenation Engine mainly uses Feature Similarity an Ratings Similarity to construct the MovieExplain algorithim. It attains both accurate and justifiable recommendations, giving the ability to a user, to check the reasoning behind a recommendation.

Katarya and Verma (2016) combined k-means clustering and cuckoo search optimization algorithm, applied it to Movielens datasets and got better testing results compared with other approaches.To overcome the limitations of typical collaborative recommender system, k-means algorithm is applied for clustering and cuckoo search is implemented for optimization.The approach discussed provided high performance regarding various metrics ( e.g. MAE and SD) and accuracy. But one limitation would be that the approach suffer from cold start problem.

A fully content-based movie recommender system is proposed by HUNG-WEI CHEN(2017). The Word2Vec CBOW model is used to extract features from the content of movies and transform the textual content data into feature vectors, and then the average cosine similarity can be calculated to measure the similarity f movies. Content-based methods are crucial when it comes to cold-start problem, but combination of collaborative filtering and content-based method into hybrid model is a promising way to improve the performance.

A hybrid approach using collaborative filtering and content-based filtering for recommender system has been presented by Geetha G(2018), where the k-means algorithm is used, as well as the Pearson correlation Score. Pearson correlation score tends to be more accurate for determine the similarity between peoples interest when the data isnt well normalized, while its more complicated. Also it shows that the recommendation accuracy is usually higher in hybrid systems, as the combination of both leads to common knowledge increase.

Inan et al. (2018) used Moreopt, a movie recommender system, to select top N movies to users.Firstly, a sparse dataset, UMR was used to get data.

Then feature weights were calculated with a mathematical tool. After that, item-based Pearson Correlation, together with feature weights, was used to do missing rating prediction so that UMR could be updated. Then Inan et al. used user-based Pearson Correlation to select top movies to users.

Bhalse and Thakur (2021) suggested recommendation lists through singular value decomposition and collaborative filtering, which were used to deal with sparsity problem and cosine similarity, which was used to select movies.

Joorabloo et al. (2021) considered increasing diversity in recommender system to improve users quality of experience and found that gender and age affected the balance between precision and diversity.

Then they proposed a probabilistic graph-based recommender system (PGBRS) to recommend items.

# 4 Exploratory Data Analysis

This report is based on The Movies Dataset collected from Kaggle, which containing over 45000 movies released on or before July 2017 as well as 26 million ratings from over 270000 users. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

The most important files that we are going to focus on are ratings small.csv(The subset of 100,000 ratings from 700 users on 9,000 movies) and movies meta- data.csv(The main Movies Metadata file). The ratings that users gave to movies range from 0 to 5, with mean 3.5436 and variance 1.1195. As shown in figure1, most of the movies are rated with 4, and the lowest and highest rating are 0.5 and 5 respectively:

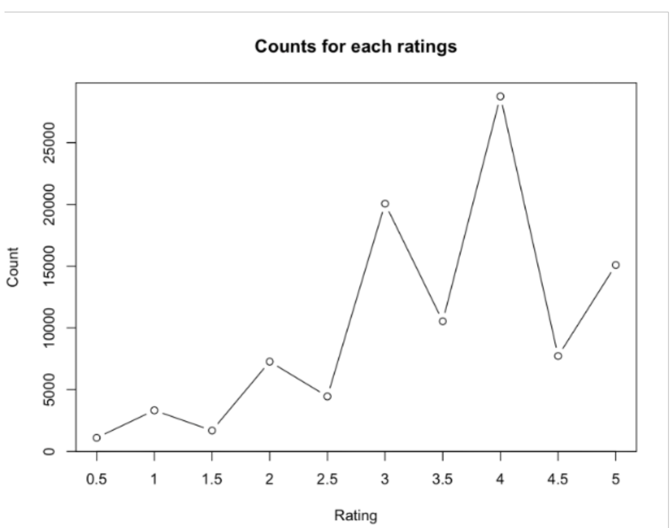



Figure 1: Counts for each ratings

There are twenty different genres in the movie dataset, which are Drama, Comedy, Romance, Music etc. The five most popular genres are Drama, Comedy, Thriller, Action and Romance, while only a small portion of movies are related to genres like TV movies, Foreign and War.

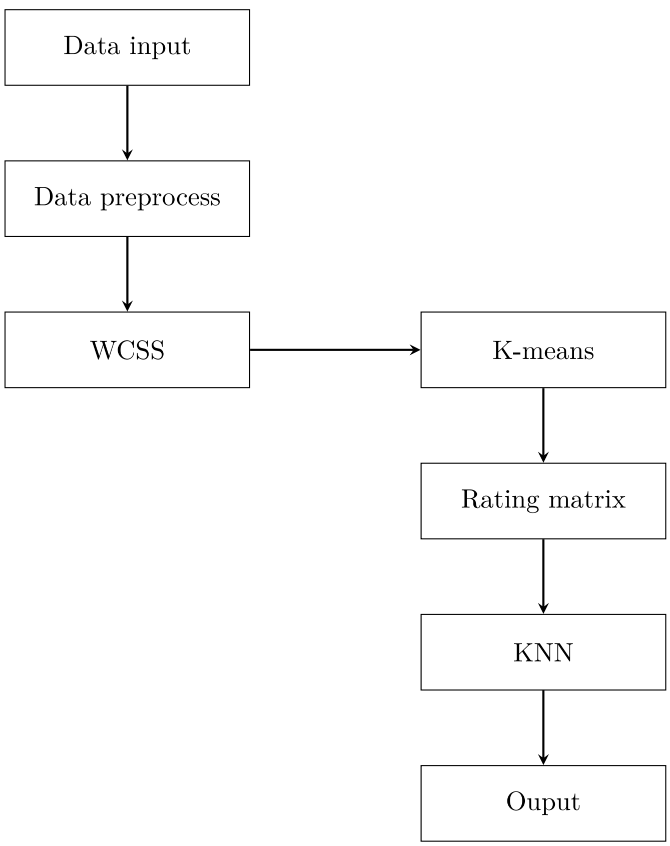
# 5 K-means KNN Model

## 5.1 The Adoption of K-means

One of the important ideas in some recommender systems is that the users are equal, there is no approach to claim that one user should be more qualified than one another when it comes to recommending the movie. However, a far more progressive solution is the collaborative system. Collaborative ltering is the technique to lter or calculate the items through the sentiments of other users. An effective technique is clustering.

As the most famous clustering algorithm, we can use K-means to build a recommender system. The system can be divided into three modules, the input module, the processing module and the output module. The inputs include the user information, and the rating the users gives to the movies. The processing module includes the preprocessing of the dataset, the building of a utility matrix which shows which user rated which movie, the application of K-means algorithm, the calculation of correlation between the uses and the KNN system. The output module describes the movies the input user may like.

The flow chart of this model is as follows:




## 5.2 The Specified Steps of the Algorithm

1. Read the csv file and build a data frame to represent the rating the users gives to the movie. Fill in the blank with 0.

2. Use the K-means algorithm and calculate the within cluster sum of square using different number of clusters. Chose the proper number of clusters to go on.

$$WCSS = \frac{\sum |x_i - \hat{x}_l|^2}{n}$$

$\hat{x_l}$ is the cluster center of $x_i$.

