# Movie Recommendation System

**Team Members:** Derrie Susan Varghese, Sayan Biswas, Sneha Agarwal, Varun Jagadeesh

**Aim:**
To build a movie recommendation system using the MovieLens dataset, which has 25 million records of user-given ratings for various movies. We are primarily using Market Basket Analysis and Cosine Similarity techniques to recommend movies during this phase.

**Exploratory Data Analysis:**
We started with tidying the data using R for this part:

- The movie genres column which had values like “Action | Comedy” were separated into separate rows.
- The year was extracted from the movie title column and placed in a separate column.

From this extensive dataset, we were able to answer questions like:

1. Variation of count of movies across genres and with time, as can be seen in Figure. 1 and 2. ‘Drama’ has the maximum number of movies overall followed by Comedy. The same categories show a huge increment in the number of movies produced in recent years.
2. Most appropriate and popular tags for each genre.

We could answer all the EDA questions as proposed in the abstract. We then started exploring the idea of recommending movies using Market Basket Analysis and Cosine Similarity.

**Market Basket Analysis:**
Our first approach was to consider each user id as the transaction id and the movies watched by the user as the items in the basket. We used the apriori package under mlxtend(library in Python) to generate the association rules. Due to the high volume of transactions and the number of items in each transaction the system was crashing as apriori reads everything in-memory. Hence, we used the FP growth algorithm to generate the rules. The rules found using this naive approach were not very significant (upon manual verification).

We then decided to incorporate the user ratings for each movie to come up with meaningful association rules. We first filtered for movies that were rated by at least 20 users. We then found the mean user rating for each movie and considered only those movies in the basket which the specific user has rated equally or higher than the mean user rating. This approach gave us much more meaningful rules as can be seen in Figure. 3. A high conviction value means that the consequent is highly dependent on the antecedent and the antecedents and consequents in Figure. 3 are the movie Id(s).

**Cosine Similarity:**
Our main idea here was to group similar movies/users/genres together to be used later for recommending movies.

Initially, we created a movie-user matrix where each row represents unique movies and each column represents a unique user in the dataset who has rated the movie. The values in the cell represent the rating. Each row was treated as a vector and cosine similarity was used to find the likeness between each movie. This was coded using the sklearn package in python. Using this technique, if a user has watched a certain movie, we can then recommend multiple other movies which have high cosine similarities to the given movie. Figure. 4 is a snapshot of the table with the cosine similarities for movies. To find similarity between two movies, we treat them as vector and find the cosine angle between them using the formula:
$\cos(a,b) = a.b/(|a|*|b|)$

We also tried to recommend movies by taking into account the genre of the movies, but we failed because of the large amount of data as each movie belonged to multiple genres and the matrix grew very large. However, based on your input in the last update we will try to sample the data and rerun our tests. We are also aiming to run the code on GCP to avoid frequent memory issues.

**Future work:**
We would want to try to run Market Basket analysis again after filtering for movies with rating 4.0 and above. Given the extensive and voluminous dataset, we are hoping this would give us the best result of all the approaches.

Furthermore, we will try to recommend movies based on a combination of a user's experience and experience of other users. We are aiming to use cosine similarity to first find similar users and then either filter or use clustering to recommend movies to these users.

Our major focus for the 2nd milestone is to use Clustering techniques such as Probabilistic clustering, Hierarchical clustering, etc. With this, we are aiming to create clusters of similar movies, so if a user has watched the majority of their movies from one cluster, we can recommend the next movie from the same cluster or the nearest neighborhood cluster. We would also try creating clusters of similar users so any user from a cluster can be recommended movies which the other users in the clusters might have watched and rated higher.

