

# Movie Recommendation System

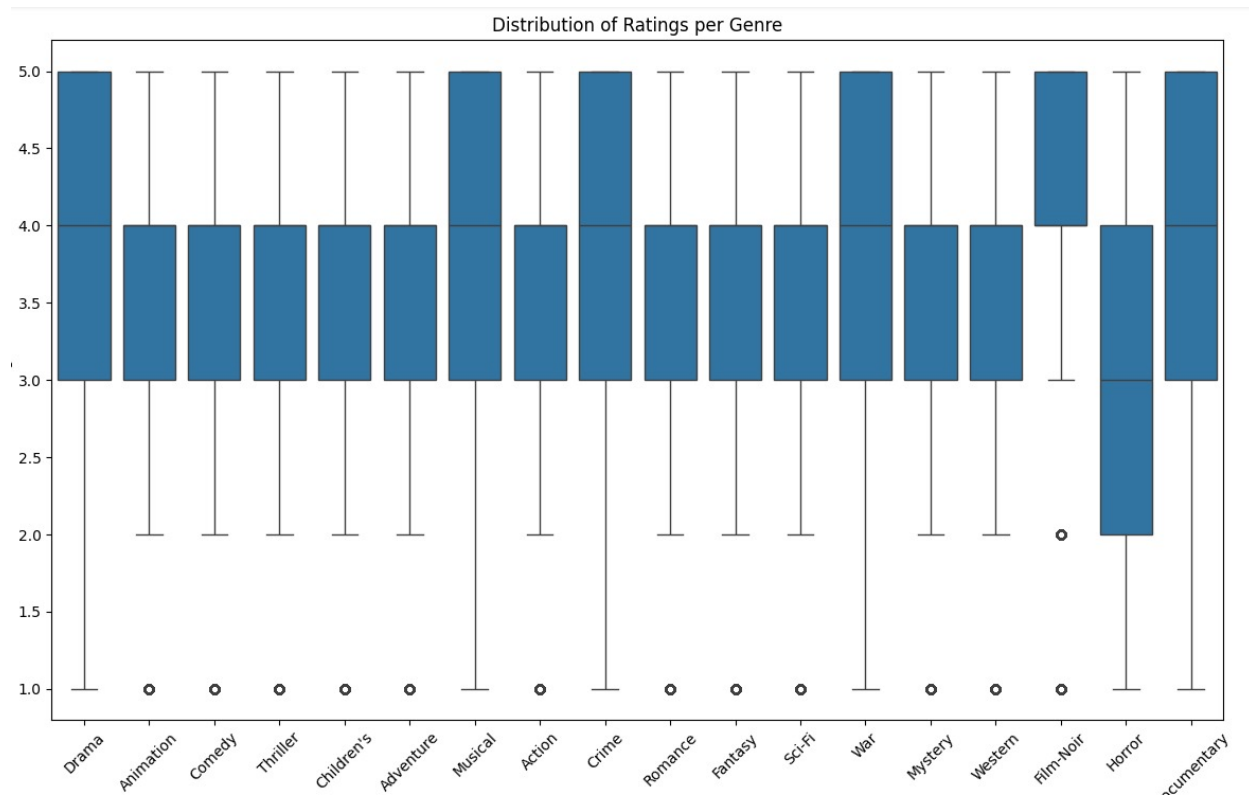
-Aasmaan Gupta (IMT2021006)  
-Ayush Singh (IMT2021092)  
-Mayank Sharma (IMT2021086)  
-Nimish Gaurish Sinai Khandeparkarn (IMT2021077)

