Predicting Movie Recommendations using MovieLens dataset

Group - 6

***Abstract*— Through this project we try to solve the problem of movie recommendation which predicts the ratings of movies for a user based on the movie ratings provided by other users. Recommendation systems are crucial part of information and e-commerce ecosystem, filtering through large information and product spaces. They use a different number of technologies and are broadly classified as**

Content-based systems look for the properties of items recommended. Suppose, a Netflix user has watched many action movies, then mostly likely recommended movie to the user is going to be of action genre.

Collaborative filtering systems recommend items based on similarity measures between users and/or items. In a recommendation-system application there are two classes of entities, which we shall refer to as users and items. Users have preferences for certain items, and these preferences must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item.

I. INTRODUCTION

Let us say that a target user seeks recommendations of movies. We would build a system that would examine the rankings of all movies provided by other viewers and use this information to find users with similar preferences to this user. The recommendation system would then combine their preferences and make predictions of movies that user might like.

To achieve the above task, models like K-Nearest Neighbors, K-means models seem to be more suitable as this is a recommendation task. Although this problem can be described as a classification problem, standard models like linear regression, logistic regression do not fit because of the huge amounts of missing data, and the non-linearity in the output ratings. In addition we implement Singular value decomposition which also works in a similar fashion. We also implement Neural Network model for a comparison. i.e we implement the following models.

- K-Nearest Neighbors
- K-means
- Singular Value Decomposition
- Neural Network

II. SUMMARY OF THE DATA AND PREPROCESSING

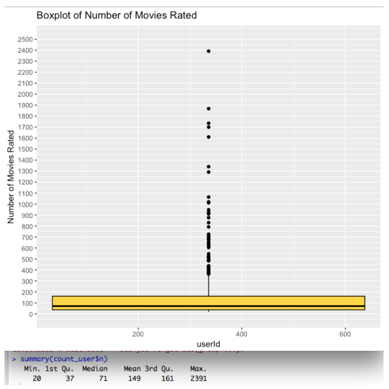
We will be using the ratings dataset generated from the MovieLens web site. This is a publicly available dataset collected by GroupLens. We have two datasets available, one contains $100,000$ ratings across 9066 movies created by 671 users. The other contains 1 million ratings over 3706 movies. The dataset contains the following attributes:

- UserId
- MovieId
- Rating
- Timestamp

Each movie and user will be represented by their respective IDs. Ratings are made on a 5-star scale, with half-star increments (0.5 - 5.0 stars).

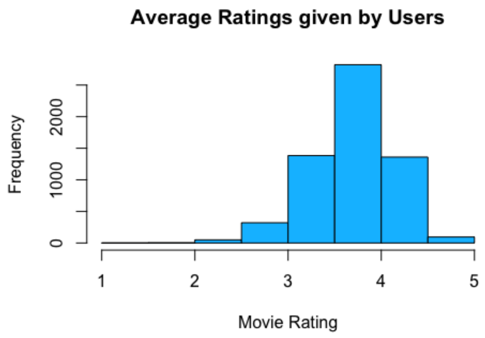
A. Observations

- We are assuming that each user's rating with be independent of the ratings of other users. There might be some similarity between user ratings.
- Each user will only rate a small subset of the total number of movies.

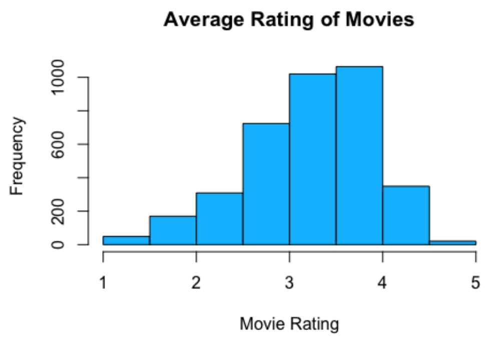


```
> summary(count_user$n)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
     20      37      71     149     161    2391 
```

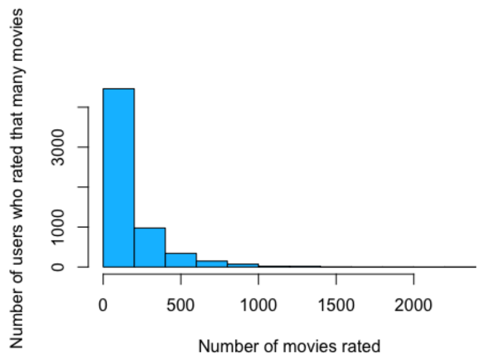
By analyzing the dataset, we can see that each user has rated at least 20 movies. The outliers include an user who rated more than 2000 movies.



