# A Movie Recommendation System Using K-mean Clustering and K-nearest neighbor

Krishna Kanhaiya
student Id. 109335377
Stony Brook University
Stony Brook, New York
krishna.kanhaiya@stonybrook.edu

Gautam Kumar
student Id. 109335378
Stony Brook University
Stony Brook, New York
gautam.kumar@stonybrook.edu

## ABSTRACT
In this paper, we have described the implementation of a movie recommender system using k-mean clustering and k-nearest neighbor. The entire process starts with preprocessing of data, which includes converting the dataset into structured form, reducing the dimension using singular value decomposition, and finally predicting the missing value using naive bayes. Then the dataset is divided into clusters using k-mean clustering, and on the top of which we recommend a new user for movies using k-nearest neighbors.

## General Terms
Clustering, recommendation, data mining, dimensionality reduction, missing value prediction, nearest neighbors

## Keywords
K-mean clustering, K-nearest neighbor, Singular value decomposition, Naive Bayes

## 1. INTRODUCTION
Prediction of user respons corresponding to different options by a class of web application is called recommendation system[1]. Recommendation system can be classified into two main group. That is content based systems and collaborative based systems.

- Content Based Systems :- These kind of systems do recommendation based on person's past history[1]. For example if someone has watched comedy movie a lot, then these kind of systems do make movie recommendation having comedy genre from it's database, for that particular person. Youtube uses this kind of systems.
- Collaborative Filtering :- In these kind of recommendation systems, the recommendation is done on the basis of what other people similar to that of you in many ways have prefered for[1]. This kind of recommendation is done at 'amazon.com'. Like if we buy something then it gives recommendation that those who bought these books also went for the following books.

With the opening of global market and the dawn of online shopping, recommendation system has become a very useful tool to capitalise on this growing market. Today recommendation system is finding it's extensive usage in the following area of our life[1].

- Product Recommendation :- Today Product Recommendation is the main area in which recommendation systems are being used most extensively. All the online retailers like 'Amazon' uses it on a massive scale. Once we decide to buy something, then based on the fact that it's past customers who went for that product also prefered to go for some more products along with it, product recommendation system do recommend it's customer with those extra products
- News Article Recommendation :- Based on what article we read, News article recommendation system do recommend us with articles of similar contents. Like in the case of 'Youtube', Based on our history 'Youtube' recommend us with videos having similar topic. Also based on the most trending topic that is being read or watched by other users, we are recommended for it.
- Movie Recommendation :- Movie recommendation is also an important aspect of it. With the growing number of movies being produced day by day, today's generation wants to have a prior information regarding the movie they should go for, and the movie which is certainly not of their taste.

With these motivation, we went to build a movie recommendation system for our data mining course project. It involves several commonly known algorithms like K-mean clustering, K-nearest neighbors, singular value decomposition, and naive bayes.