3. Use K-means to cluster the data.

4. Build a matrix to represent the rating one gives to each cluster.

5. Use KNN to find the users similar with the target user. Here we make a little alteration on KNN algorithm. We choose k nearest neighbor who have already watched the target movie to do the prediction. If there are no enough people who have watched the movie, then we use the average rating on the movie to predict. And if there is no any user who has watched the movie, we use the average rating on all movies.

6. Make recommendation according to these users preference.

7. Use the RMSE on test set to see whether the recommendation is correct.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y_l} - y_i)^2}{n}}$$

## 5.3 The Implementation and Result of the System

The plot showing how the WCSS varied with the number of clusters is as follows:

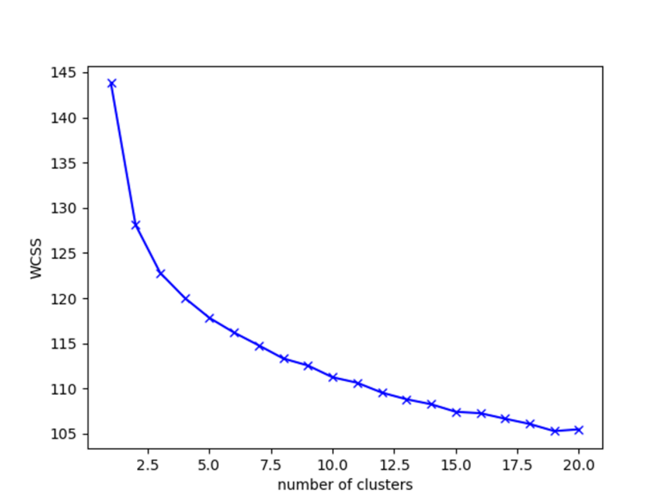
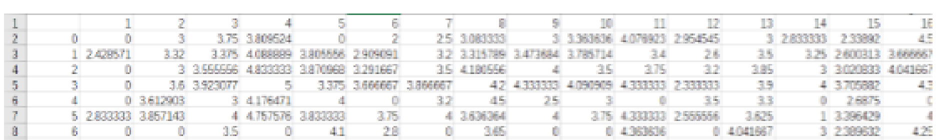



Figure 2: Variation of WCSS with the number of clusters

We choose k=8 as the number of clusters.

The matrix to represent each users rating on each cluster is as follow:



| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 2 | 0 | 0 | 3 | 3.75 | 3.809524 | 0 | 2 | 2.5 | 3.083333 | 3 | 3.363636 | 4.076923 | 2.954545 | 3 | 2.833333 | 2.33892 | 4.5 |
| 3 | 1 | 2.428571 | 3.32 | 3.375 | 4.088889 | 3.805556 | 2.909091 | 3.2 | 3.315789 | 3.473684 | 3.785714 | 3.4 | 2.6 | 3.5 | 3.25 | 2.600313 | 3.666667 |
| 4 | 2 | 0 | 3 | 3.555556 | 4.833333 | 3.870968 | 3.291667 | 3.5 | 4.180556 | 4 | 3.5 | 3.75 | 3.2 | 3.85 | 3 | 3.020833 | 4.041667 |
| 5 | 3 | 0 | 3.6 | 3.923077 | 5 | 3.375 | 3.666667 | 3.866667 | 4.2 | 4.333333 | 4.090909 | 4.333333 | 2.333333 | 3.9 | 4 | 3.705882 | 4.3 |
| 6 | 4 | 0 | 3.612903 | 3 | 4.176471 | 4 | 0 | 3.2 | 4.5 | 2.5 | 3 | 0 | 3.5 | 3.3 | 0 | 2.6875 | 0 |
| 7 | 5 | 2.833333 | 3.857143 | 4 | 4.757576 | 3.833333 | 3.75 | 4 | 3.636364 | 4 | 3.75 | 4.333333 | 2.555556 | 3.625 | 1 | 3.396429 | 4 |
| 8 | 6 | 0 | 0 | 3.5 | 0 | 4.1 | 2.8 | 0 | 3.65 | 0 | 0 | 4.363636 | 0 | 4.041667 | 3 | 2.389632 | 4.25 |
| 9 | 7 | 0 | 4 | 0 | 5 | 0 | 0 | 0 | 3.125 | 4 | 0 | 3.5 | 2.2 | 0 | 0 | 2.778325 | 4 |

Figure 3: Each user's rating on each cluster

And we calculated the Pearson correlated matrix according to the matrix above.

Then we choose 5 nearest neighbor to do the prediction.

The eventually RMSE on the test set is 1.004.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.268209 | 0.269762 | 0.181554 | 0.32616 | 0.427671 | 0.414029 | -0.25743 | 0.257744 | 0.393233 | 0.183147 | 0.09081 | 0.193659 | -0.03813 | 0.20956 | 0.104865 |
| 2 | 0.268209 | 1 | -0.25713 | 0.963732 | -0.2765 | -0.21553 | 0.528537 | 0.038295 | 0.898147 | 0.505091 | -0.27102 | 0.812668 | -0.40193 | -0.39372 | 0.485969 | -0.22412 |
| 3 | 0.269762 | -0.25713 | 1 | -0.19202 | 0.575577 | 0.747044 | 0.656729 | 0.358268 | -0.13245 | 0.676116 | 0.195672 | 0.041808 | 0.950798 | 0.618337 | 0.209321 | 0.083225 |
| 4 | 0.181554 | 0.963732 | -0.19202 | 1 | -0.23544 | -0.07349 | 0.582055 | 0.113902 | 0.967197 | 0.558496 | -0.14956 | 0.821453 | -0.32195 | -0.23532 | 0.57682 | -0.09517 |
| 5 | 0.32616 | -0.2765 | 0.575577 | -0.23544 | 1 | 0.479302 | 0.426156 | 0.687285 | -0.22868 | 0.302829 | -0.17436 | -0.12126 | 0.764752 | 0.23189 | 0.301836 | -0.31246 |
| 6 | 0.427671 | -0.21553 | 0.747044 | -0.07349 | 0.479302 | 1 | 0.480526 | 0.083807 | 0.125787 | 0.491378 | 0.899741 | -0.20909 | 0.712259 | 0.748809 | 0.474983 | 0.600102 |
| 7 | 0.414029 | 0.528537 | 0.656729 | 0.582055 | 0.426156 | 0.480526 | 1 | 0.480533 | 0.573235 | 0.973267 | -0.12259 | 0.683506 | 0.560816 | 0.233944 | 0.581731 | -0.18433 |
| 8 | -0.25743 | 0.038295 | 0.358268 | 0.113902 | 0.687285 | 0.083807 | 0.480533 | 1 | 0.011681 | 0.338041 | -0.48364 | 0.258538 | 0.514015 | 0.008471 | 0.456073 | -0.57996 |
| 9 | 0.257744 | 0.898147 | -0.13245 | 0.967197 | -0.22868 | 0.125787 | 0.573235 | 0.011681 | 1 | 0.568661 | 0.093422 | 0.896111 | -0.27258 | -0.05925 | 0.65135 | 0.141244 |
| 10 | 0.393233 | 0.505091 | 0.676116 | 0.558496 | 0.302829 | 0.491378 | 0.973267 | 0.338041 | 0.568661 | 1 | -0.05104 | 0.683259 | 0.547072 | 0.35476 | 0.47043 | -0.08604 |
| 11 | 0.183147 | -0.27102 | 0.195672 | -0.14956 | -0.17436 | 0.899741 | -0.12259 | -0.48364 | 0.093422 | -0.05104 | 1 | -0.49072 | 0.098431 | 0.586999 | 0.227688 | 0.980585 |
| 12 | 0.09081 | 0.812668 | 0.041808 | 0.821453 | -0.12126 | -0.20909 | 0.683506 | 0.258538 | 0.896111 | 0.683259 | -0.49072 | 1 | -0.07841 | -0.25572 | 0.213098 | -0.42719 |
| 13 | 0.193659 | -0.40193 | 0.950798 | -0.32195 | 0.764752 | 0.712259 | 0.560816 | 0.514015 | -0.27258 | 0.547072 | 0.098431 | -0.07841 | 1 | 0.618799 | 0.180133 | -0.02264 |
| 14 | -0.03813 | -0.39372 | 0.618337 | -0.23532 | 0.23189 | 0.748809 | 0.233944 | 0.008471 | -0.05925 | 0.35476 | 0.586999 | -0.25572 | 0.618799 | 1 | 0.111906 | 0.578543 |
| 15 | 0.20956 | 0.485969 | 0.209321 | 0.57682 | 0.301836 | 0.474983 | 0.581731 | 0.456073 | 0.65135 | 0.47043 | 0.227688 | 0.213098 | 0.180133 | 0.111906 | 1 | 0.135629 |
| 16 | 0.104865 | -0.22412 | 0.083225 | -0.09517 | -0.31246 | 0.600102 | -0.18433 | -0.57996 | 0.141244 | -0.08604 | 0.980585 | -0.42719 | -0.02264 | 0.578543 | 0.135629 | 1 |

