IDS Project Report

I. Introduction

In this project we aim to implement a Collaborative Filtering process grouped with clustering methods in order to achieve lower RMSE as well as MAE values along with that we also aim to reduce the computation time required to calculate the rating for the movies. This method is implemented in a research paper (Mark O'Connor & Jon Herlocker, 1999) where the authors have provided results showing lower MAE values when compared with Collaborative Filtering without clustering methods. Here in this paper we also consider the time required for execution of the program as one of the metric. In the above mentioned research paper clustering algorithms such as hMetis, kMetis, Average Link and ROCK have been used. Contrary to them we have implemented algorithms such as K-Means, Agglomerative and MiniBatch K-Means to produce similar results.

II. Prior Work

Such type of experiments where Collaborative Filtering is assisted with clustering were reported in the following paper : *O'Connor, M., & Herlocker, J. (1999, August). Clustering items for collaborative filtering. In Proceedings of the ACM SIGIR workshop on recommender systems (Vol. 128). UC Berkeley.* In this research paper, the authors have experimented on MovieLens 100k database where there are 100,000 ratings provided by 943 unique users to 1682 unique items. By doing so the following outcome was observed, based on these observations they concluded that the execution time decreases but along with that so does the accuracy of the model. According to the writes kMetis provided the most promising result in terms of accuracy and execution time.

Method	MAE	Coverage
Unpartitioned base case	0.7594	99
Random	0.8211	74
Genre	0.7806	87
Average Link k = 50	0.7754	61
hMetis k = 5, Ubfactor = 10	0.7859	79
hMetis k = 5, Ubfactor = 1	0.7951	82
kMetis k = 5	0.8033	79

III. Methods

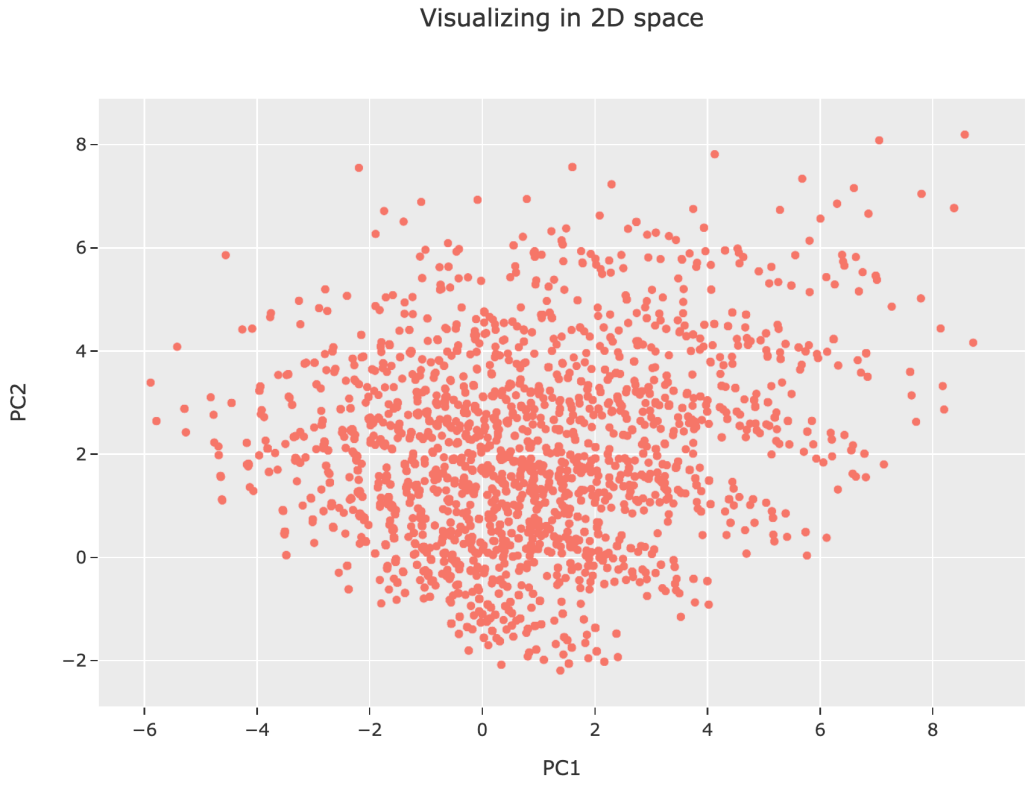
i. Data Preprocessing and Visualisation :