## 2. PRIOR WORK
A lot's of work has been done in the area of recommendation system. And depending on the requirements

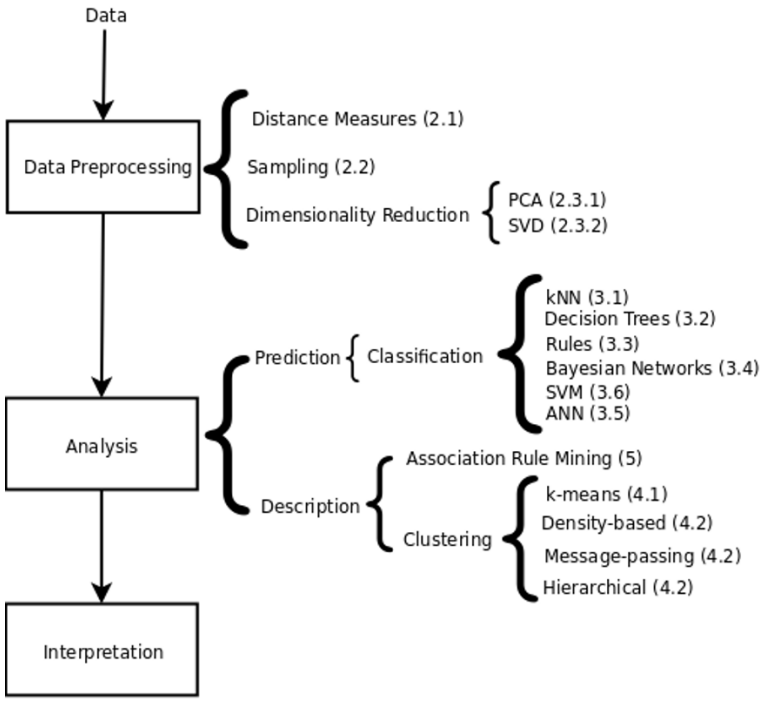


**Figure 1: Different work flow that can be taken to build a recommendation system**

# 3. A MOVIE RECOMMENDER MODEL

## 3.1 Data statistics

Initially the movie dataset was taken from the online source as mentioned in reference[2]. Initially the dataset was having the following attributes.

- used Id.
- Age of the user.
- Gender of the user.
- Occupation of the user.
- Zip code of the place where this user lives.
- The movie watched by him/her.
- The genre of the movie.
- The rating given to that movie by this user after watching the movie.
- Time stamp at which the movie was watched.

The entire dataset has got one million ratings given by six thousand users over four thousand movie. It was like one user watches more than one movie, and each cobination like user A waching movie B, and user A watching movie C ware treated as different data points.

## 3.2 Data Preprocessing

The data set was cleaned by removing all sorts of noise from it and converting it into structured dataset.
Then the dimensionality of the data set was reduced using singular value decomposition or popularly known as SVD.

- Singular Value Decomposition :- In singular value decomposition, we take the entire dataset as a matrix, and decompose it into product of three smaller matrix.

  $$M = U\Sigma V'$$

  The middle matrix is a diagonal matrix, consisting of the eigen values. The attributes corresponding to the dominant eigen values is retained and the rest are discarded. In this way we were able to achieve dimensionality reduction using singular value decomposition.

Using singular value decomposition we were able to bring down the dimensionality of the dataset from nine to seven. The left over attributes were user Id, age, gender, occupation, movie watched by him/her, genre of the movie, and the rating given to it.
Then we proceeded with predicting the missing values using the naive baye's.

$$P(A/B_1B_2B_3) = \frac{P(A) * \Pi_i P(B_i/A)}{P(B_1B_2B_3)}$$

The above formula tries to predict the probability of all possible value that the attribute $A$ can take given that corresponding to other attributes, we know the corresponding values. By baye's theorem and conditional independence it reduces to the right hand side of the equation. That is to

