BT4222 Final Report

Recommendation System 7



| **Name** | **Matriculation Number** |
| --- | --- |
| Aarushi Mundra | A0246207J |
| Dylan Wo Qiying | A0233634J |
| Kaavya Senthil Kumar | A0246227E |
| Kenny Low Sheng Wei | A0233869N |
| Ng Wei Han | A0234338E |
| Tasneem Benazir | A0238339W |

**Contents Page**

[**Abstract 2**](#_fhkiuratmm4i)

[**Proposal Review 3**](#_iok0xomih2if)

[**Data Description 4**](#_v0d9i6xgo3a6)

[Overview of Datasets Used 4](#_jswog4a6jq7f)

[Features Used 4](#_norzzxe5628d)

[**Models and Performance 6**](#_ngja5ed89s3b)

[Content-Based Filtering (TF-IDF / CountVec + Cosine Sim) 6](#_vvko2znnxz3i)

[Collaborative Filtering (Memory-Based / Model-Based / Hybrid) 7](#_4oqu15ng5ic1)

[**Contribution and Justification 8**](#_smm833tg7ffw)

[New Datasets 9](#_bia22mq12dp4)

[a. Incorporating an image dataset with movie posters 9](#_hgr9uu908g81)

[Convolutional Neural Network 9](#_uoicdkv5pd0q)

[Creativity in Feature Engineering 10](#_81gz33jy6ih4)

[a. TF-IDF vs. CountVectorizer 10](#_pn2r2fz0m08k)

[b. `Liked` and `Disliked` creation with chronological ordering 10](#_qe5kdt85j7za)

[c. Bayesian Average 11](#_ud52djcgeuvm)

[Adoption of new ML Methods / Architecture 12](#_8a7skdoxh91q)

[a. Hybrid Collaborative Filtering Model 12](#_b3a0i8ecelgr)

[b. Improved CF-CBF Hybrid Model with Threshold 12](#_dgdn6gezbukt)

[Understanding Prediction Results 14](#_ozyv9x77pnvc)

[a. Content-Based Filtering Evaluation 14](#_mupratdw1vb7)

[b. Collaborative Filtering Evaluation 15](#_cx7lgp7j1jou)

[**References 16**](#_wlxm2j4v3p6i)

# Abstract

This report addresses a critical business challenge within the movie recommendation domain: the need to evaluate the quality of search results and offer users movies that are seamlessly aligned with their queries or their browsing history. Thus the primary machine learning problem that we will be tackling is the need to predict individual preferences and behaviour and provide tailored movie recommendations for each user.

Leveraging the extensive MovieLens1 dataset, this research incorporates a multifaceted approach. We explored the use of demographic, content-based, collaborative recommendation systems. To enhance comprehensiveness, we integrated content-based and collaborative systems into a hybrid recommender, mitigating limitations associated with their separate use. Additionally, Convolutional Neural Network models were employed, to furnish recommendations grounded in visual elements.

**Key Achievements:**

* Incorporation of new datasets
  + Web Scraped OMDb API for movie posters to be used as input for Convolutional Neural Network models
* Innovative Feature Engineering
  + Combined the genres, actors and directors attributes for Content-Based Filtering to enhance robustness
  + Created a binary variable to denote the likes and dislikes of a user which was used as a personalised evaluation metrics for Content-Based Filtering
  + Usage of Bayesian Average for Demographic Filtering
* Adoption of new Machine Learning Methods
  + Incorporation of Hybrid Recommender that uses a threshold to resolve the cold start problem
* Creativity in Evaluating Results
  + Introduced Precision, Recall, F1-score as evaluation metrics for Content-Based Filtering
  + Collaborative Filtering Evaluation

This comprehensive methodology aims to significantly advance the user satisfaction and engagement in movie recommendation systems by providing nuanced and personalised movie suggestions.

# Proposal Review

The following table below summarises the potential contributions that we had listed in our proposal along with a self-evaluation of what we have achieved in our final project.

| ***Proposed Idea*** | ***Exceeds Expectations*** | ***Satisfactory*** | ***Under Expectations*** | ***Not Achieved*** |
| --- | --- | --- | --- | --- |
| ***Hyperparameter tuning (Model-Based CF)*** |  |  |  |  |
| ***CF using kNN*** |  |  |  |  |
| ***Bayesian Average in Demographic Filtering*** |  |  |  |  |
| ***Hybrid RS*** |  |  |  |  |
| ***Re-ranking Models*** |  |  |  |  |

We decided against implementing reranking models in our movie recommender system, primarily due to the perceived complexity associated with their integration. Following a thorough assessment, we acknowledged that the implementation of reranking models could introduce a considerable level of intricacy to our recommender system. While we considered a simpler reranking approach based on popularity scores, we ultimately decided against it due to its simplicity. Given the constraints of our current scope and available resources, we strategically chose to prioritise efficiency and feasibility in our approach.