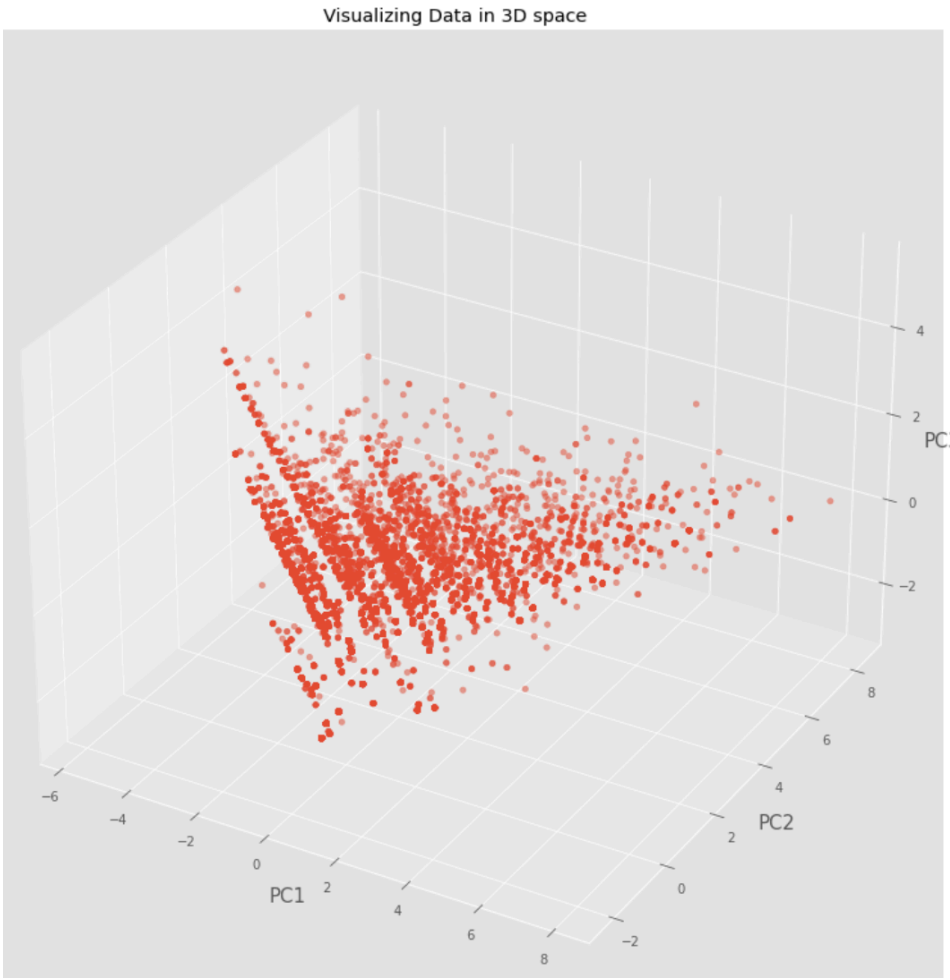
In this project we first tried to use MovieLens 25M Dataset (https://grouplens.org/datasets/movielens/) which consists of over 25M ratings from 1,62,000 unique users to 62,000 movies. We first wanted to visualise the data to understand whether we can find any patterns in the data. For this purpose we tried to visualise data associated with movies to see whether there are any visible cluster present in them. The reason why we were visualising movies data and not user

data is because further ahead we were going to perform item based collaborative filtering and not user based filtering. For the movies data we preprocessed the data in a way that the number of columns is equal to the number of unique genres present and the number of rows is equal to the number of unique movies present in the table. The cell value is 1 if a movie has that particular genre associated with it or else it is a 0. The processed data frame is as follows :

