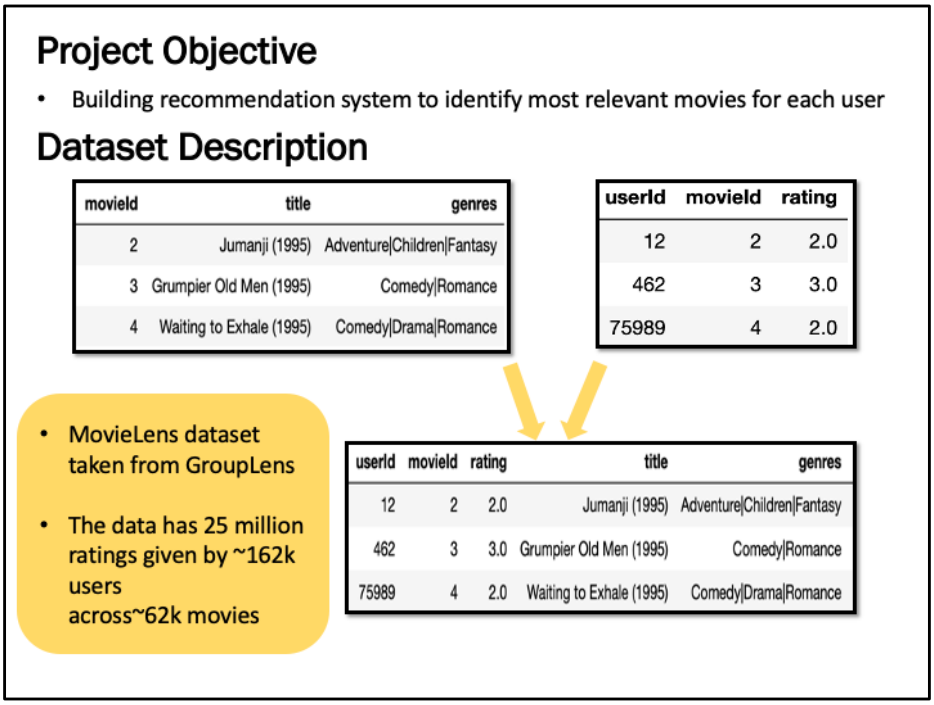
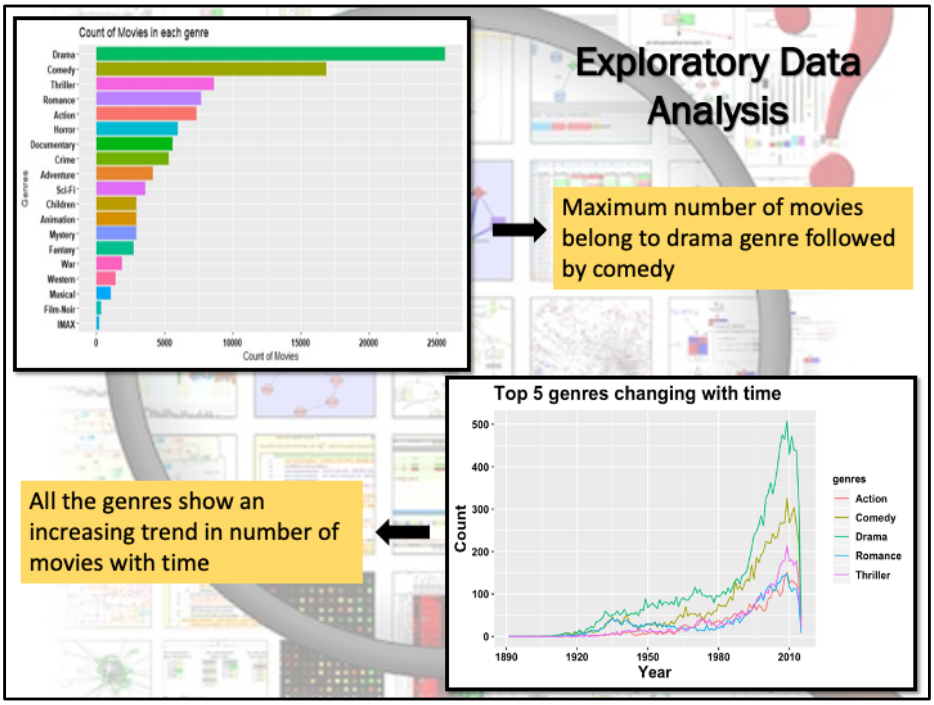
MOVIE RECOMMENDATION SYSTEM