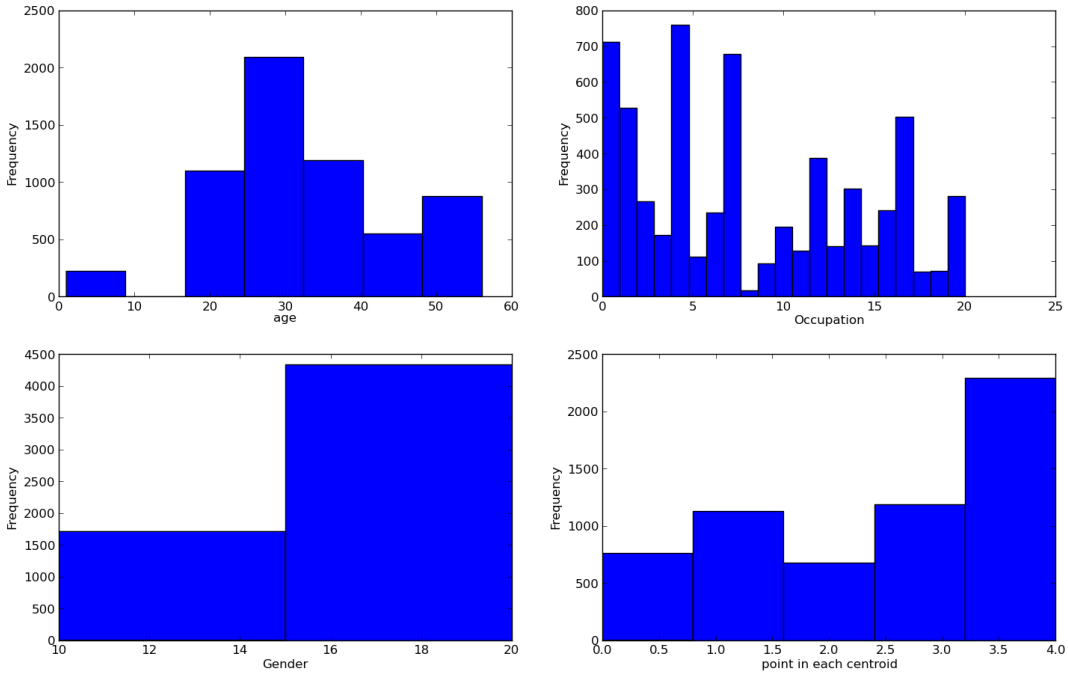


**Figure 2: Data statistics corresponding to different attributes**

find the probability of $A$ multiplied by the conditional probability of rest attributes given $A$.
Next we divided our data set into two parts. The training part, which consist of ninty percent of the data set, and the rest ten percent constitute the test part. The training part is the portion of dataset on which we will be building our movie recommender system. And the test part would be the one which will help us to evaluate the performance of our recommender system.

## 3.3 K-mean Clustering

K-mean clustering is a very classical way of unsupervised learning, which is used extensively in data mining and machine learning. The algorithm starts with the assumption that there are k-clusters in the dataset. Then using k-seed values as centroids, we calculate the distance of each point from these centroids. After that each point is associated to a cluster based on the fact that the centroid of that cluster is nearest to that point. The entire process is explained below in step by step manner.

1. Initially take k seeds as the k centroids
2. calculate the distance of each point from all the k-centroids using euclidean measure.

$$distance = \sqrt{\Sigma_{i=1}^{n}(X_i - Y_i)^2}$$

3. the centroid which is nearest, is the one with which a particular point gets associated. Or in other words become part of the cluster whose centroid is nearest to it.
4. Once all the points have been associated with a cluster, we recalculate the k-centroids by taking the mean of all the points in a particular cluster. Thus evaluating it's new centroid.

$$mean = \frac{\Sigma_{i=1}^{N}(X_i)}{N}$$

5. We repeat step 2 to 4, till the change in position of the k centroids are negligible in the following itterations.

## 3.4 Issues with k-mean clustering

There are broadly two issues with k-mean clustering. They are discussed below.

- K-mean clustering assumes that we know in prior that what is the value of k. In other words how many clusters are there in the data set.
- Making the initial guess for the seeds, because a good guess reduces the number of iterations for which the algorithm will run before it converges.

To solve the second issue, we tried to have the k centroids as far as possible.

## 3.5 Finding K for the K-mean

Objective function is defined as the total distance of the point from their respective centroids once the k-mean has been done. Clearly the value of objective function is going to decrease, as the value of k increases. For example objective function is maximum when k=1, and minimum when k=n, that is the number of datapoints in the data set. In fact in the last case, objective function is zero, as all the points in the dataset, themself becomes the k centroids.

$$ObjectiveFunction = \sqrt{\Sigma_{i=1}^{K}\Sigma_j(X_{ij} - \mu_j)^2}$$

In order to get the value of k for k-mean we did a small experiment. The steps of which are discussed below.

1. We took a small data sample from our training dataset.
2. We applied k-mean clustering to it, with the value of k taken in increasing order. That is we started with k=1, then proceeded with k=2,3,4 and so on.
3. After the completion of k-mean for each value of k, we calculated the value of objective function.
4. The value of obejctive function falls sharply initially, and then the change is small. The very point at which this happens, is the value of k for which we should go for.

In our movie dataset, this point at which the change in objective function was not that sharp for the first time was with $K = 5$. The entire relation of objective function versus K for the movie data set has been plotted in Figure 3. Thus, we discovered that our dataset can be braodly divided into five clusters.

## 3.6 Clustering of movie dataset

With the motive that people of same age group, occupation and gender have similar taste in movies they go for, we went for k-mean clustering on the basis of age, occupation and gender. Also, the issue of k was resolved as discussed in earlier section. Thus we were able to divide our data set into five clusters. The 2-D scatterplot of the dataset, with occupation and age as attributes taken,all the five clusters along with their centroids have been shown in Figure 4.

## 3.7 K-Nearest Neighbors

K-Nearest Neighbor is an instance based learning algorithm, in which the algorithm or the idea proceed at the time a query is made. It involves calculating the distance of all the training points from a test point. And the distance to the k-nearest neighbor becomes the diameter for a d-dimensional sphere, which contains all it's k neighbors.