	IMAX	Crime	Sci-Fi	Western	Thriller	Adventure	Musical	Documentary	Fantasy	Horror	Mystery	Action	Animation	Children	Drama	Romance
1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0
2	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Indexes of these rows are nothing but the id's of the movies. Post this we have scaled the data as there are multiple cell values with zeroes and we had to make sure that this does not create a problem while visualizing the data. To reduce the dimensionality of the data we had two option one was using **PCA** and another was by converting the data matrix into respective **SVD**. For this data we have converted it into 2 dimensions and 3 dimensions in order to visualise data in 2d and 3d space. The following was produces when we plotted the data :

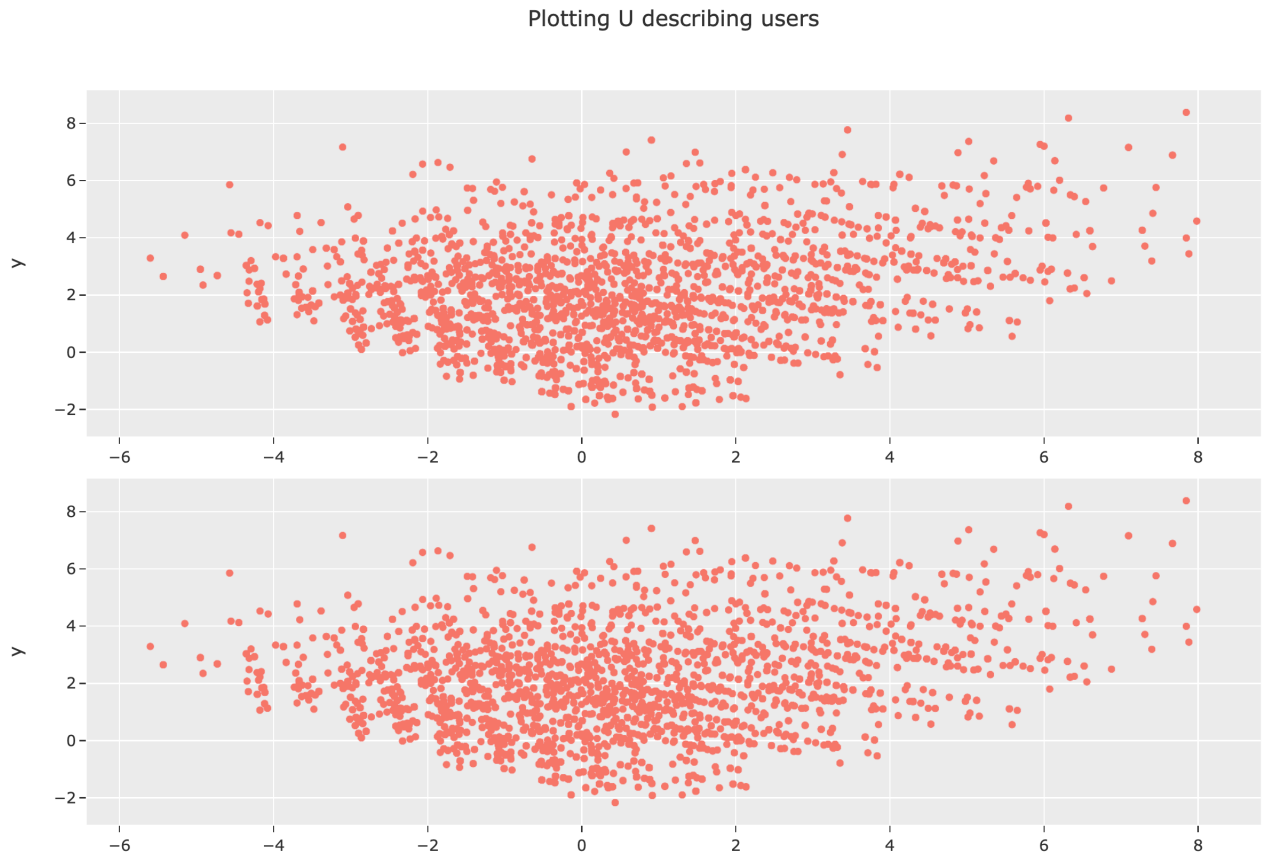




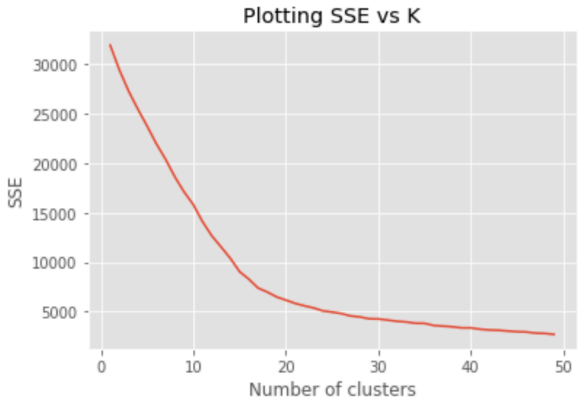
In order to compute the SVD we fist have to compute matrix with the users as rows and all the movies as columns wherein the cell values are the rating given by the user to that particular movie. In order to do so we had to iterate over all the 25M lines and create the matrix, this process was computationally very heavy and required a lot of time hence we decide to proceed ahead with a smaller subset of MovieLens Dataset which is the 100K dataset consisting of 100K ratings provided by 943 unique users to 1682 unique items. For this data set we computed the data matrix where rows depict users, columns depict movies(items) and cell value is the rating given by that particular user to the movie. Since there were many user-movie pairs with no rating we have assigned the average rating of the movie to that particular cell. After computing SVD we get 3 matrix U, sigma and V where U and V depict the users and items respectively. When visualised we get the graphs displayed on the next page.

ii. Clustering Movies Based on Genres