DERRIE SUSAN VARGHESE
SAYAN BISWAS
SNEHA AGARWAL
VARUN JAGADEESH



Objective:
Our goal of the project is to recommend new movies to users, to dig out what users may like but they did not know before. We aim to build a movie recommendation system by exploring different approaches such as market basket, collaborative filtering, clustering, etc.

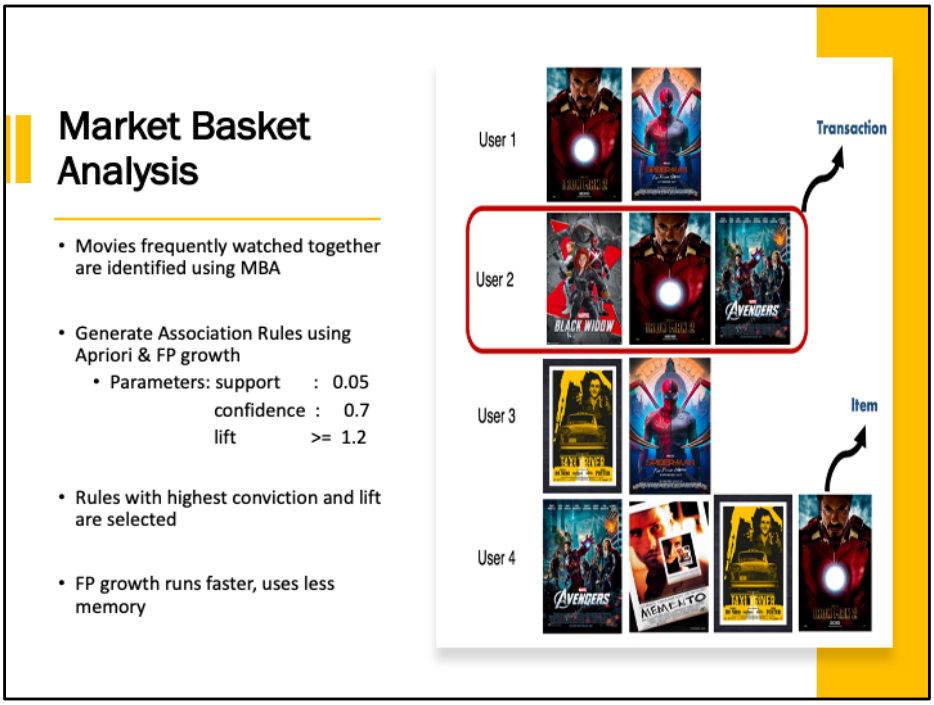
Dataset description:
The data is MovieLens dataset is taken from GroupLens website.
The dataset is spread across multiple csv files. As we can see, one csv file has information about movies like the movie Id, movie title and the genres the movie belong to. Another csv file has information about the user and the ratings the user has given to the movie. In order to make maximum use of the data we have at hand we have merged all the tables together and what we see is a snapshot of what the merged data looks like.



Exploratory Data Analysis

We did some initial analysis to get some insights on what raw data can tell us.

The first graph here shows the number of movies in each genre. We can see here that that the maximum number of movies are made in Drama genre, followed by Comedy. The least number of movies are made in Musical, Film-Noir and IMAX category. This also hints about what kind of movies are preferred and watched by movies. The second graph shows the variation in genres with time. We can see here that every genre has seen increase in number of movies produced with time. The biggest leap is shown by Drama genre and the least by Action genre.