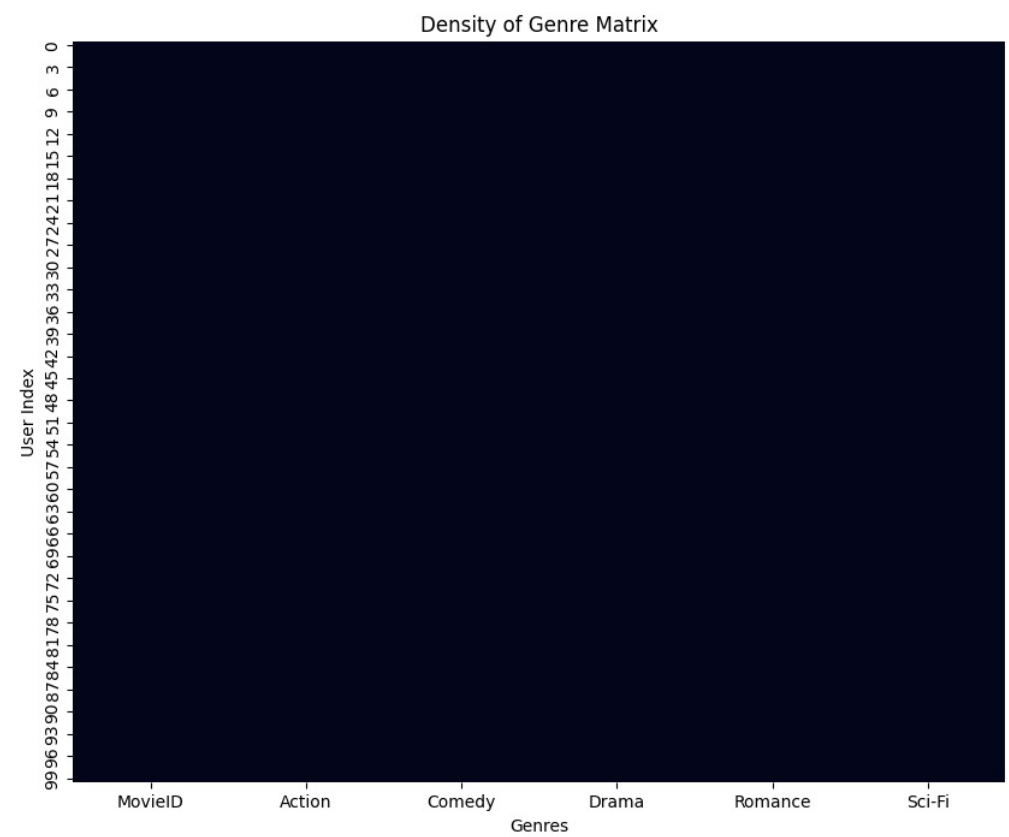
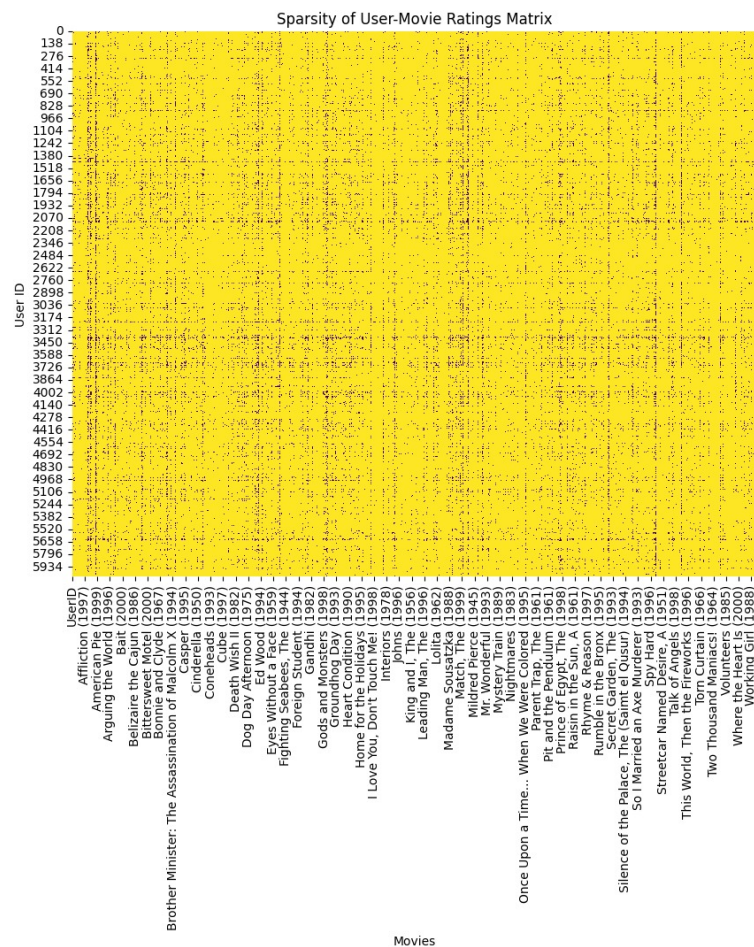
## Preprocessing