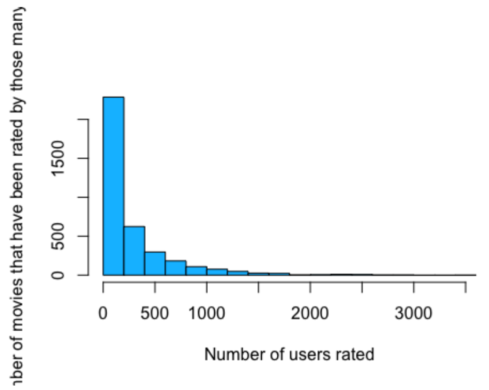
Most of the users have an average rating around 4, over the movies they have seen. Due to this distribution we divide the final prediction into two categories **like and dislike**, where we consider a movie as like if rating is greater than 4, and dislike otherwise.



Most movies have an average rating of 3.5-4



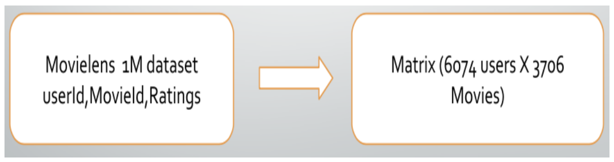
It can be seen that most of the users have rated less than 250 movies and as we increase the number of movies rated we find that the number of such users decreases.



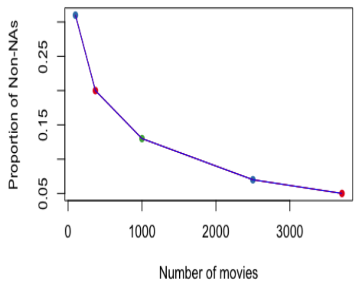
It can be seen that most of the movies have been rated by less than 250 users and as we increase the number of users we find that the number of such movies decreases.

B. Missing values

As the data is not available in ready to analyze format, we build a super matrix with users as rows and ratings of the movies as columns.


Movielens 1M dataset userId,MovieId,Ratings
Matrix (6074 users X 3706 Movies)

We observe that in the 100k dataset only 5% of the movies have been rated by more than 50 users, where the total number of movies is 9066, this results in an extremely sparse matrix. We further analyzed the dataset with 1 million ratings, this has a much better 67% of movies that has been rated by at least 50 users.

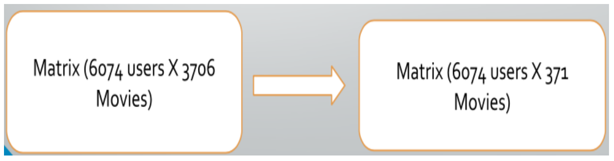


Graph representing the proportion of missing values

Further more as it can seen from the graph even in this dataset a staggering 95% of values are missing from the super matrix, which makes it quite difficult for the models to predict, this poses serious limitations on the area of models we can implement, for instance decision trees cannot be implemented because of the sheer number of missing predictors.

Replacing missing values with any kind of average would rather get rid of statistical significance as 95% of the table is now filled with average values.