## 3.8 Advantage of Clustering

With the clustering of the training data set, we have built the backbone of the movie recommender system. Next when a user wants to have some recommendation of movie for himself/herself, the system will ask some basic question from it. They are about his/her age gender, occupation, and genre of the movie he/she prefers. Next we assign this new data point to the cluster with whose centroid it's distance is going to be minimum among the five clusters. Next in that cluster, we will discover some $k$ nearest neighbor for the new data point on the basis of age, occupation, gender attributes, and also that these neighbors should have the same genre as the new data point or the new user.
Had we avoided clustering, and had diectly calculated k-nearest neighbors, the compution involved would have increased many fold. As we know that in k-nearest neighbors, nothing is done initially. Once the new point arrive we calculate the distance of the new point from the rest of the training points. Thus each time we have to involve in n calculations of distance of the new point with a training point. Now once we have made say five clusters, then the new point is associated with a partcular cluster. Thus to get the k-nearest neighbor, now we have to just do $n/5$ calculations involving distance measure. Thus increasing the time coplexity of our algorithm.

## 3.9 Issue of K in K-Nearest Neighbors

The issue with K-Nearest Neighbor remains, the same as with, that of K-means, that is what should be the value of K in order to get a good result. For that we did a small experiment, whose idea is explained below.

1. We took a small data set from the test data.
2. we applied K-nearest neighbors for them on our clustered data set.
3. The movie associated with the test data, was looked in the k movie set associated with the k-neighbors. If it was present there, it was treated as a success, else a failure.
4. Step 2 and 3 were repeated with the value of K taken in an increasing order. That is we started with k=1, then proceeded with K=2,3,4 and so on.
5. Each time the performance was measured.

With the increasing value of k, the performance of our recommender system initially improved a lot, and then became stagnant. The very point at which this happened was taken as the k for our k-nearest neighbor. In our case it was with k=10. The entire experiment has been plotted in Figure 5.

# 4. RECOMMENDER SYSTEM AND IT'S PERFORMANCE

With this, our recommender system is ready. Now when the test dataset is feed one by one, based on their age, gender and occupation values, it's distance from the five centroid is calculated. And the cluster whose centroid is nearest to the new point, is the one with which the point gets associated. Now, in the cluster based on the three attributes of age, gender and occupation, K-nearest neighbor is calculated. But apart from the three attribute, it is also taken care of that these k-attributes must have the same genre as the new one and the movie associated with them have a rating of atleast four on a scale of five. If the last two conditions are not met then even though that point is close enough, it is not considered as the k-nearest neighbors.
The movie associated with these k nearest neighbors become the recommendation for the new guy. Now we look at the