We noticed that in our original dataset, some movie ID's were missing so we re-assigned new movie ID's to the movies in a serial number order. We kept a mapping of old and new movie ID's. We used this to reassign movie IDs in ratings.dat file. From this we got new input files that are then used in the rest of the code.

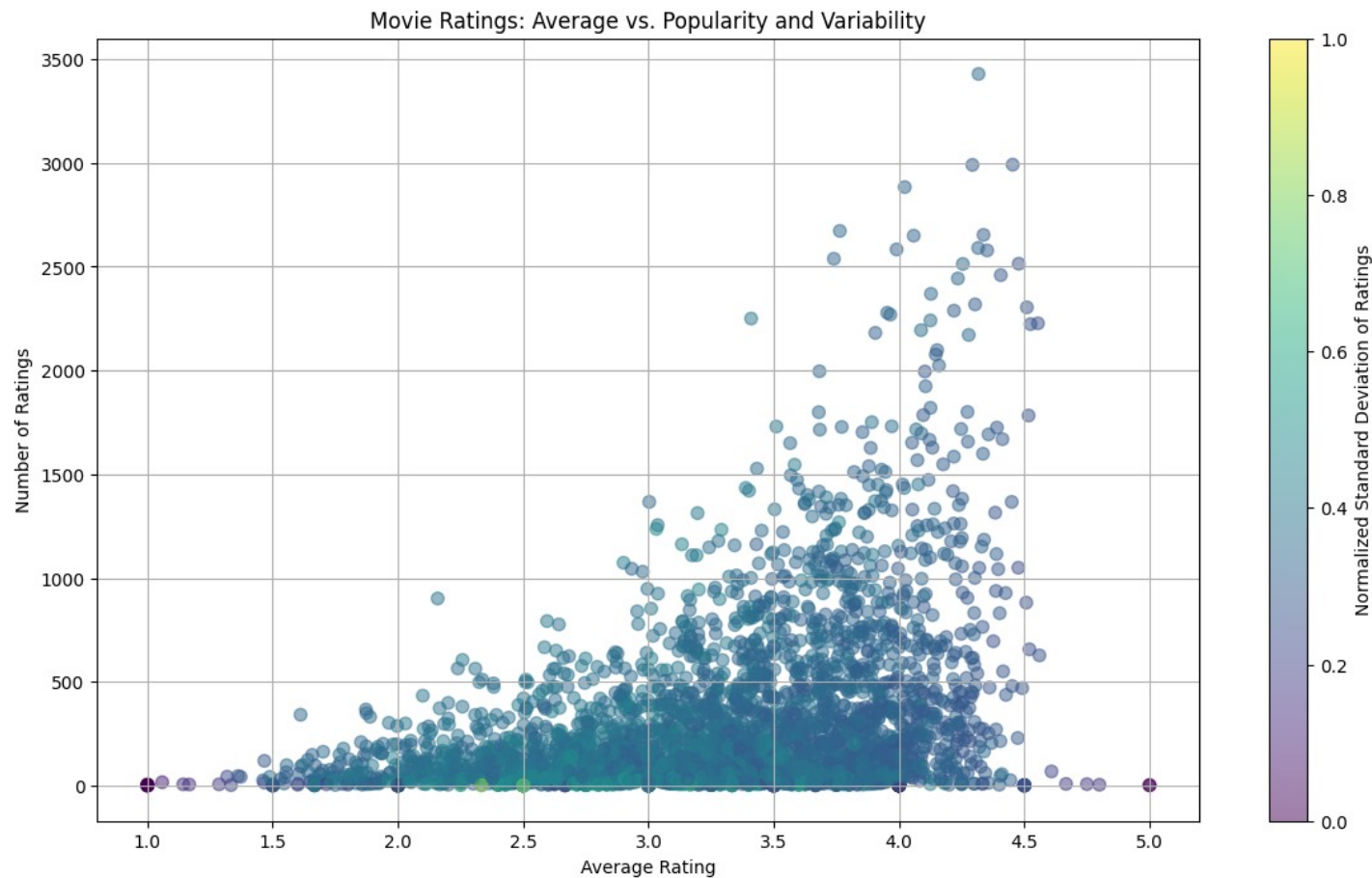
# EDA (Exploratory Data Analysis)