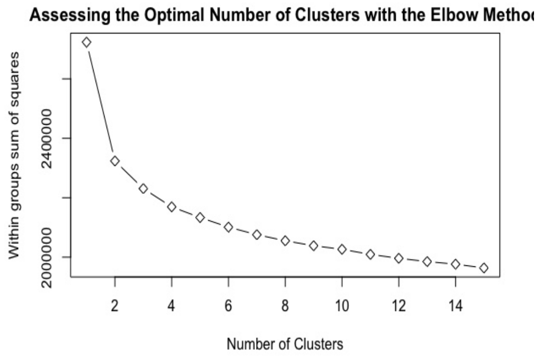
Hence we chose to reduce the dataset to decrease the missing values to a manageable proportion. i.e we choose to rank the movies according number of ratings given to them, and then chose the top **370** movies into our dataset. this matrix now has 20% of missing values which is relatively better. This reduces the final matrix to a dimension of **(6074, 371)**


Matrix (6074 users X 3706 Movies)
Matrix (6074 users X 371 Movies)

III. METHODS IMPLEMENTED

A. K-MEANS

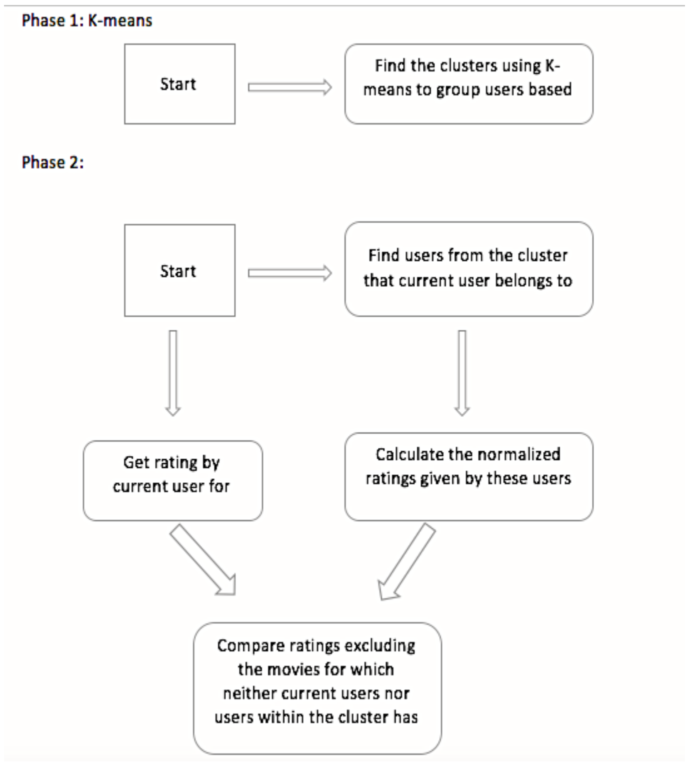
K -means clustering is a vector-quantization method, popular for cluster analysis. The observations are divided into K clusters ,where each observation belongs to cluster with closest mean. The choice the number of clusters is made starting with $K = 2, 3, 4, ...n$ until suitability. In order to determine this suitability, we have used **Elbow Method** which tries to define clusters such that total intra-cluster variation (or total within-cluster sum of square(WSS)) is minimized.



The basic idea is to divide the observations into clusters using K-means . Now, in the implementation for each active user, we identify the users in the cluster which active user belongs to from the training set. For these selected users, we obtain the ratings given by them to movies in the testing data. For each movie in test data, and for the selected users we normalize the ratings. If the rating given by all these users is zero for any of these movies, we store the overall result as zero itself, in order to finally avoid it in calculations of comparing the actual rating and predicted rating for the active user.

With these actual rating and predicted rating vector for the active user, we skip the rating for any movie whose actual rating or predicted rating is zero as it not useful in comparison.

One potential advantage of this method is that overall solution is improved, as observations are not permanently committed to a cluster over number of iterations in K-means. One probable reason for low F1-score as compared to other models is to base prediction from all users in the cluster, which on average has 755 users. Because, if we chose to consider only 20 random users from the cluster the F1 score is 0.645 an improvement upon the F1 score of 0.583 from before.