Figure 4: Caption

# 6 K-means-cuckoo Based Collaborative Filtering Model

## 6.1 Introduction

This model uses k-means clustering to construct collaborative filtering and cuckoo search algorithm to optimize the method. There is a drawback in k-means that a record is classified solely by its absolute distance from the centers without considering the dispersity of records in a cluster. A record is easily classified into a nearby small compact cluster if the record is at the edge of a large loose cluster. That is, relative distance shall be considered here. Because of this, cuckoo search algorithm is introduced to solve the problem and the basic thought of implementation is as follows:

1. Randomly select some users Ua to conduct k-means clustering.

2. Randomly select some users Ub from the remaining pool.

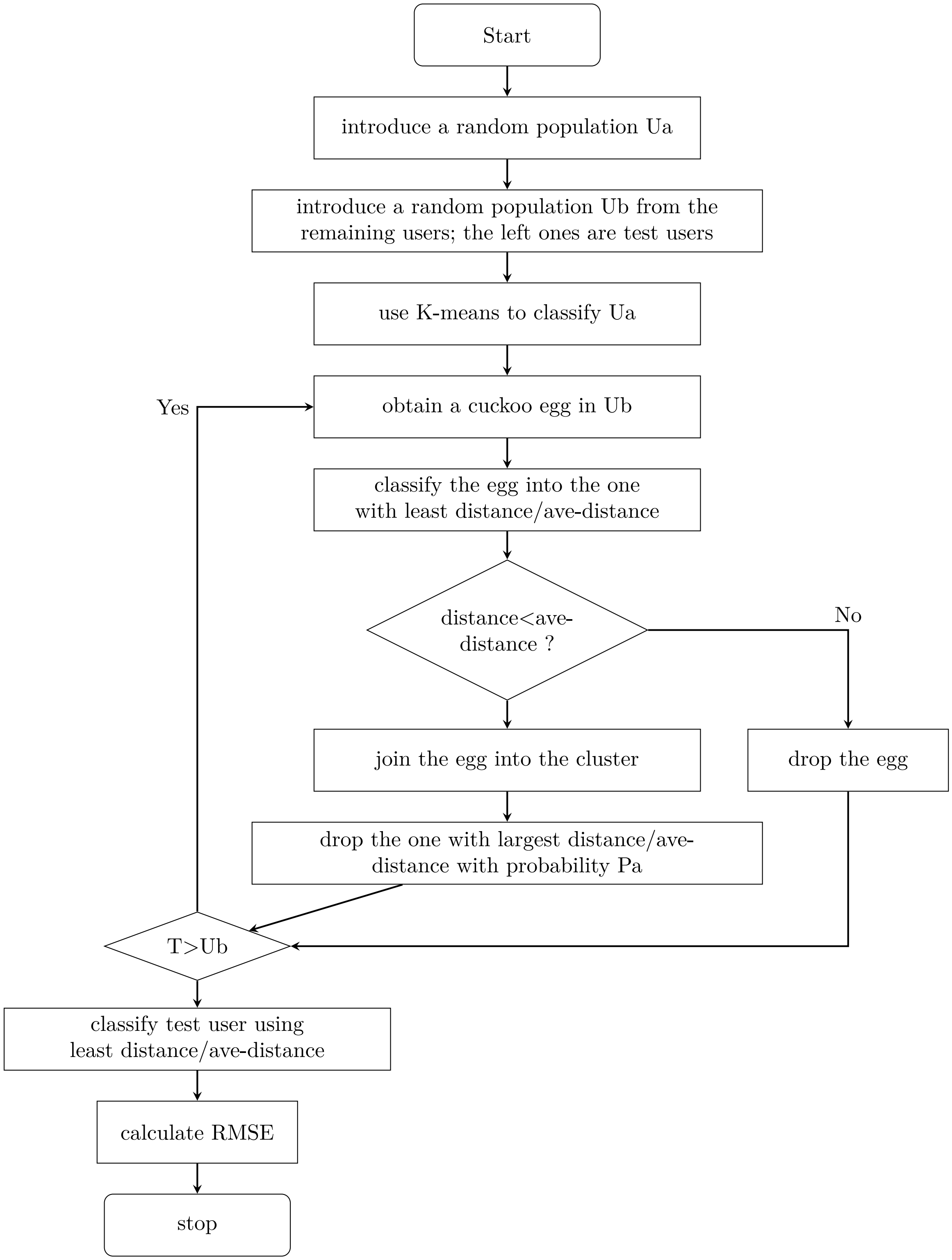
3. Classify the users Ub into the cluster with least (distance / average distance of points in the cluster).

4. For each user in Ub, if the distance meet the requirement, then the user with the largest $\frac{distance}{ave_{distance}}$ will be dropped with probability Pa; if the requirement is not fulfilled, then the user will be dropped.

5. Recalculate the centers and predict ratings.

## 6.2 Analysis

Cuckoo search algorithm is developed from the natural habits of cuckoos. Cuckoos lay eggs in other birds nests. If the hosts fail to identify cuckoo eggs, then the newly born cuckoos will be fed and grow up. Sometimes the eggs are identified by host birds and dropped. In this case, the users are eggs and the clusters are nests. The whole process is shown below:




The process starts from splitting the population into training set and test set where the training set splitting into one for k-means (Ua) and one for optimization (Ub). Then k-means is used to classified Ua. Each user in Ub is first classified into the cluster with the least $distance/ave_{distance}$ where distance is the Euclidean distance between the user and the cluster center while $ave_{d}istance$ is the mean distance of the Ua users in the cluster to the cluster center. After that, the original user with the largest $\frac{distance}{ave_{distance}}$ in the cluster will be dropped with probability Pa if distance of the new user is less than $ave_{distance}$, otherwise the new user will be dropped. After the training set has been classified, the users in the test set are classified to the cluster with the least $\frac{distance}{ave_{distance}}$, after which RMSE is calculated to compare the effects of the model with previous ones.