The provided box plot indicates consistent median ratings and narrower interquartile ranges across genres, suggesting genre preferences are a more stable indicator of user taste than individual movie ratings. With fewer outliers and a denser data representation, the Genre Matrix offers a reliable and robust foundation for predictive modeling. Hence, it's a preferable choice for our recommendation system over a sparse and variable user ratings matrix.





- Sparsity of User-Movie Ratings Matrix: The first plot illustrates a user-movie ratings matrix with extensive yellow areas, indicating many unrated movies. Such sparsity can lead to challenges in calculating reliable user similarities and predicting ratings, as there is insufficient data.
- Density of Genre Matrix: The second plot depicts a genre matrix that is largely black, signifying a higher density of ratings data. A dense matrix suggests more complete information, which is crucial for accurate predictions. The genre matrix, therefore, provides a more solid data foundation for modeling user preferences than the sparser user-movie matrix.



- The EDA plot illustrates a strong popularity bias, where widely-rated movies tend to have higher average ratings, and a high variability in ratings for less popular films, which can introduce noise into the user-rating matrix. In contrast, the genre matrix, with its reduced dimensionality, offers a more efficient and less biased alternative for Singular Value Decomposition (SVD). The genre matrix avoids the skew of popularity and provides a denser, more manageable dataset for SVD, leading to faster computations and potentially more accurate recommendations that better reflect genuine user preferences across genres.

### Complexity Point :

The choice to utilize the genre matrix is further justified by its dimensions of 3000 by 18, in contrast to the user data matrix's significantly larger size of 6000 by 3000. When applying SVD, the computational efficiency hinges on the size of the matrix. By calculating  $A^T A$  (where  $A$  is the genre matrix), we capitalize on the smaller number of genres (18) to expedite computation. This substantially reduces the computational load and enhances processing speed, making the genre matrix a much more efficient option for SVD in our recommendation system.

## **What is Genre Matrix and why was it chosen ?**

The **genre matrix**, composed of 18 columns representing the average rating for each genre by a user, encapsulates a denser, more meaningful representation of user preferences compared to a sparse user-item matrix. This denser matrix allows for a more nuanced understanding of user preferences across genres, facilitating a more informed and potentially more effective clustering process. SVD's role in this context is to further distill user preferences into latent factors that capture underlying patterns or affinities beyond the explicit genre ratings. By reducing the dimensionality to these latent factors, SVD simplifies the complex, high-dimensional space of user preferences into a more manageable, yet richly informative, form. This simplification is crucial for the subsequent K-means clustering, enabling it to operate on a cleaner, more abstract representation of user tastes.

## SVD (Algorithm explanation)

- QRDecomposition(A)

Purpose: Performs QR decomposition of a matrix A using the Gram-Schmidt process.

Process: Decomposes matrix A into an orthogonal matrix Q and an upper triangular matrix R. The Gram-Schmidt process is used to orthogonalize the columns of A, which become the columns of Q, while R contains coefficients that arise during the orthogonalization process.

- RankOfMatrix(mat)

Purpose: Determines the rank of a matrix mat using the Gaussian elimination method.

Process: Applies row operations to reduce mat to its row echelon form. The rank is determined by counting the non-zero rows in this form, adjusting for any row swaps or eliminations needed to handle zero elements on the diagonal.

- Eigen(A)

Purpose: Computes the eigenvalues and eigenvectors of a matrix A.