Market Basket Analysis

Market Basket Analysis is one of the first method we used to discover movies that are frequently watched together. First we built the required association rules on a set of transactions; second, we deployed the rule engine to generate recommendations for new basket data and/or new transactions.

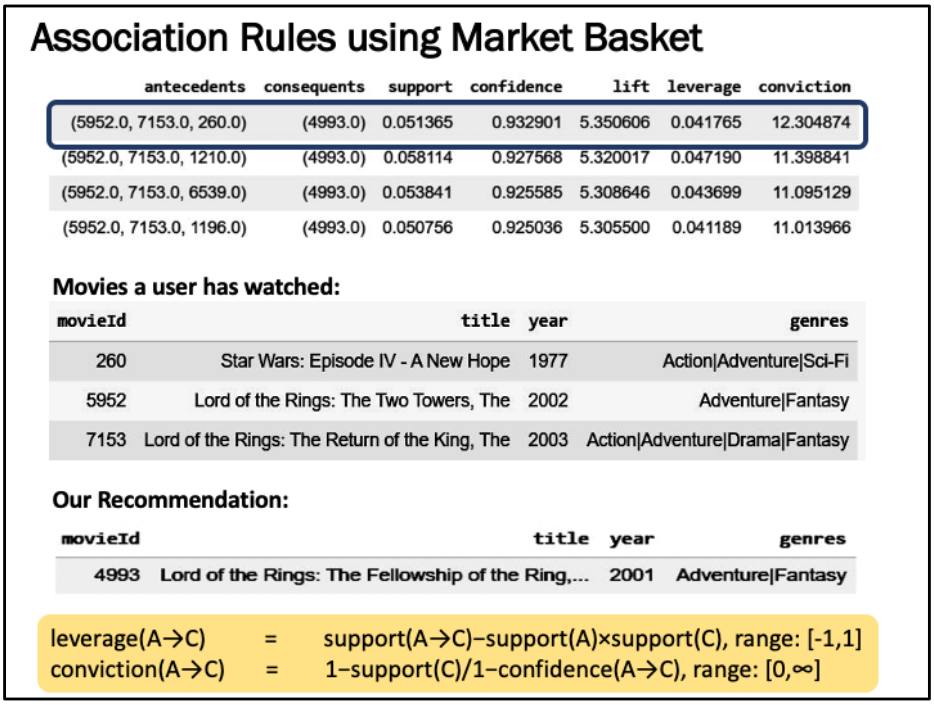
Blueprint of Transactions :

a. Each Transaction ID is User ID b. Movie IDs are the items in the basket c. Movies were considered in the basket, only if they were rated equally or higher by the user compared to the average movie rating.

Association rules are generated using Apriori and FP growth. On a sample set of the data both the algorithms perform decently and gave same results.

Parameters used: We set the support threshold at 0.05, confidence at 0.7 and lift greater than or equal to 1.2.

We saw that the Apriori algorithm uses a lot of memory during the candidate generation process and thus runs very slowly. Whereas, FP growth runs much faster and uses less memory. Therefore the association rules on the entire dataset were generated using FP growth.