Cuckoo search algorithm is used here to optimize the results of k-means clustering. It can drop the edge points in the clusters to lower the influence of outlays in training set. In figure5, the red point in the middle of the picture should be classified into the right cluster. However, the point is closer to the left center in absolute distance so it is classified into the left one. With cuckoo search, the $\frac{distance}{ave_{distance}}$ of the middle red point is fairly large and is therefore easily dropped in this model.

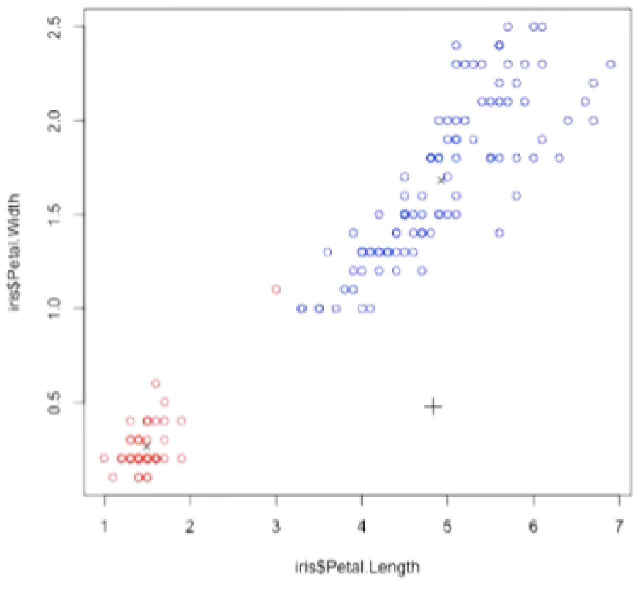



Figure 5: An example of k means cluster

## 6.3 Result

In this model, 60% of all users are randomly selected to comprise Ua and 20% in Ub, with the rest in test set. Number of groups of k-means is 4 and Pa is set as 0.5. The whole process is iterated 10 times to get a mean result. The model uses the same dataset as the above one with 671 users and 9066 movies.

RMSE for this model is 1.05, which is not stable because of the randomness in this model. The result is not so satisfactory mainly because k-means is not suitable for high dimensional data (we have 9066 features but only 671 records). The result shows that most users are grouped into one cluster and few in other clusters, implying that it is not a good clustering. There is also the feature that most users in Ub are classified into a minor cluster instead of the major one, which indicates there are also some problems with cuckoo search in this case. This failure can be due to the logical link between k-means and cuckoo search. With the failure of the former, the latter would also lose its effectiveness.

A final but minor reason for the unsatisfactory result is that test users are classified by $\frac{distance}{ave_{distance}}$ instead of absolute distance (which is used by k-means), which in turn increases RMSE calculated using absolute distance.

To deal with high dimensional problem, PCA, a method of dimensionality reduction, is applied to this dataset. We shall only retain the first six dimensions because with the number of dimensions increasing, effect of k-means clustering drops rapidly as most users are classified into a single cluster. As a result, only 30% of total information is preserved and the result remains unsatisfying.

After all these trials and analysis, it can be inferred that the model should work well on low dimensional data but poorly for this case so a new model is needed to address the problem.

# 7 Latent Factor Model (LFM)

## 7.1 LFM Introduction and Notation

Based collaborative approach can also be used to predict the ratings for movies that are given by users. Model based collaborative filtering only rely on user-item interactions information and the latent factor model (LFM) is one of the most commonly used model to understand these interactions. The basic principle of LFM is to decompose the sparse user-item interaction matrix into a product of two smaller and dense matrices: a user-factor matrix and a factor item matrix. Our goal is to minimize the error between the dot product of the two matrices and the original user-item matrix. We select square loss function with regularization as the loss function. Since the loss function contains two independent variables, we apply gradient descending algorithm to iterate these two matrix.

However, the model that we are gonna introduce in this paper is a bit different from the traditional Latent Factor Model. As people usually prefer a certain type of movie, for example, someone might like comedy more than others, therefore, if the movie is comedy, he/she would be more likely to like it. Therefore, we decided to cluster the movies into difffferent groups according to their genres, other than the original interaction matrix, LFM is also applied to the matrices for each genre after clustering. Finally, a weighted average rating would be calculated for each user-item pair.

The following are some notations that we are going to use next.

- R: represents the original user-item interaction matrix where each row represents a user, each column represents a movie, of size (m Œ n).
- k: represents the number of latent factors that we are going to use.
- U: represents the matrix of preference of the users to factors, of size (m Œ k).
- ui : represents the i-th row of user matrix U.
- M: represents the matrix of items belonging to the factors, of size (nŒk).
- mi : represents the i-th row of item matrix M.

## 7.2 LFM Methodology

**A. Matrix Factorization**

Firstly, we consider the original interaction matrix R of ratings where only some movies have been rated by each user ( most of the ratings are not available as its hardly possible for each user to have already watched all movies and rated them), our job is to predict the missing value based on what we have got. We want to factorize R such that:

$$R \approx U \cdot M^T$$

In order to obtain U and M, the alternating least squares algorithm will be used.

**B. Loss Function**

We want to minimize the error between the predicted matrix and the original matrix R, the following would be the loss function that we are going to use:

$$L = \arg\min_{U,M} \sum_{\{i,j|r_{i,j}\neq 0\}} (r_{i,j} - u_i m_j^T)^2 + \lambda(\sum_i (\|u_i\|)^2 + \sum_j (\|m_j\|)^2)$$

Where ri,j is the entry in the i-th row and j-th column of R, with being the regularization factor.This regularization scheme to avoid over-fifitting is called weighted--regularization.

By fifixing one of the matrix U or M, we obtain a quadratic form which can be solve directly, then the gradient descent optimization process could be used to update U and M alternatively.

**C. Gradient Descent**

The basic steps of gradient descent process will be discussed in this section. From the loss function, the partial derivative with respect to ui and mj can be calculated respectively:

- The derivative with respect to ui:

$$\frac{\partial L}{\partial u_i} = \frac{\partial[\min_{U,M} \sum_{\{i,j|r_{i,j}\neq 0\}} (r_{i,j} - u_i m_j^T)^2 + \lambda \sum_i (\|u_i\|)^2]}{\partial u_i} = \sum_i 2(u_i m_j^T - r_{i,j}) m_j + 2\lambda u_i$$

- Gradient descent iteration:

$$u_i := u_i - \alpha \cdot \frac{\partial L}{\partial u_i} = u_i - \alpha \cdot [\sum_j 2(u_i m_j^T - r_{i,j}) m_j + 2\lambda u_i]$$

- Similarity:

$$m_j := m_j - \alpha \cdot \frac{\partial L}{\partial m_j} = m_j - \alpha \cdot [\sum_i 2(u_i m_j^T - r_{i,j}) u_i + 2\lambda m_j]$$

Notice : $\alpha$ is the learning rate parameter which is used to control how much the coefficients can change on each update. The gradient descending algorithm will be repeated until the loss function converges.