Process: Uses an iterative approach (power iteration) to approximate the eigenvalues and eigenvectors. The process involves repeatedly multiplying A by an initial set of vectors and applying QR decomposition until the vectors converge to eigenvectors of A. The eigenvalues are then estimated from these eigenvectors.

- FullSVD(A)

Purpose: Computes the full Singular Value Decomposition of a matrix A.

Process: Determines whether to work with  $A^T A$  or  $AA^T$  based on the dimensions of A, then uses the eig function to compute eigenvalues and eigenvectors. The square roots of the eigenvalues are the singular values (s), and the eigenvectors **form the** columns of U (or rows of  $V^T$ , depending on the dimensionality). Adjustments ensure U and  $V^T$  are properly derived from the eigenvectors.

- [\*\*ReducedSVD\(A, k\)\*\*](#)

Purpose: Computes a reduced Singular Value Decomposition of  $A$ , retaining only the top  $k$  singular values and their corresponding singular vectors.

Process: Similar to full\_SVD, but focuses on the largest  $k$  eigenvalues and their eigenvectors from  $A^T A$  or  $AA^T$ . This effectively provides a lower-rank approximation of  $A$ , useful for data compression or analysis tasks where full decomposition is unnecessary or computationally expensive.

## **Brief comparison and discussion about the methods used:**

In our exploration of recommendation systems,

- we employed two distinct methods for collaborative filtering: [Neighbourhood-based Collaborative Filtering](#) and a hybrid approach combining [Reduced Singular Value Decomposition \(SVD\) with K-means clustering](#)
- Neighbourhood-based Collaborative Filtering leverages the Pearson correlation coefficient, enhanced with inverse document frequency (IDF) weights, to calculate similarities between users
- This method predicts ratings by aggregating the ratings of similar users, specifically those who have previously interacted with the item, through a weighted sum approach where the weights are based on user similarity

On the other hand, our **hybrid method**,

- first utilizes Reduced SVD to capture the inherent latent variables in user-item interactions, effectively reducing the dimensionality of our dataset and uncovering a more concise user representation.
- Following this dimensionality reduction, K-means clustering is applied to group users into clusters based on their preferences. For a new user, the system determines their cluster and then computes predictions using a weighted sum of ratings from users within the same cluster, again employing the Pearson correlation coefficient with IDF for similarity calculation.
- This approach narrows the scope from the entire dataset to more relevant subsets, potentially enhancing efficiency and relevance.

## Contrast

In comparing neighbourhood-based collaborative filtering and the integrated **SVD-K-means approach** for predicting user ratings, a key distinction lies in how they conceptualize and utilize user similarities and data dimensionality. Neighbourhood-based methods rely on calculating similarities directly between users, using a Pearson correlation coefficient adjusted by inverse document frequency to refine similarity measures. This approach emphasizes direct user-to-user comparisons but may struggle with scalability and sparsity, as it requires computing and storing all pairwise user similarities.

Conversely, the **SVD-K-means method** introduces a dimensionality reduction step through Singular Value Decomposition (SVD), capturing latent variables that represent underlying patterns or preferences in the data. These latent variables provide a more detailed and compact user representation, facilitating the identification of user clusters via K-means clustering. This preliminary clustering allows the algorithm to consider only a subset of similar users (those within the same cluster) when predicting ratings, thus improving efficiency and potentially capturing more nuanced user relationships that direct comparison methods might overlook. By incorporating Pearson correlation with IDF within clusters, the SVD-K-means approach further refines its predictions,