Phase 1: K-means
Start
Find the clusters using K-means to group users based
Phase 2:
Start
Find users from the cluster that current user belongs to
Get rating by current user for
Calculate the normalized ratings given by these users
Compare ratings excluding the movies for which neither current users nor users within the cluster has

B. K-NEAREST NEIGHBORS

We implemented the following steps for a target user

- Use cosine similarity to find out how similar a target user's preferences are as compared to another user.Cosine similarity is the cosine of the angle between two n-dimensional vectors in an n-dimensional space. It is the dot product of the two vectors divided by the product of the two vectors' lengths (or magnitudes). For two vectors A and B in an n-dimensional space:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

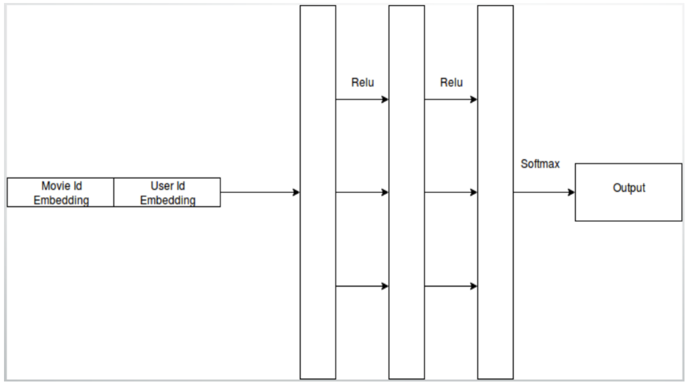
  Cosine similarity ranges between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar.
- Get the top 20 nearest neighbors(Chosen through cross-validation) in the training data having similar preferences to the target user.
- Use these top 20 users to calculate the normalized movie ratings for the hidden set of movies and use them to make predictions.
- Compare predictions with actuals and calculate precision, recall.

We repeat the same steps for each user, and final score is calculated by **F1 Score** based on micro-averaged precision and recall.

C. Neural Network

All the methods so far use a measure of closeness and find the users that are closest to the target user in terms of

preferences, and using ratings of them to predict target users preferences.
In this model we train embeddings for user and movie and use these embeddings as input to the neural network model predict the rating.
We implemented a $Multilayer\ Perceptron$ model containing three hidden layers with 128 hidden units each. Each of these hidden layers have a $Relu$ activation function, we obtain a one hot encoding as output predicting the rating given to the movie by a target user using $softmax$ activation layer
In this task we trained embeddings which are 32 in dimension for our use case.

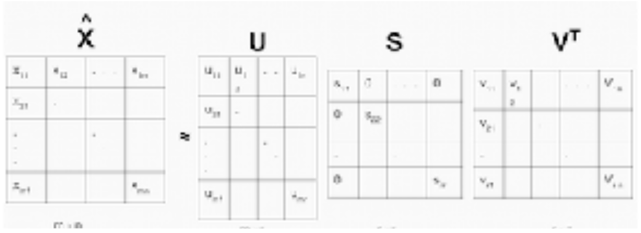


MLP layers

D. SINGULAR VALUE DECOMPOSITION

Singular Value Decomposition is one of the matrix factorization algorithms used for collaborative filtering. This type of algorithms finds the latent features of users and movies, and makes the predictions based on these factors.It reduces the number of features of a data set from N to K where $K < N$.The value of k may vary. The reduced matrix X is built by considering only the first k singular values.The U_k matrix represents the feature vectors corresponding to the users in the hidden feature space and the V_k matrix represents the feature vectors corresponding to the items in the hidden feature space. Multiplying these three reduced matrices, the matrix $\hat{X}$ is obtained. The reconstructed matrix $\hat{X}$ is the closest approximation to the original matrix .

$$\hat{X} = U_k * S * {V_k}^t$$



Each item can be represented by a vector q_i and each user can be represented by a vector p_u such that the dot product of those 2 vectors is the expected rating

$$expectedrating = \hat{r_{ui}} = {q_i}^T.p_u$$

In order to minimize the difference between the predicted rating and actual rating,we also add some bias terms to the predicted rating. For each user-item (u, i) pair we can add 3 parameters. μ which is the mean ratings of all items, b_i which is the mean rating of item i minus μ and b_u which is the mean rating given by user u minus μ which makes the expected rating:

$$\hat{r_{ui}} = {q_i}^T.p_u + \mu + b_i + b_u$$

To make our model a generalized one and no to over fit the training set of data, we add a penalty term to our minimization equation. This is represented by a regularization factor multiplied by the square sum of the magnitudes of user and item vectors.