**D. Weighted Average Rating**

As mentioned above, the predicted matrix for the original interaction matrix R, and the predictions for the matrices after the movies are grouped according to their genres will all be computed. Weights will be assigned to each of the predicted values.

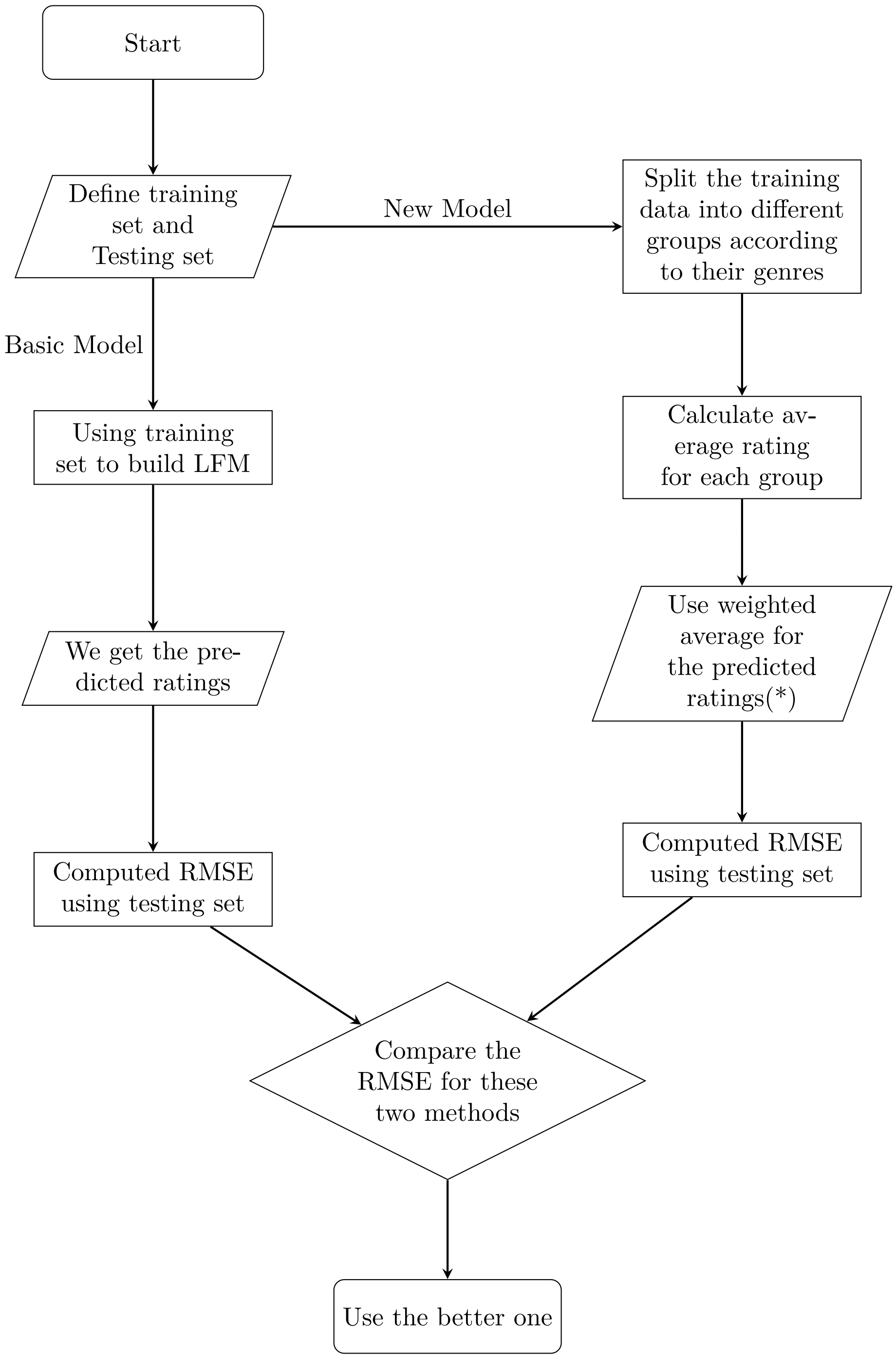
**E. Recommendation List**

The Final predicted interaction matrix can then be converted back to data frame, so that the ratings given by each user to each of the movies can be store in a file, then we can find out the highest rated movies for different user and make a recommendation list for each of them.

## 7.3 Experimental Result

We implemented the proposed method for constructing a collaborative filtering movie recommender system.

Due to the finite computational ability, only a subset of the full MovieLens dataset was used. The proposed method was implemented with Python. We split our dataset into training data and testing data, the model was trained by using the training data, and evaluation metrics (we used Root Mean Square Error (RMSE) here) were calculated from the testing data. The same metric was used for other two methods in this paper, we will compare them in the end. The flowchart of the model is shown below.




(*) : detailed calculated would be presented later.

There are some important things about the main process that need to be noticed:

- How to decide the value of k: In general, the model would be more accurate(in terms of RMSE) with larger number of latent factors. However, the computation time will also increase as k increase.

  Therefore, we would plot RMSE against different value of k, and see if there is any minimum, if the minimum point exist(and the computation times are similar), we would take it as the final value of k; if not, we considered the accuracy vs computation time tradeoff, by looking for the value of k beyond which the model stops yielding significant better result while the computation time increases a lot.

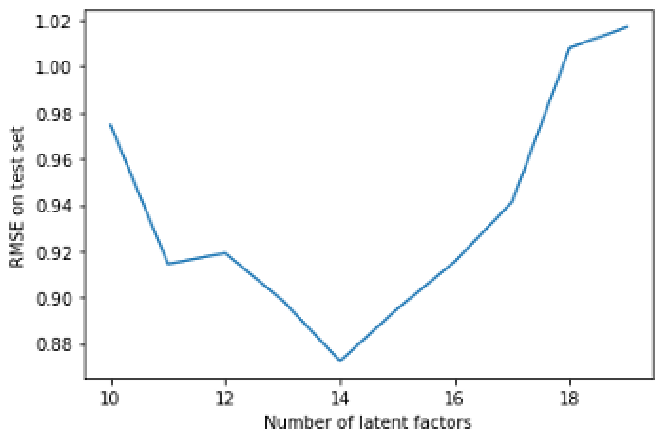



Figure 6: Variation of RMSE with number of latent factors

The plot below indicates that k=14 is the optimal value:

- How to calculate the weighted average of predicted ratings(FOR EACH MOVIE):

  Notations :

  (1) Nj=the number of genre that movie j belongs to.

  (2) PRnew(i,j)= predicted rating user i gives to movie j in our new model.

  (3) PR(i,j)= predicted rating user i gives to movie j in our basic model.

  (4) Avek= average rating for movies in genre, where k Comedy, action, ...