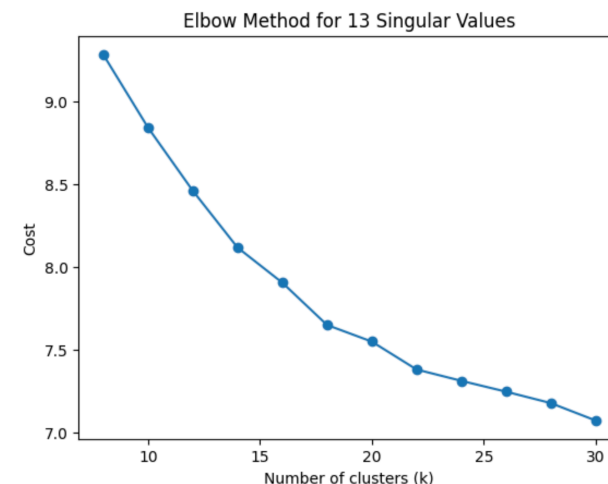
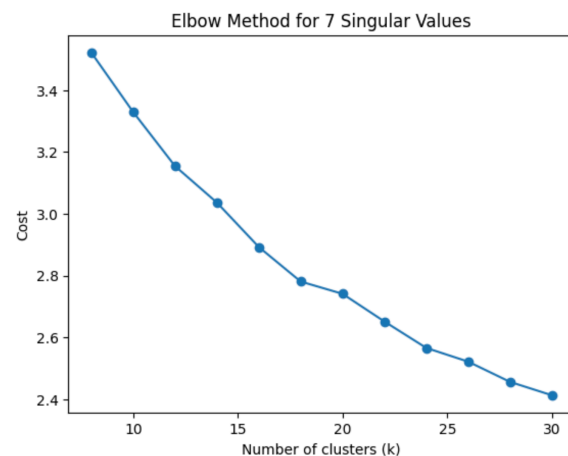


blending the benefits of dimensional reduction with localized similarity measures. This contrast highlights a fundamental trade-off between the direct, all-encompassing comparison of the neighbourhood approach and the efficiency, scalability, and nuanced user representation achieved through the combination of SVD and K-means clustering.

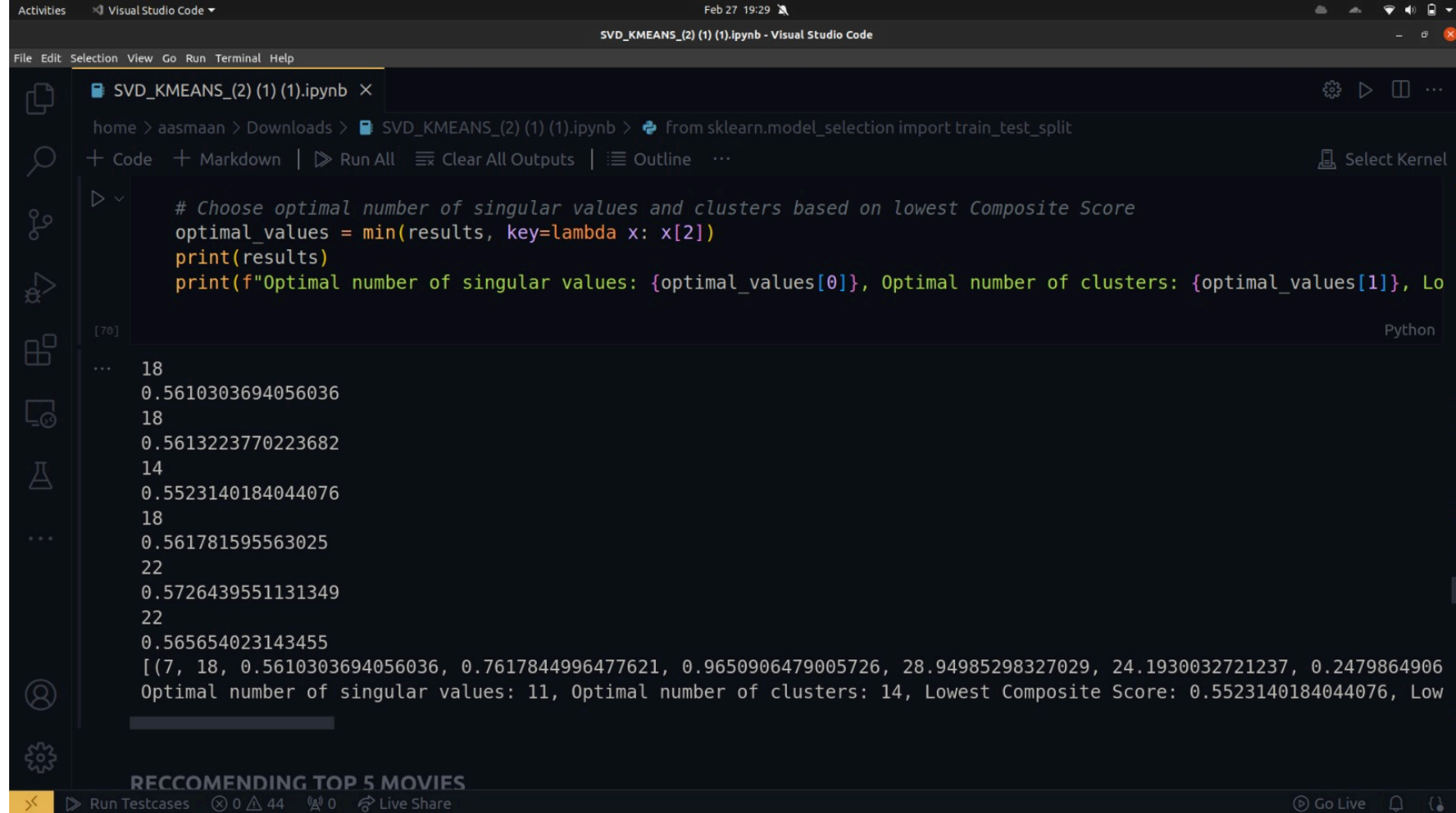
The **genre matrix**, composed of 18 columns representing the average rating for each genre by a user, encapsulates a denser, more meaningful representation of user preferences compared to a sparse user-item matrix. This denser matrix allows for a more nuanced understanding of user preferences across genres, facilitating a more informed and potentially more effective clustering process. SVD's role in this context is to further distill user preferences into latent factors that capture underlying patterns or affinities beyond the explicit genre ratings. By reducing the dimensionality to these latent factors, SVD simplifies the complex, high-dimensional space of user preferences into a more manageable, yet richly informative, form. This simplification is crucial for the subsequent K-means clustering, enabling it to operate on a cleaner, more abstract representation of user tastes.