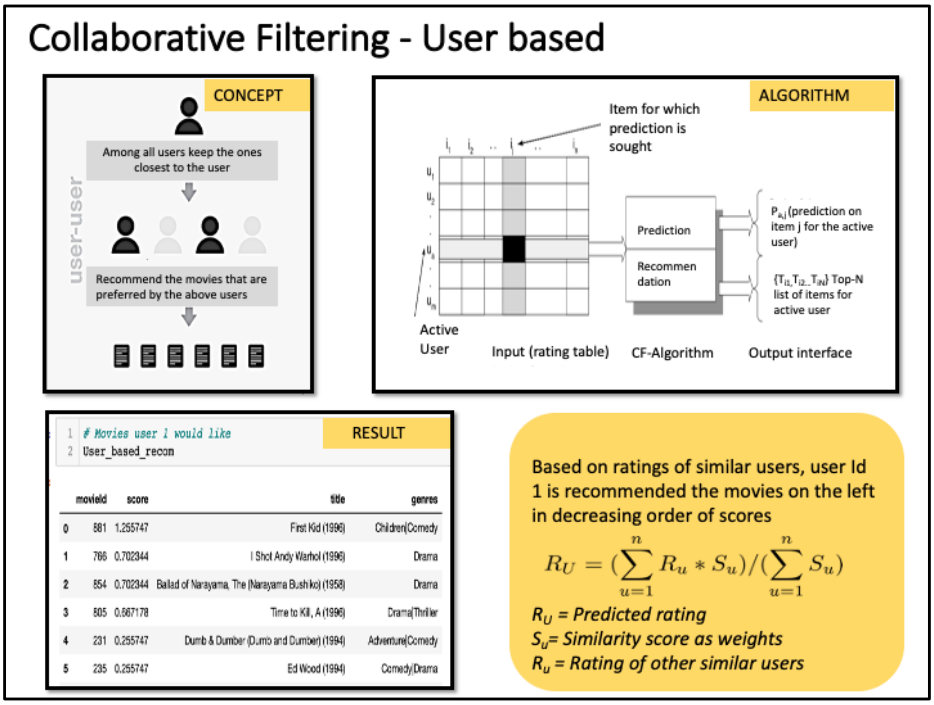
Association rules results

The first part of the rule is called “antecedent”, the second part is called “consequent”. A few measures, such as support, confidence, and lift, define how reliable each association rule is.

Here the antecedent corresponds to the "movies watched by the users" and consequent of the rule is used here to recommend the movies to user they have not already watched.

Leverage computes the difference between the observed frequency of A and C appearing together and the frequency that would be expected if A and C were independent.
A high conviction value means that the consequent is highly depending on the antecedent.

The rules with the highest conviction and lift are considered to recommend movies.
Sample recommendation: If a user has watched movies with movie ID 5952 and 7153 but not 4993, we can recommend him to watch movie ID 4993.