# Data Description

## Overview of Datasets Used

We chose to use the MovieLens1 dataset for our demographic, content-based, collaborative and hybrid recommender systems. It consists of the following files as shown in the table below.

| File Name and Source | Number of Original Features | Number of Original Data Points |
| --- | --- | --- |
| [movies\_metadata.csv](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv) | 24 | 45,466 |
| [keywords.csv](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=keywords.csv) | 2 | 46,419 |
| [credits.csv](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=credits.csv) | 3 | 45,476 |
| [ratings\_small.csv](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=ratings_small.csv) | 4 | 100,004 |
| [links\_small.csv](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=links_small.csv) | 3 | 9,125 |

As for the Convolutional Neural Network model, we utilised web scraping to extract the movie posters from the OMDb API4 to be used as the input to the model.

## Features Used

The following tables below contain more information on the features from MovieLens that we have used in our models.

From credits.csv:

| Feature Name | Feature Brief Description | Details |
| --- | --- | --- |
| cast | Contains cast information in stringified JSON object form | Keys: cast\_id, character, credit\_id, gender, name, order, profile\_path |
| crew | Contains crew information in stringified JSON object form | Keys: credit\_id, department, gender, id, job, name, profile\_path |

From keywords.csv:

| Feature Name | Feature Brief Description | Details |
| --- | --- | --- |
| keywords | Contains movie plot keywords in stringified JSON object form | Keys: keyword\_id, keyword\_name |

From ratings\_small.csv:

| Feature Name | Feature Brief Description | Details |
| --- | --- | --- |
| rating | A float that represents the user’s rating for a movie out of five | min: 0.5, max: 5.0, mean: 3.543608, median: 4.0, s.d: 1.058064, missing: none |
| timestamp | Epoch unix timestamp that represents when the user gave the rating | - |

For movies\_metadata:

| Feature Name | Feature Brief Description | Details |
| --- | --- | --- |
| genres | Contains movie’s genres in stringified JSON object form | Keys: genre\_id, genre\_name |
| original\_title | String of the movie’s original title | - |
| overview | String of movie’s overview | - |
| tagline | String of movie’s tagline | - |
| vote\_average | Float of movie’s average rating out of ten | min: 0.0, max: 10.0, mean: 5.618329, median: 6.0, s.d: 1.924139, missing: 6 |
| vote\_count | Integer of total movie rating count | min: 0, max: 14075, mean: 109.935989, median: 10, s.d: 491.466335, missing: 6 |

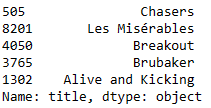
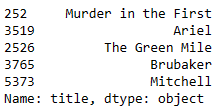
# Models and Performance

## Content-Based Filtering (TF-IDF / CountVec + Cosine Sim)

Based on the dataset, we split and engineered 2 movie features:

* **Movie Description**: Contains movie features in the form of sentences, obtained by concatenating `overview` and `tagline` features from movie metadata
* **Movie Information**: Containing movie features in the form of words, phrases and names, obtained by concatenating `genres` from movie metadata, `keywords` from keywords dataset, and `cast` and `director` from `credits` dataset.

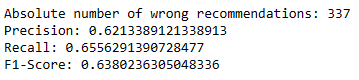
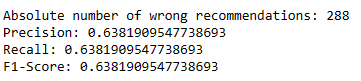
Based on the 2 engineered features, we chose to utilise TF-IDF for `Movie Description` and CountVectorizer for `Movie Information`.

. 

Figures 1 and 2: Output of the Description Recommender (left) and Information Recommender (right) based on the same movie ‘The Shawshank Redemption’.

As we can see from the outputs above, each recommender gave largely different outputs although we used the same input ‘The Shawshank Redemption’.

Based on our designed evaluation pipeline that will be explained in detail in section 4, we can obtain the following results.



Figures 3 and 4: Evaluation metrics of our Description Recommender (left) and Information Recommender (right).

Based on these 2 recommenders, we used a grid search algorithm to find a weighted sum of their cosine similarity matrices to obtain a recommender with the highest F1-score.

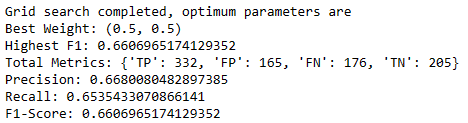


Figure 5: Evaluation metrics for the optimal combined CBF recommender.

The optimal recommender uses 0.5 of each cosine matrix and has an F1-score of 66.0%, greater than that of each individual recommender, which had an F1-score of 63.8% each.