  Formula :

$$PR_{new(i,j)} = \omega_1 \times PR_{(i,j)} + \omega_2 \times \frac{\sum_{\forall k.that.j.has} Ave_k}{N_j}$$

$$\omega_1 + \omega_2 = 1$$

  $\omega_1$=weight assigns to the basic model

  $\omega_2$=weight assigns to the second part

  Initial values of $\omega_1$ and $\omega_2$ were set to 0.9 and 0.1 respectively, and we would change the values of $\omega_1$ and $\omega_2$ in order to find their optimal values (in terms of RMSE of testing set). From the plot obtained, we knew that when $\omega_1 = 0.8$ and $\omega_2 = 0.2$, the RMSE of testing set is the smallest.

**Prediction Accurancy**

The RMSE for the basic Model is about 1.383, while the RMSE of our initial new model (k=3; $\omega_1$=0.9; $\omega_2$=0.1) is only 1.298, which mean the accuracy of the Latent Factor Model does improve after we take the genres of items into consideration, even before we try to optimize the values of k, $\omega_1$ and $\omega_2$. One of the possible explanation is that this is kind of like a hybrid system, because some of the known features of movies are used. After the optimal values (which we obtained from the plots above) of k=14, $\omega_1$=0.8 and $\omega_2$=0.2 were used, the RMSE decreased to only 0.821 for our testing set, and this value would be used to compare with the results from the other two models in the end.

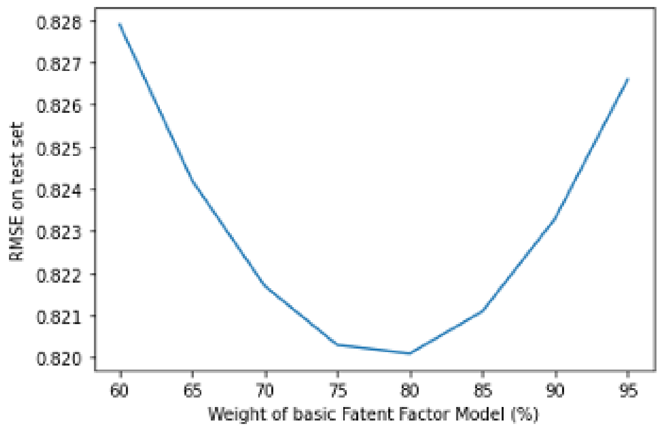



Figure 7: Variation of RMSE with weight of basic LFM

# 8 Discussion and Conclusion

## 8.1 Evaluation Criteria

Evaluation indexes are mainly used to evaluate the performance of the recommendation system in various aspects, which can be usually divided into offline evaluation and online testing according to application scenarios.

The main offline evaluation methods include Root Mean Squared Error (RMSE), holdout test, cross test, retention test and self-help method, etc. The evaluation indexes mainly include user satisfaction, prediction accuracy, recall rate, coverage rate, diversity, novelty, popularity, root mean square error, logarithmic loss, P-R curve, AUC, ROC curve and so on. The evaluation methods of online testing mainly include A/B testing, Interleaving method, etc. The evaluation indicators mainly include CTR, conversion rate, retention rate, average number of clicks, etc.

This paper will mainly focus on one of the most important off-line evaluation methods, that is RMSE. The RMSE lower, the model better.

## 8.2 Summary

Table 1: Comparison of RMSE of three models

| | K-means | Cuckoo Search | Latent Factor Model |
|---|---|---|---|
| RMSE | 1.004 | 1.050 | 0.820 |

Overall, from all 3 models using the same evaluation method, RMSE, the comparison among 3 models RMSE above shows that the latent factor model method achieves the smallest RMSE valued 0.8201, which is lower than 1.004 of the K-means model and 1.050 for the cuckoo search model. Therefore low RMSE of latent factor model indicates the greatest potential to improve the performance of movie recommender system.

## 8.3 Limitations and Future Study

The latent factor model usually have strong generalization ability, better scalability and flexibility. Moreover, the space complexity of this model is relatively low, which is possible to conduct the recommending process without considerable user or item features. While these advantages and its lowest RMSE in this paper, latent factor model still cannot recommend items to users without historic data, which means user and item cold start issues will not enable this model to perform well.

There also exists limitations on choosing RMSE as the evaluation criteria. Since in general, RMSE can well reflect the deviation between the predicted value and the real value of the regression model. While in practical application, if there are individual outliers with very large deviation degree, even if the number of outliers is very small, RSME will indicate very poor index.

Therefore, in the future, it is still worthy thinking about the issue about how to deal with the possible item or user cold start problems by constructing a better recommendation model. And to solve the possible inaccurate result of RMSE, a similar assessment can be performed using the more robust Mean Absolute Percent Error (MAPE), which can normalized the errors of each point and reduced the influence of absolute errors brought by an outlier point compared with RMSE.

# References


[1] Ahuja, Rishabh, Arun Solanki, and Anand Nayyar. Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. In 2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence), 26368. Noida, India: IEEE, 2019. https://doi.org/10.1109/CONFLUENCE.2019.8776969.

[2] Badaro, Gilbert, Hazem Hajj, Wassim El-Hajj, and Lama Nachman. A Hybrid Approach with Collaborative Filtering for Recommender Systems. In 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), 34954. Sardinia, Italy: IEEE, 2013. https://doi.org/10.1109/IWCMC.2013.6583584.

[3] Bhalse, Nisha, and Ramesh Thakur. Algorithm for Movie Recommendation System Using Collaborative Filtering. Materials Today: Proceedings, February 2021, S2214785321003242. https://doi.org/10.1016/j.matpr.2021.01.235.

[4] Breese, John S., David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. ArXiv:1301.7363 [Cs], January 30, 2013. http://arxiv.org/abs/1301.7363.

[5] Geetha, G, M Safa, C Fancy, and D Saranya. A Hybrid Approach Using Collaborative Filtering and Content Based Filtering for Recommender System. Journal of Physics: Conference Series 1000 (April 2018): 012101. https://doi.org/10.1088/1742-6596/1000/1/012101.

[6] Hofmann, Thomas. Latent Semantic Models for Collaborative Filtering. ACM Transactions on Information Systems 22, no. 1 (January 2004): 89115. https://doi.org/10.1145/963770.963774.

[7] Huang, Yinghui, Hui Liu, Weiqing Li, Zichao Wang, Xiangen Hu, and Weijun Wang. Lifestyles in Amazon: Evidence from Online Reviews Enhanced Recommender System. International Journal of Market Research 62, no. 6 (November 2020): 689706. https://doi.org/10.1177/1470785319844146.

[8] Inan, Emrah, Fatih Tekbacak, and Cemalettin Ozturk. Moreopt: A Goal Programming Based Movie Recommender System. Journal of Computational Science 28 (September 2018): 4350. https://doi.org/10.1016/j.jocs.2018.08.004.

[9] Joorabloo, Nima, Mahdi Jalili, and Yongli Ren. A Probabilistic Graph-Based Method to Solve Precision-Diversity Dilemma in Recommender Systems. Expert Systems with Applications 184 (December 2021): 115485. https://doi.org/10.1016/j.eswa.2021.115485.

[10] Kalita, Jugal, Valentina Emilia Balas, Samarjeet Borah, and Ratika Pradhan, eds. Recent Developments in Machine Learning and Data Analytics: IC3 2018. Vol. 740. Advances in Intelligent Systems and Computing. Singapore: Springer Singapore, 2019. https://doi.org/10.1007/978-981-13-1280-9.