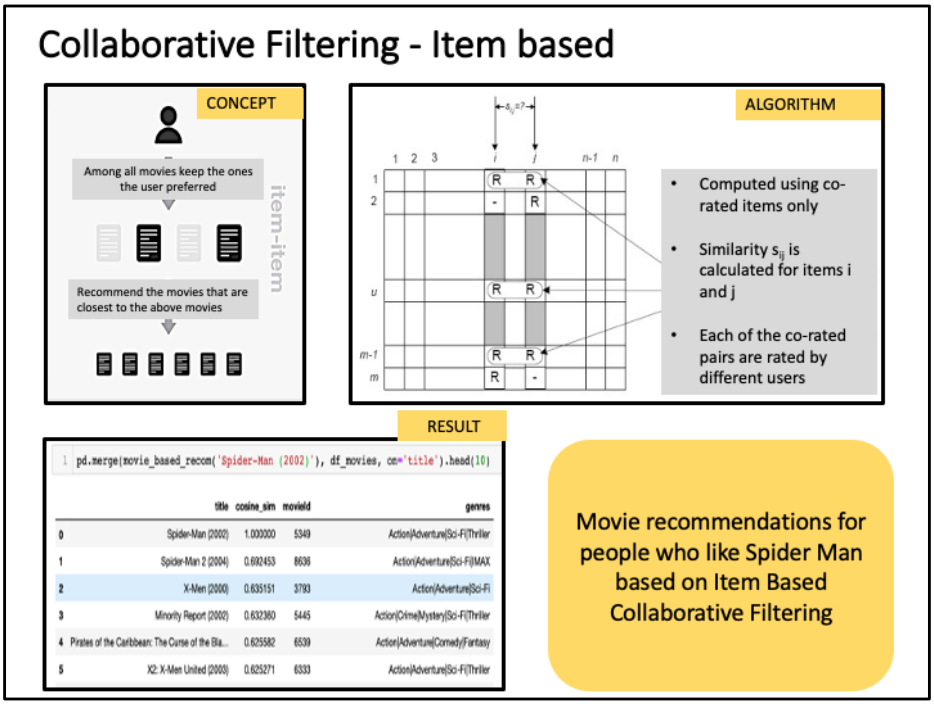
Collaborative Filtering - User based

It is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. In user-based filtering, the rating of a movie M which hasn't been rated by user U, is calculated by the ratings of other similar users who have rated movie M. For this we predict the rating R, that user U would give to a movie M. If the rating is above a threshold, we recommend the movie to the user.

To find the rating R that a user U would give to a movie M, the approach is:

- We generate the User-Item matrix.
- Find users similar to U who have rated the movie M by first removing the rating bias of each user by subtracting the user's mean rating from all their ratings, and then computing Cosine Similarity to find similar users.
- Once we have similar users, we calculate the rating R the user U might give; based on the weighted average rating of similar users.

We multiply each rating by a similarity factor. By multiplying with the similarity factor, we add weights to the ratings. The heavier the weight, the more the rating would matter.

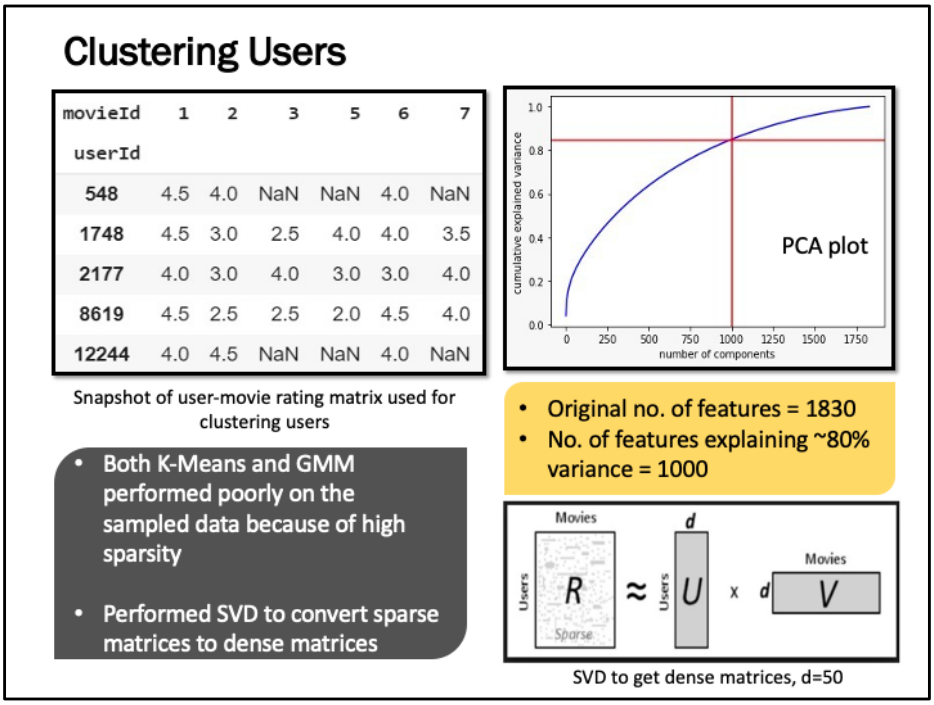


Collaborative Filtering - Item based

We identify similar items based on users' previous ratings and then give recommendation. For ex.: Consider 3 users, 1, 2 and 3 who have given a 5 star rating to movies Iron Man and Spider Man. When a new user watches the movie Spider Man the user gets recommendation to watch Iron Man as well. This is because the system identifies Iron Man and Spider Man to be similar based on the ratings given by the users 1,2 and 3.