## Why certain number of **clusters** and **singular values** were used ?

- In our optimization process for enhancing collaborative filtering using Singular Value Decomposition (SVD) and K-means clustering, we employed a strategic two-step approach to identify the optimal combination of dimensionality reduction and clustering granularity. Our preliminary analysis identified 18 non-zero singular values within our genre matrix, indicating diverse data variation dimensions. We explored singular values from 7 to 18 in steps of 2 to balance data simplification and maintain complexity.



- For each singular value, we aimed to find the corresponding optimal k value for K-means clustering. The range for selecting k values was informed by the heuristic of the square root of half the number of users, guiding us to a practical cluster number that reflects the dataset scale and complexity. By plotting inertia against k values and identifying the elbow point where inertia's decrease rate changes significantly, we determined the optimal k value for each singular value. This elbow point signifies the most efficient balance between cluster compactness and quantity.
  - We then evaluated each singular value-k value pair by doing a train:test split on the original data set against weighted sum of key performance metrics such as Mean Absolute Error (MAE) , Root Mean Square Error (RMSE), to select the combination that best improves prediction accuracy. This methodical evaluation led to identifying an optimal singular value and k value pairing that optimally balances dimensionality reduction, effective clustering, and predictive performance, enhancing our collaborative filtering model's accuracy and efficiency
- 
- We found the Optimal number of singular values to be- **11** and the optimal number of clusters to be- **14**



The screenshot shows a Jupyter Notebook titled 'SVD\_KMEANS\_(2) (1) (1).ipynb' in Visual Studio Code. The code cell contains the following Python code:

```
# Choose optimal number of singular values and clusters based on lowest Composite Score
optimal_values = min(results, key=lambda x: x[2])
print(results)
print(f"Optimal number of singular values: {optimal_values[0]}, Optimal number of clusters: {optimal_values[1]}, Lo
```

The output of the code is displayed below the code cell:

```
[70]
... 18
0.5610303694056036
18
0.5613223770223682
14
0.5523140184044076
18
0.561781595563025
22
0.5726439551131349
22
0.565654023143455
[(7, 18, 0.5610303694056036, 0.7617844996477621, 0.9650906479005726, 28.94985298327029, 24.1930032721237, 0.2479864906
Optimal number of singular values: 11, Optimal number of clusters: 14, Lowest Composite Score: 0.5523140184044076, Low
```

The bottom of the notebook shows the text 'RECCOMENDING TOP 5 MOVIES'.