Post visualisation we continue with clustering the movies data, based on the genres as this was the only feature in the dataset which was most appropriate to cluster the data with there are 19 unique genres associated with each movie and



the cell value indicates whether the movie is associated with that particular genre or not. Here we have used 3 types culturing methods, K-Means, MiniBatch K-Means and Agglomerative. For K-Means in order to find the optimal K we make use of elbow method by which we get K = 17 as the ideal cluster number.



We now used K mean to generate 17 clusters. For simplicity purposed we kept the number of clusters same across all the cluttering algorithms.

K Means - This clustering is a method of vector quantisation that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

Mini Batch K Means - The Mini-batch K-means clustering algorithm is a version of the standard K Means algorithm in machine learning. It uses small, random, fixed-size batches of data to store in memory, and then with each iteration, a random sample of the data is collected and used to update the clusters.

Agglomerative - In this method each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

iii. Item Based Collaborative Filtering w/o Clustering

Here we have implemented Item Based Collaborative filtering to compute rating for a particular movie using the following formula, here in this formula we make use of similarity matrix which is computed using cosine similarity formula which is computed by the given formula. Along with that we also make use of the mean ratings for movies as well as for users while computing ratings of a particular movie .

$$r_{u,i} = \bar{r}_i + \frac{\sum_{j \in N(i)} sim(i,j)(r_{u,j} - \bar{r}_j)}{\sum_{j \in N(i)} sim(i,j)}$$

$$sim(U_i, I_j) = cos(U_i, I_j) = \frac{\sum_{l=1}^{k} u_{i,l} i_{j,l}}{\sqrt{\sum_{l=1}^{k} {u_{i,l}}^2} \sqrt{\sum_{l=1}^{k} {i_{j,l}}^2}}$$