We can see a matrix where the rows represents users and the columns represents items (movies in our case). A similarity is being calculated between two items i and j and it done only for co-rated items. At first, we are isolating the users who have rated those two items (i, j) and then we are computing similarity between them. We are using cosine-based approach to calculate the similarity between two items. As we can see in the result, those are some of the movie recommendations for people who like Spider Man with a cosine similarity score. The higher the score the more similar the movies are.

Item-Based approach performs better than **User-Based approach** because avg rating received by an item doesn't change as quickly as avg rating given by a user to different items.



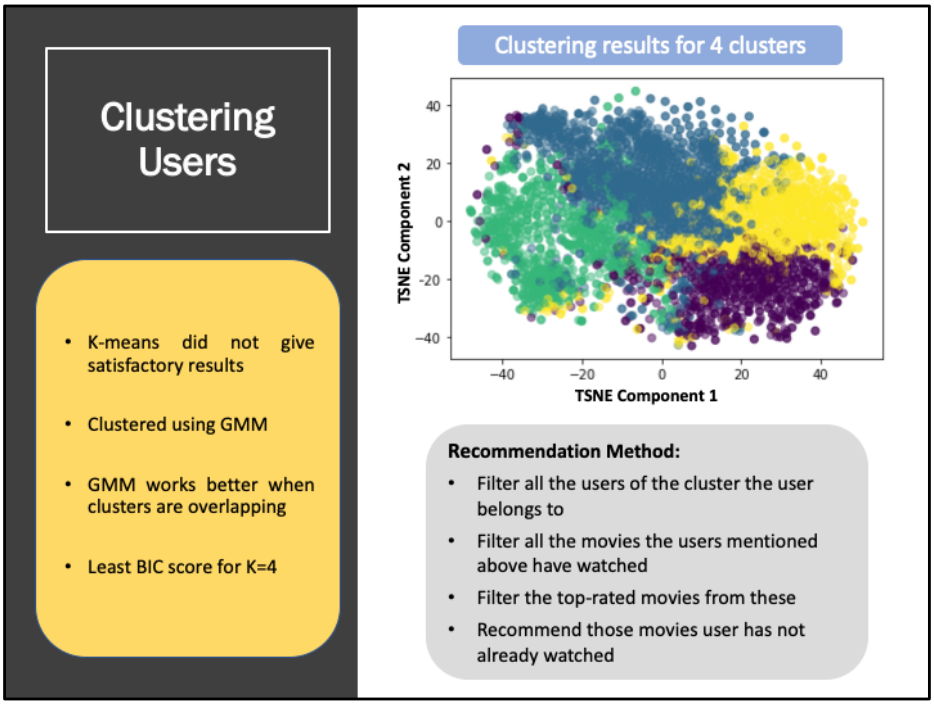
Clustering Users

In this approach, we clustered users based on movie ratings. A pivot table was created with user IDs as rows, movie IDs as columns and the user ratings as the pivot values The bias in the movie ratings of each user was removed by subtracting their mean rating from all their ratings, NAs were then replaced with zeroes.

PCA: Dimensionality reduction is highly recommended while performing clustering since it suppresses noise and increases the speed of computation while calculating pairwise distances between datapoints. The original number of features were 1830. As we can see in the plot on the top right 1000 features are chosen to explain 80% variance using PCA.

Clustering: We implemented K-Means clustering on the data(K=2 to 15), however we got very low silhouette score(close to 0) and the optimal number of clusters as 2. We then implemented Gaussian Mixture Model (GMM), however this technique also performed poorly due to the high sparsity in the data.