[11] Katarya, Rahul, and Om Prakash Verma. An Effective Collaborative Movie Recommender System with Cuckoo Search. Egyptian Informatics Journal 18, no. 2 (July 2017): 10512. https://doi.org/10.1016/j.eij.2016.10.002.

[12] . An Effective Collaborative Movie Recommender System with Cuckoo Search. Egyptian Informatics Journal 18, no. 2 (July 2017): 10512. https://doi.org/10.1016/j.eij.2016.10.002.

[13] Li, Hui, Fei Cai, and Zhifang Liao. Content-Based Filtering Recommendation Algorithm Using HMM. In 2012 Fourth International Conference on Computational and Information Sciences, 27577. Chongqing, China: IEEE, 2012. https://doi.org/10.1109/ICCIS.2012.112.

[14] Mansur, Farhin, Vibha Patel, and Mihir Patel. A Review on Recommender Systems. In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 16. Coimbatore: IEEE, 2017. https://doi.org/10.1109/ICIIECS.2017.8276182.

[15] Sarwar, Badrul, George Karypis, Joseph Konstan, and John Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. In Proceedings of the Tenth International Conference on World Wide Web - WWW 01, 28595. Hong Kong, Hong Kong: ACM Press, 2001. https://doi.org/10.1145/371920.372071.

[16] Singh, Rahul Kumar, Pardeep Singh, and Gourav Bathla. User-Review Oriented Social Recommender System for Event Planning. Ingénierie Des Systèmes d Information 25, no. 5 (November 10, 2020): 66975. https://doi.org/10.18280/isi.250514.

[17] Smitha, N, D Anusha, C Chaithanya, J Sindhu, R Tanuja, and H S Hemanth Kumar. A Review on Movie Recommendation System Using Machine Learning. In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 76973. Tirunelveli, India: IEEE, 2021. https://doi.org/10.1109/ICICV50876.2021.9388619.

[18] Sunilkumar, Chaurasia Neha. A Review of Movie Recommendation System: Limitations, Survey and Challenges. ELCVIA Electronic Letters on Computer Vision and Image Analysis 19, no. 3 (September 17, 2020): 18. https://doi.org/10.5565/rev/elcvia.1232.

[19] Tahmasebi, Hossein, Reza Ravanmehr, and Rezvan Mohamadrezaei. Social Movie Recommender System Based on Deep Autoencoder Network Using Twitter Data. Neural Computing and Applications 33, no. 5 (March 2021): 160723. https://doi.org/10.1007/s00521-020-05085-1.

[20] Takács, Gábor, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major Components of the Gravity Recommendation System. ACM SIGKDD Explorations Newsletter 9, no. 2 (December 2007): 8083. https://doi.org/10.1145/1345448.1345466.

[21] Bobadilla J, Serradilla F, Hernando A. Collaborative filtering adapted to recommender systems of e-learning[J]. Knowledge-Based Systems, 2009, 22(4): 261-265.

# Appendix

# A Code of K-means KNN Model

## A.1 K-means Algorithm

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
from sklearn.model_selection import train_test_split
```

```
data = pd.read_csv("ratings_small.csv")
data = pd.DataFrame(data)
rating = np.array(data['rating'])
meanrating=np.mean(rating)
userid0 = np.array(data['userId'])
movieid0 = np.array(data['movieId'])


def getNonRepeatList(data):
    new_data = []
    for i in range(len(data)):
        if data[i] not in new_data:
            new_data.append(data[i])
    return new_data


userid = getNonRepeatList(userid0)
# userid_train,userid_test = train_test_split(userid,test_size=0.2)
movieid = sorted(getNonRepeatList(movieid0))
DF = pd.DataFrame(index=userid, columns=movieid)
for i in range(len(rating)):
    DF.loc[userid0[i], movieid0[i]] = rating[i]
m = np.mean(rating)
DF.fillna(0, inplace=True)
a = np.array(DF).T

meandistortions = []
K = range(1, 21)

for k in range(1, 21):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(a)
    meandistortions.append(sum((np.min(
        cdist(a, kmeans.cluster_centers_,
              'euclidean'), axis=1)) ** 2) / a.shape[0])
plt.plot(K, meandistortions, 'bx-')
plt.xlabel("number of clusters")
plt.ylabel("WCSS")
plt.show()

kmeans = KMeans(n_clusters=8)
kmeans.fit(a)
df_cluster = pd.DataFrame(columns=userid, index=sorted(getNonRepeatList(kmeans.
    labels_)))
movierank = range(len(movieid))
for i in sorted(getNonRepeatList(kmeans.labels_)):
    for j in userid:
        count = 0
        score = 0
        for k in movierank:
            if kmeans.labels_[k] == i and DF.loc[j, movieid[k]] != 0:
                count += 1
                score += DF.loc[j, movieid[k]]
        if count != 0:
            df_cluster.loc[i, j] = score / count
df_cluster.fillna(0, inplace=True)
corr = df_cluster.corr(method='pearson')

```

```
corr = df_cluster.corr(method="pearson")
corr_rank = corr.rank(ascending=False)
testDF = pd.read_csv("rating_test.csv",header=None)
testDF = pd.DataFrame(testDF)
rating = []
rating_real = np.array(testDF.iloc[:,2])
print(testDF)

for i in range(testDF.shape[0]):
    testuser = testDF.iloc[i,0]
    testmovie = testDF.iloc[i,1]
    countuser = 0
    userrank = []
    while countuser < 671:
        totalscore = 0
        countmovie = 0
        countuser += 1
        userrank.append(countuser + 1)
        nearest_user = np.array(corr_rank[corr_rank.loc[:, testuser].isin(userrank)
            ].index)
        for j in nearest_user:
            if DF.loc[j, testmovie] != 0:
                countmovie += 1
                totalscore += DF.loc[j, testmovie]
        if countmovie == 5:
            break
    if countmovie==0:
        rating.append(meanrating)
    else:
        rating.append(totalscore / countmovie)
print(len(rating_real))
print(len(rating))
rating=np.array(rating)
rating_real = np.array(rating_real)
RMSE = (sum((rating-rating_real)**2)/len(rating))**0.5
print(RMSE)
```

### A.2 Algorithm on Test set

```
import pandas as pd
if __name__ == "__main__":
    data = pd.read_csv("ratings_small.csv")
    data:pd.DataFrame = data.sample(frac = 1.0)
    rows,cols = data.shape
    split_index_1 = int(rows*0.02)
    data_test:pd.DataFrame = data.iloc[0:split_index_1,:]
    data_train:pd.DataFrame = data.iloc[split_index_1:rows,:]
    data_test.to_csv("rating_test.csv",header=None,index=False)
```

## B Code of Cuckoo Search Model

```python
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from tqdm import tqdm
import random


## parameters
group_no = 4
Pa = 0.5
ite = 1
raw_train_1_Pa = 0.6
raw_train_2_Pa = 1-0.2/(1-raw_train_1_Pa)
file_path = 'C:\\Users\\Zhiyuan\\Desktop\\project\\ratings_small.csv'



rmse_array = np.array([])
for ite_n in tqdm(range(ite)):
    raw = pd.read_csv(file_path)
    raw.set_index(['userId','movieId'],inplace = True)
    raw = raw.unstack().rating.fillna(0)
    raw_origin = raw.copy(deep = True)
    film_index = raw_origin.columns
    raw['location'] = raw.apply(lambda x: np.array(x),axis = 1)