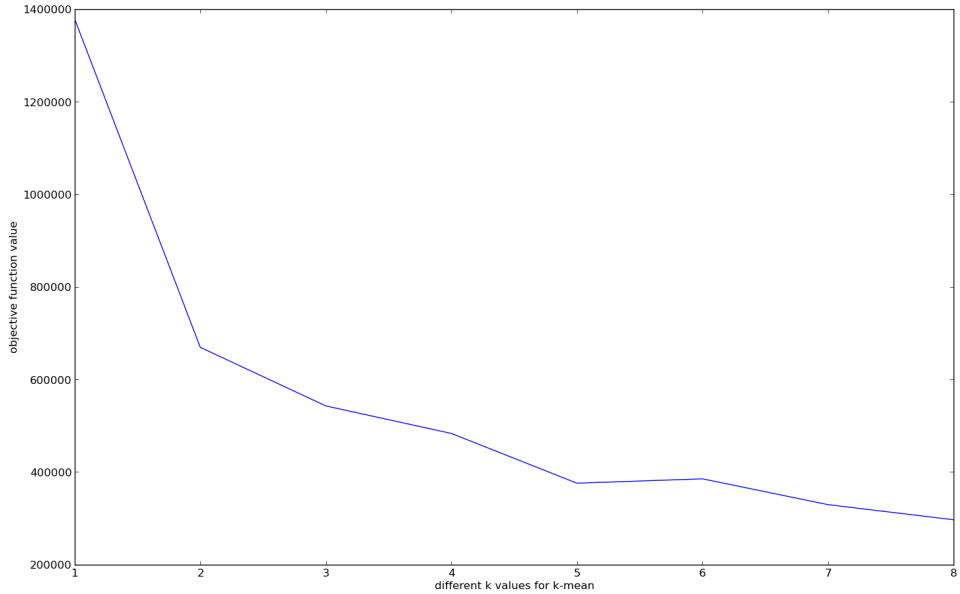


**Figure 3: Objective function versus k**

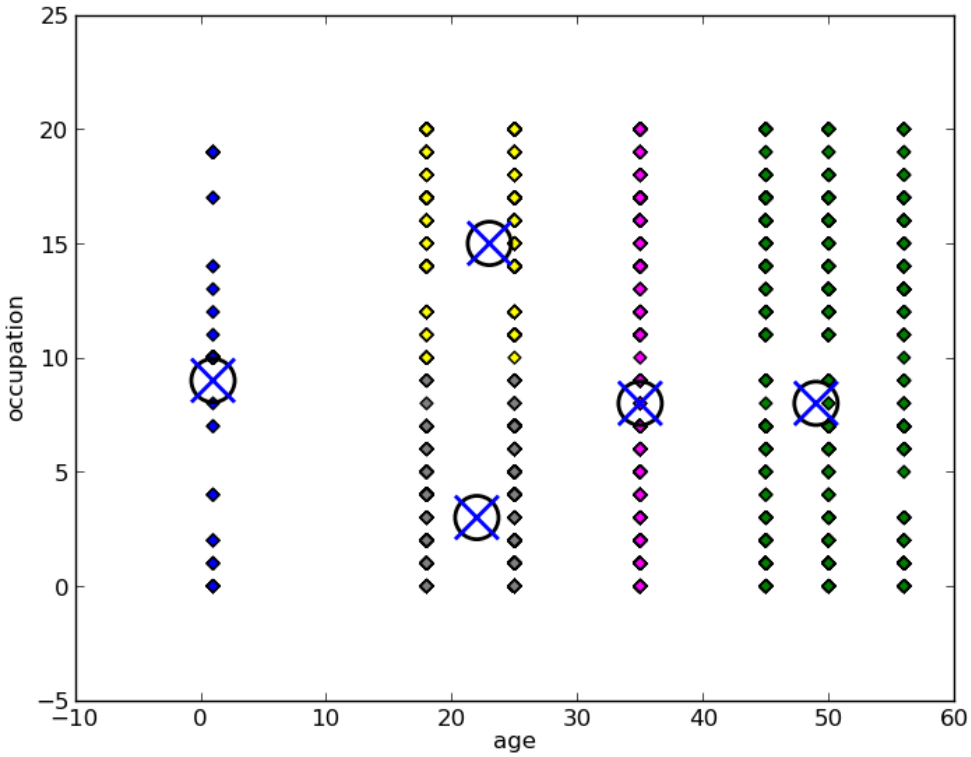


**Figure 4: Five clusters of the movie dataset along with their centroids**

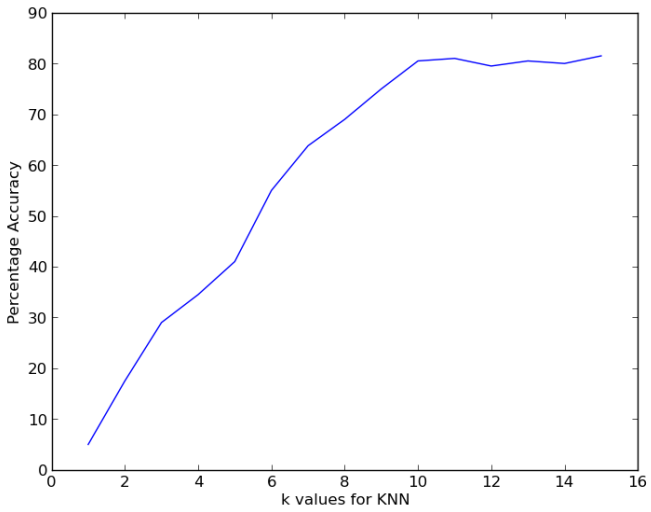


**Figure 5: Performance of recommender system as a function of k**

list of movie watched by that guy. If it contains even a single movie from the recommended list, we mark it a success, else a failure.
with this parameter to measure the performance of our recommendation system, we tested our test data set. And out of 600 test data 483 were found to give positive results, thus marking the performance accuracy as 80.6 percent.

# 5. REFERENCES

[1] A. Rajaraman, J. Leskovec, and J. D. Ullman. Chapter 9. Mining of Massive Datasets.
[2] http://grouplens.org/datasets/movielens/attachments
[3] Data Mining Methods for Recommender Systems, Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M. Pujol.