Singular Value Decomposition: To solve the issue of sparsity, SVD was done on the sparse matrix resulting in two dense matrices of users and movies

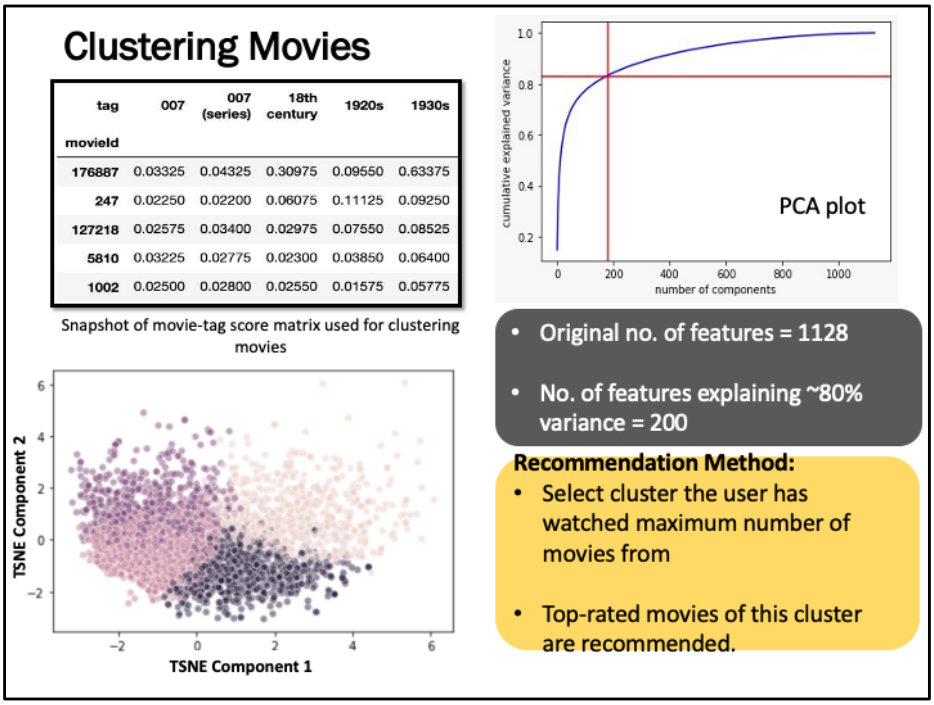


Clustering Users

After implementing SVD, K-Means clustering did not provide the results we expected because of high overlap between the data points. Whereas, GMM, which is known to perform better on such data, gave better results. For GMM, least BIC score was obtained for four clusters.

For the plot on the top right, TSNE was used for visualization . This scatter plot is the visualization for four clusters in GMM. As we can see, there is high overlap between the clusters.

Recommendation method: To recommend movies for a user, initially all the users belonging to the same cluster as the given user is filtered. After which, all the movies these users have watched are obtained and from this pool of movies the top-rated movies are selected. These movies are then recommended to the given user.



Clustering Movies

In this approach, we clustered movies based on tag relevance scores (genome scores). A pivot table was created with movie IDs as rows, tags as columns and the tag scores as the pivot values. Tags here are words or phrases that users have used to describe a movie. The scores were normalized using Min-max scaling.

Principal Component Analysis: We performed PCA here as well to reduce the number of features . The original number of features were 1138. As we can see in the plot on the top right, to obtain around 80% of explained variance, 200 features are chosen.

Clustering: We implemented Gaussian Mixture Model (GMM) to cluster movies. GMM worked well here and we achieved the lowest BIC scores for K=4 and the clusters formed when visualized using tSNE.

Recommendation method: To recommend movies for a user, firstly we select the cluster that the user has watched maximum number of movies from. Then we select the movies with the highest average rating for recommendation.

Clustering Results for Recommendation

```
x=recommendation(1652)
x['user_watched']
```

	movieId	title
52395	593	Silence of the Lambs, The (1991)
167670	195159	Spider-Man: Into the Spider-Verse (2018)
195394	16	Casino (1995)
541703	4432	Sweet Smell of Success (1957)
747819	48516	Departed, The (2006)
794633	4262	Scarface (1983)
1362366	1653	Gattaca (1997)
1483612	1259	Stand by Me (1986)
1755348	81932	Fighter, The (2010)
1958766	74545	Ghost Writer, The (2010)