## Visualizations:

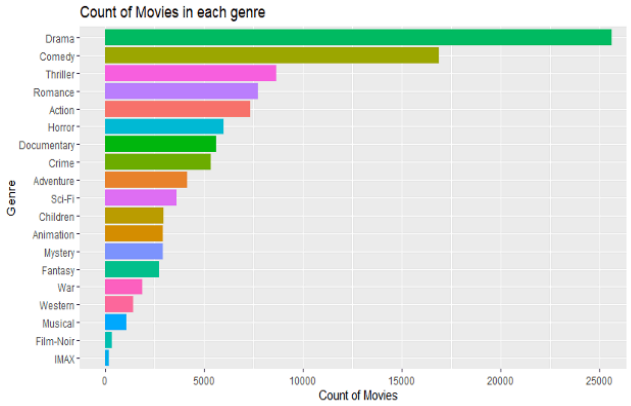


Figure-1:Distribution of movies across genres

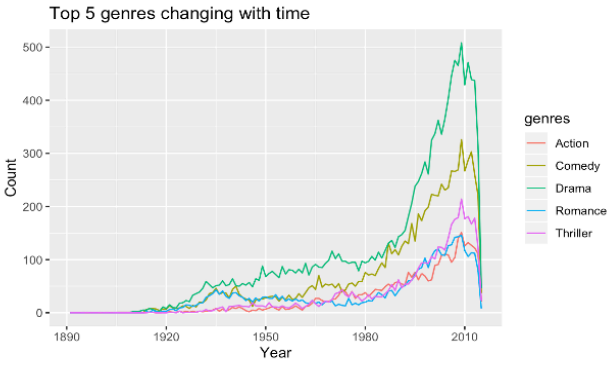


Figure-2:Top 5 genres variation with time

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| **51** | (5952.0, 7153.0) | (4993.0) | 0.124577 | 0.174354 | 0.113038 | 0.907374 | 5.204201 | 0.091318 | 8.913802 |
| **69** | (1221.0) | (858.0) | 0.118217 | 0.195314 | 0.104843 | 0.886869 | 4.540731 | 0.081753 | 7.112874 |
| **50** | (5952.0, 4993.0) | (7153.0) | 0.129220 | 0.160463 | 0.113038 | 0.874774 | 5.451559 | 0.092303 | 6.704156 |
| **52** | (4993.0, 7153.0) | (5952.0) | 0.129916 | 0.154663 | 0.113038 | 0.870089 | 5.625716 | 0.092945 | 6.507031 |
| **46** | (5952.0) | (4993.0) | 0.154663 | 0.174354 | 0.129220 | 0.835496 | 4.791948 | 0.102254 | 5.019015 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **71** | (318.0) | (593.0) | 0.318204 | 0.210055 | 0.111160 | 0.349336 | 1.663070 | 0.044320 | 1.214060 |
| **58** | (318.0) | (2959.0) | 0.318204 | 0.201509 | 0.103544 | 0.325400 | 1.614818 | 0.039423 | 1.183651 |
| **9** | (318.0) | (2571.0) | 0.318204 | 0.226908 | 0.107903 | 0.339100 | 1.494437 | 0.035700 | 1.169756 |
| **19** | (318.0) | (457.0) | 0.318204 | 0.214057 | 0.104005 | 0.326851 | 1.526931 | 0.035891 | 1.167561 |
| **3** | (318.0) | (589.0) | 0.318204 | 0.241938 | 0.107374 | 0.337436 | 1.394717 | 0.030388 | 1.144133 |

76 rows × 9 columns

Figure - 3: Association rules for movies using Market Basket (FP Growth)

```
1 pd.merge(movie_based_recom('Spider-Man (2002)'), df_movies, on='title').head(10)
```

| | title | cosine_sim | movieId | genres |
|---|---|---|---|---|
| **0** | Spider-Man (2002) | 1.000000 | 5349 | Action\|Adventure\|Sci-Fi\|Thriller |
| **1** | Spider-Man 2 (2004) | 0.692453 | 8636 | Action\|Adventure\|Sci-Fi\|IMAX |
| **2** | X-Men (2000) | 0.635151 | 3793 | Action\|Adventure\|Sci-Fi |
| **3** | Minority Report (2002) | 0.632360 | 5445 | Action\|Crime\|Mystery\|Sci-Fi\|Thriller |
| **4** | Pirates of the Caribbean: The Curse of the Bla... | 0.625582 | 6539 | Action\|Adventure\|Comedy\|Fantasy |
| **5** | X2: X-Men United (2003) | 0.625271 | 6333 | Action\|Adventure\|Sci-Fi\|Thriller |
| **6** | Lord of the Rings: The Fellowship of the Ring,... | 0.624800 | 4993 | Adventure\|Fantasy |
| **7** | Lord of the Rings: The Two Towers, The (2002) | 0.617733 | 5952 | Adventure\|Fantasy |
| **8** | Ocean's Eleven (2001) | 0.616562 | 4963 | Crime\|Thriller |
| **9** | Shrek (2001) | 0.613155 | 4306 | Adventure\|Animation\|Children\|Comedy\|Fantasy\|Ro... |

Figure - 4: Movie recommendation using cosine similarity on movies