## Novelty

### 1. Novelty on [evaluation metrics](#)

- We Introduced composite score such that it integrates five key metrics: [Mean Absolute Error \(MAE\)](#) and [Root Mean Square Error \(RMSE\)](#) focus on the absolute and squared differences between predicted and actual values, respectively, highlighting

general accuracy and sensitivity to large errors. [Mean Absolute Percentage Error \(MAPE\)](#) and [Symmetric Mean Absolute Percentage Error \(SMAPE\)](#) measure errors in percentage terms, offering insights into relative prediction accuracy while addressing asymmetry in over- and under-predictions. Mean Squared Logarithmic Error (MSLE) emphasizes the accuracy of predictions in datasets with exponential growth, penalizing underestimations more than overestimations.

- Assigning different weights to these metrics allows for a nuanced evaluation of model performance, balancing the importance of various error characteristics. The weights (MAE and RMSE at 0.25, MAPE and SMAPE at 0.15, and MSLE at 0.20) reflect a deliberate choice to prioritize absolute error magnitudes (MAE, RMSE) while still considering the proportional errors (MAPE, SMAPE) and the specific context of exponential data growth (MSLE).
- This balanced weighting scheme ensures that the composite score is a comprehensive reflection of model performance, accounting for both the magnitude and relative size of prediction errors. It facilitates a more holistic assessment than relying on a single metric, making it a versatile tool for evaluating and comparing predictive models across different datasets and objectives.

## 2. Novelty on [Diversity of Movies recommendation](#)

- Originally, our method simply picked the top 5 movies with the highest predicted ratings for recommendations. However, we realized that adding variety from different genres could introduce users to new types of movies they might enjoy.
- To achieve this, we revamped our approach. First, we created a genre-to-movie list dictionary. From this, we selected a set number of top-rated movies across different genres for variety.
- Then, if there were spots left to reach five recommendations (for instance, if we only picked 3 diverse movies), we filled these spots with the highest-rated movies regardless of their genre. This way, we provided users with personalized recommendations that also encouraged exploring new genres.

## 3. Novelty on [Using IDF in Similarity Calculation and SVD and K-Means](#)

- In our movie recommendation system, which cleverly combines the techniques of Singular Value Decomposition (SVD) and K-means clustering, we've introduced a smart twist to enhance the accuracy of our predictions: the Inverse Document Frequency (IDF).
- By applying IDF, we give more weight to the movies that aren't as commonly rated high or low by everyone. This means that if two users both like or dislike movies that are not as universally acknowledged, it's considered a stronger indicator of their similar tastes.
- We also integrated this refined similarity measure, bolstered by IDF, into our SVD and K-means model. Our model first uses SVD to reduce the complexity of user preferences into more manageable, underlying patterns. Then, with K-means, we group users into clusters based on these simplified preferences. Within each of these clusters, we use our enhanced similarity measure (thanks to IDF) to predict how much a user might enjoy a movie they haven't seen yet. This approach allows us to make more nuanced and accurate recommendations, ensuring that we're not just echoing popular opinions but uncovering genuine, shared interests among users.

#### 4. Novelty on [Choice of User- Genre Matrix over User Movie Matrix:](#)

The Genre Matrix is characterized by consistent median ratings, narrower interquartile ranges, fewer outliers, and a denser data representation, indicating that genre preferences are a stable indicator of user taste and offer a reliable foundation for recommendations. In contrast, the User-Movie Ratings Matrix exhibits extensive sparsity with many unrated movies, leading to challenges in calculating reliable user similarities and predicting ratings. The Genre Matrix's reduced dimensionality and higher density of data not only facilitate more accurate predictions but also improve computational efficiency, especially when applying Singular Value Decomposition (SVD), due to its significantly smaller dimensions (3000x18) compared to the user data matrix (6000x3000), making it a preferable choice for recommendation systems.