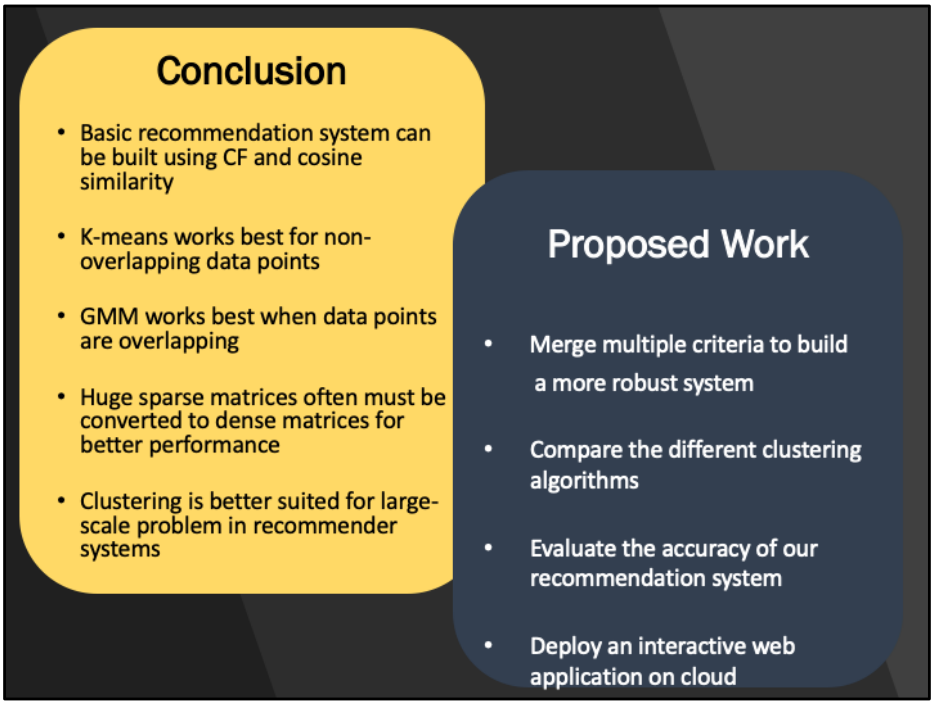
Movies user ID 1652 has watched

```
x['movies_recommended']
```

	movieId	title
1679066	858	Godfather, The (1972)
4564028	8183	Educating Rita (1983)
5565177	4292	Norma Rae (1979)
6580544	50	Usual Suspects, The (1995)
1340639	3467	Hud (1963)
151051	1235	Harold and Maude (1971)
1640850	858	Godfather, The (1972)
1861245	4226	Memento (2000)
6699104	79132	Inception (2010)
551467	91529	Dark Knight Rises, The (2012)

Movie recommendations for user ID 1652

Here we can see the recommendation results. We randomly choose a user, in this case User ID 1652. We see on the left the movies this user has already watched. On the right are the movies we recommend this user to watch after running the clustering algorithm. So a user who has watched movies like "Silence of the Lambs", "Spider-Man" is recommended movies like "Inception", "Godfather", etc.



Conclusion

Cosine similarity and collaborative filtering are some of the useful techniques to build recommendation systems. It however fails to solve the cold start problem for new movies that are added to the list. We found that K means works best when data points are non-overlapping since it easily gets stuck in local minima where as GMM is best when the data points are overlapping and can assign a data point to multiple clusters with some probability. Clustering with very sparse data is not recommended as it yields poor clusters due to high number of dimensions. It is also highly inefficient as it uses a lot of computational power and memory. Instead, the sparse matrix could be converted to dense using say SVD(Singular Value Decomposition).

Proposed Work

Our approach to recommend movies would not help to solve the cold start problem where we do not have any user preference so instead of recommending movies using one criterion, we can merge multiple criteria and build a more concrete and robust recommendation system. Also, to better measure the efficiency of the system, we aim to compare the different clustering algorithms and evaluate the accuracy of our recommendation system. Deploying this recommender system to an interactive web application on cloud will help customers to easily get movie recommendation.