$$min(p, q) \sum_{u,i \in K} (r_{ui} - {q_i}^T.p_u)^2 + \lambda((\| (q_i) \|)^2 + (\| (p_u) \|)^2)$$

Thus, the final equation to minimize is:

$$min(p, q, b_i, b_u) \sum_{u,i \in K} (r_{ui} - {q_i}^T.p_u - \mu - b_i - b_u)^2 + \lambda((\| (q_i) \|)^2$$

$$+(\| (p_u) \|)^2 + {b_i}^2 + {b_u}^2)$$

Thus the above equation is minimized using the stochastic gradient descent approach.SGD starts by initializing the values of the parameters and then iterating to reduce the error between the predicted and the actual value each time correcting the previous value by a small factor .

Cross validation is used to evaluate the accuracy of the predicted rating. The data is split into folds and the algorithm is trained on few folds and tested on the remaining folds of data. The actual and predicted ratings are classified into user-liked movies and user-disliked movies by keeping a threshold of 4. Based on the classification we obtain a precision of 0.8592, recall of 0.5230, and $F1$ measure as 0.6502 are obtained.

IV. EVALUATION METRICS

In order to evaluate the performance of each algorithm, we used $Precision$, $Recall$ for each user. We used Macro-average Precision, i.e average precision over all users as an aggregate measure, similarly we use macro-average recall. We calculate $F1score$ over these aggregate measures.

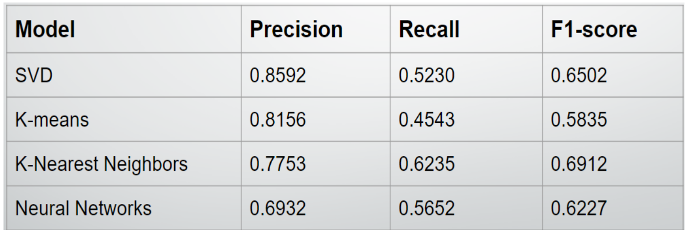
$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

V. RESULTS

Below are the Precision, Recall, F1 scores of all our models:



Model	Precision	Recall	F1-score
SVD	0.8592	0.5230	0.6502
K-means	0.8156	0.4543	0.5835
K-Nearest Neighbors	0.7753	0.6235	0.6912
Neural Networks	0.6932	0.5652	0.6227

VI. CONCLUSION

As it can be seen, K-Nearest Neighbors outperforms all other models, because it looks for users that are closest to our target user in terms of preferences of movies. Although, K-Means algorithm has a similar fashion of finding closest users, it under performs in comparison because its scope is global whereas scope of K-NN is local. Although SVD gives a competitive result, it is computationally expensive for large datasets. Neural network model on the other hand has a very different approach compared to these models so it is not fair to strictly compare with other models. Neural Network model can be further improved by increasing training data, and further including genres of movies as it can be quite flexible in terms of its input.
This project helped us in improving our understanding of using machine learning methods for a complicated task like movie recommendation, we also developed insights into reducing sparsity of the dataset by considering a carefully selected subset of features.

REFERENCES

[1] MovieLens dataset
https://grouplens.org/datasets/movielens/

[2] Neural Network
http://blog.richardweiss.org/2016/09/25/movie-embeddings.html

[3] Matrix Factorization
https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

[4] K-Means
https://rpubs.com/tim-dim/recommender

[5] K-NN
https://www.youtube.com/watch?v=kkM89uW1rJ8

[6] SVD
Singular Value decomposition (SVD) in recommender systems for Non-math-statistics-programming wizards by Maher Maleab

TASKS DONE BY GROUP MEMBERS

Rohit - Trained and implemented neural networks model, prepared project report and presentation
Ramkishan - Created initial super matrix, performed data analysis to reduce number of features, trained and implemented knn model, prepared project report and presentation
Apoorva - Performed data analysis, trained and implemented k-means model, prepared project report and presentation
LakshmiManaswitha - Trained and implemented SVD model, prepared project report and presentation