    # divide users into Ua(raw_train_1), Ub(raw_train_2) and test group(raw_test)
    raw_train_1_df = raw_origin.sample(frac = raw_train_1_Pa)
    raw_train_1 = raw_train_1_df.copy(deep = True)
    raw_train_1 = np.array(raw_train_1)

    raw_origin = raw_origin.append(raw_train_1_df)
    difference_set_result_1 = raw_origin.drop_duplicates(film_index,keep=False)

    raw_train_2 = difference_set_result_1.sample(frac = raw_train_2_Pa)
    raw_train_2_df = raw_train_2.copy(deep = True)
    raw_train_2 = np.array(raw_train_2)

    difference_set_result_1 = difference_set_result_1.append(raw_train_2_df)
    raw_test = difference_set_result_1.drop_duplicates(film_index,keep=False)
    raw_test_df = raw_test.copy(deep = True)


    # k-means
    clf = KMeans(n_clusters=group_no)
    clf.fit(raw_train_1)

    centers = clf.cluster_centers_
    labels = clf.labels_


    # get location, cluster number, center location and distance to center of the
        users in Ua
    raw_train_1_df['location'] = raw_train_1_df.apply(lambda x: np.array(x),axis =
        1)
    raw_train_1_df['cluster'] = labels
    raw_train_1_df['centers'] = raw_train_1_df['cluster'].apply(lambda x: centers[x
        ])
```

```python
    raw_train_1_df['distance'] = raw_train_1_df.apply(lambda x: np.sqrt(np.sum(np.power(x['location']-x['centers'],2))),axis = 1)


    ## calculate ave_distance of each cluster
    ave_distance = np.array([])
    for i in range(group_no):
        ave_distance = np.append(ave_distance, np.mean(raw_train_1_df[raw_train_1_df['cluster'] == i]['distance']))

    total_df = pd.concat([raw, raw_train_1_df[['cluster','centers','distance']]],axis = 1,sort = True)


    ## add Ub users to training group
    for i in range(len(raw_train_2_df)):
        distance_list = np.array([])
        distance_ratio_list = np.array([])
        for j in range(len(centers)):
            distance = np.sqrt(np.sum(np.power(np.array(raw_train_2_df.loc[raw_train_2_df.index[i]]) - centers[j],2)))
#            distance_ratio = distance/ave_distance[j]
            distance_list = np.append(distance_list,distance)
            distance_ratio_list = np.append(distance_ratio_list,distance/ave_distance[j])

        total_df.loc[raw_train_2_df.index[i],'cluster'] = np.argmin(distance_ratio_list)
        total_df.loc[raw_train_2_df.index[i],'distance'] = distance_list[np.argmin(distance_ratio_list)]

        if total_df.loc[raw_train_2_df.index[i],'distance'] < np.mean(total_df[total_df['cluster'] == np.argmin(distance_ratio_list)]['distance']):

            index = np.argmax(total_df[total_df['cluster'] == np.argmin(distance_ratio_list)]['distance'])
            userId = total_df[total_df['cluster'] == np.argmin(distance_ratio_list)].index[index]
            rand = random.random()
            if rand < Pa:
                total_df.drop(total_df[total_df.index == userId].index, inplace=True)
            else:
                continue
        else:
            total_df.drop(total_df[total_df.index == raw_train_2_df.index[i]].index, inplace = True)


    # calculate average marks in each cluster and get mark_list_nan
    mark_list_nan = pd.DataFrame()
    mark = total_df.iloc[:,:-4]
    mark = mark.replace(0,np.nan)
    mark = pd.concat([mark,total_df['cluster']],axis = 1)
    for i in range(group_no):
        mark_list_nan[i] = mark[mark['cluster'] == i].iloc[:,:-1].mean()


    # reset the center of each cluster and get centers_new
```

```python
    mark_list = pd.DataFrame()
    mark = total_df.iloc[:,:-4]
    mark = pd.concat([mark,total_df['cluster']],axis = 1)
    for i in range(group_no):
        mark_list[i] = mark[mark['cluster'] == i].iloc[:,:-1].mean()
    centers_new = np.array(mark_list.T)
    centers_new_nan = np.array(mark_list_nan.T)


    # renew total_df
    total_df_new = total_df.iloc[:,:-4]
    total_df_new = total_df_new.append(raw_test_df)
    total_df_new = total_df_new.drop_duplicates(film_index,keep=False)

    total_df_new['location'] = total_df_new.apply(lambda x: np.array(x),axis = 1)
    total_df_new['cluster'] = total_df['cluster']
    total_df_new['centers'] = total_df_new['cluster'].apply(lambda x: centers_new[int(x)])
    total_df_new['distance'] = total_df_new.apply(lambda x: np.sqrt(np.sum(np.power(x['location']-x['centers'],2))),axis = 1)


    ## calculate ave_distance of each cluster
    ave_distance = np.array([])
    for i in range(group_no):
        ave_distance = np.append(ave_distance, np.mean(raw_train_1_df[raw_train_1_df['cluster'] == i]['distance']))

    total_df_new = pd.concat([raw, total_df_new[['cluster','centers','distance']]],axis = 1,sort = True)


    # calculate distance of each test user to according cluster center
    for i in range(len(raw_test_df)):
        distance_list = np.array([])
        distance_ratio_list = np.array([])
        for j in range(len(centers_new)):
            distance = np.sqrt(np.sum(np.power(np.array(raw_test_df.loc[raw_test_df.index[i]]) - centers_new[j],2)))
#            distance_ratio = distance/ave_distance[j]
            distance_list = np.append(distance_list,distance)
            distance_ratio_list = np.append(distance_ratio_list,distance/ave_distance[j])

        total_df_new.loc[raw_test_df.index[i],'cluster'] = np.argmin(distance_ratio_list)
        total_df_new.loc[raw_test_df.index[i],'distance'] = distance_list[np.argmin(distance_ratio_list)]


    # calculate RMSE
    raw_test_result = total_df_new.loc[raw_test_df.index]
    raw_test_result['RMSE'] = raw_test_result.apply(lambda x: np.array(x['location']),axis = 1)
    for i in raw_test_result.index:
        location_nan = np.array([np.nan if j==0 else j for j in raw_test_result.loc[i,'location']])
        center_nan = np.array([np.nan if j==0 else j for j in centers_new_nan[int(raw_test_result.loc[i,'cluster'])]])
```

```
        raw_test_result.loc[i,'RMSE'] = np.sqrt(np.mean(np.power((location_nan -
            center_nan)[(location_nan -center_nan <11)],2)))
    rmse_array = np.append(rmse_array, np.mean(np.abs(raw_test_result['RMSE'])))


print(np.mean(rmse_array))
```

# C Code of Latent Factor Model

Since the amount of code for this model is comparatively huge, the detail of the model can be obtained from the link below.

https://www.kaggle.com/xavier001/latent-factor-model?scriptVersionId=73877311