iv. Item Based Collaborative Filtering with Clustering

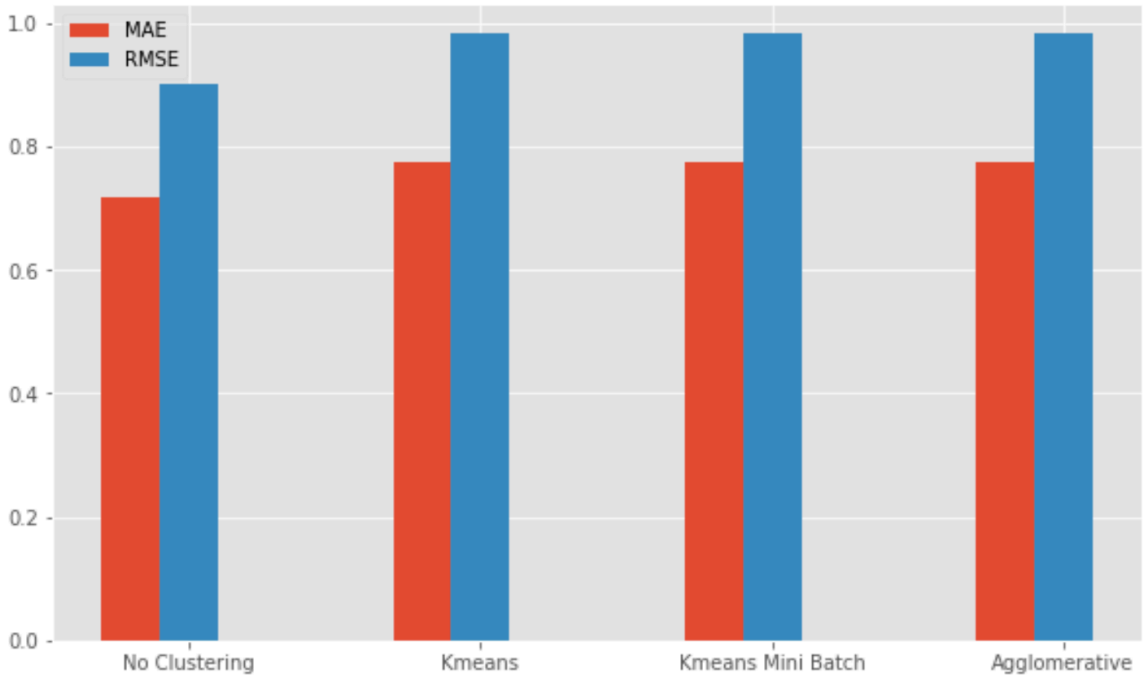
In this method also we make use of above mentioned formulas but instead of looking through the similarity matrix of all the items, we just focus on the one particular cluster which contains our queried item in it. Rest everything stays the same.

IV. Results and Findings

We have focussed majorly on the computation time and the accuracy. To measure accuracy we make use of two evaluation methods MAE(Mean of Absolute Error) and RMSE(Root of Mean Squared Error). The formulas for both are as follows:

$$MAE = \frac{\sum_{ij} |\hat{r}_{ij} - r_{ij}|}{n} \qquad RMSE = \sqrt{\frac{1}{n} \sum_{i,j} (\hat{r}_{ij} - r_{ij})^2}$$

By computing these both values we can plot the following graph:



Also we kept a tab over the computation time and got the following result :
Without Clustering :

```
100%|██████████| 20000/20000 [08:11<00:00, 40.65it/s]

CPU times: user 6min 54s, sys: 39.6 s, total: 7min 33s
Wall time: 8min 11s
```

With all 3 types of clustering:

```
100%|██████████| 20000/20000 [01:49<00:00, 183.24it/s]
100%|██████████| 20000/20000 [01:49<00:00, 182.46it/s]
100%|██████████| 20000/20000 [02:07<00:00, 156.37it/s]

CPU times: user 8min 51s, sys: 58.9 s, total: 9min 50s
Wall time: 5min 46s
```

V. Conclusion

From the findings given above we can reach to a conclusion that using clustering methods does increase the RMSE as well as MAE in turn reducing accuracy. But at the same time, computation time is reduced considerably. As we can see that if we perform filtering without clustering it takes about **8 minutes** but if we do the same by utilizing clustering algorithm it takes around **3 minute** per algorithm.

Even though there is huge difference in the execution time, the accuracy does not change much. Here we can say that if there is a huge dataset then it is advisable to first cluster it and then apply filtering over the same, this will reduce computation time significantly. Also we need to take into account the time required for clustering, here even if perform clustering once we can use the same labels again and again, hence clustering algorithm will be executed only once. Said that, the experiments performed here have similar results as that of the experiments performed by the authors in the above mentioned paper.