## Collaborative Filtering (Memory-Based / Model-Based / Hybrid)

Memory-based algorithms apply statistical techniques to the entire dataset to calculate the predictions. They can be divided into two main sections: user-item filtering or item-item filtering, and also Cosine similarity, Pearson correlation coefficients, or Mean Squared Differences, which are solely based on arithmetic operations. Upon evaluating the different methods in Surprise, we decided to use the Item-Based Pearson Correlation KNNBaseline, with hyperparameters k = 50, min\_k = 5, and min\_support = 5 (obtained by GridSearch), as Item-Based KNNBaseline gave the lowest RMSE, while Pearson Correlation inherently normalises the rating scales by considering the deviation from the mean rating of each user, which is important as every user might have a different rating scale.

| Memory-Based Algorithm | RMSE | MAE | Time |

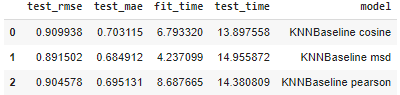
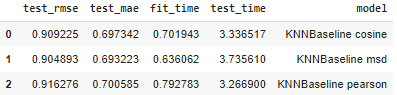
|:-------------------------|-------:|------:|:--------|

| KNNBasic | 0.969 | 0.745 | 0:00:10 |

| KNNWithMeans | 0.92 | 0.704 | 0:00:12 |

| KNNBaseline | 0.897 | 0.686 | 0:00:15 |

| KNNWithZScore | 0.919 | 0.699 | 0:00:13 |



*Figure 6, 7 8: Performance of Memory-Based Algorithms (Top), KNNBaseline User-based (Left) vs Item-based (Right)*

Model based approach involves building machine learning algorithms to predict user's ratings. They involve dimensionality reduction methods that reduce high dimensional matrices containing an abundant number of missing values with a much smaller matrix in lower-dimensional space.

| Model-Based Algorithm | RMSE | MAE | Time |

|:------------------------|-------:|------:|:--------|

| SVD | 0.895 | 0.689 | 0:00:11 |

| SVDpp | 0.886 | 0.679 | 0:07:55 |

| NMF | 0.946 | 0.726 | 0:00:26 |

*Figure 9: Performance of Model-Based Algorithms*

SVDpp performs better, but due to its longer computing time, we decided to use SVD, which is the next best model. Optimising using GridSearch gives us, N\_factors = 80, n\_epochs = 20, lr\_all = 0.005, reg\_all = 0.2.

# Contribution and Justification

The following table below summarises our contributions to this project along with a self-evaluation.

| ***Contribution*** |  | ***Low*** | ***Medium*** | ***High*** |
| --- | --- | --- | --- | --- |
| ***Use valuable and high-quality new datasets,  including integrating existing datasets,  scrawling or retrieving data, etc.*** | Effort |  |  |  |
| Effectiveness |  |  |  |
| ***Creativity in feature engineering in a way that***  ***- overcomes the limitations of current data or model;***  ***-***  ***or increases model performance;***  - ***or increases the interpretability of the models;***  ***- or reduces computing cost*** | Effort |  |  |  |
| Effectiveness |  |  |  |
| ***Design or adaptation of new ML methods/architecture or the integration of existing methods with a balance of resource and cost*** | Effort |  |  |  |
| Effectiveness |  |  |  |
| ***Creativity or insights in understanding or further explaining the prediction results and performance*** | Effort |  |  |  |
| Effectiveness |  |  |  |
| ***Any aspects that are distinct from the above*** | Effort |  |  |  |
| Effectiveness |  |  |  |

## New Datasets

### Incorporating an image dataset with movie posters

We integrated a dataset comprising movie posters to create movie recommendations based on similar themes of the posters using a Convolutional Neural Network. Traditional recommendation systems primarily focus on textual data. Our incorporation of visual elements aligns with emerging trends in multimedia recommendation research.

#### Convolutional Neural Network

We utilised an online Convolutional Neural Network (CNN) model3 to harness its capabilities for pattern recognition, specifically in discerning thematic tones through shared colour palettes in movie posters.

Our choice of the CNN model was grounded in the belief that shared colour palettes in movie posters can indicate similar thematic tones, influencing viewer preferences. Leveraging CNN’s hierarchical feature extraction we can recommend movies to users based on posters with analogous colour schemes. Upon querying the CBF model using the prompt, ‘The Shawshank Redemption’, the expected outcomes were observed. Notably, ‘The Chasers’ was excluded from the suggestions due to its lighter poster colour, while movie posters like Les Misérables and Alive and Kicking persisted within the recommended selections.



*Figure 10: Results from CNN model*

## Creativity in Feature Engineering

### TF-IDF vs. CountVectorizer

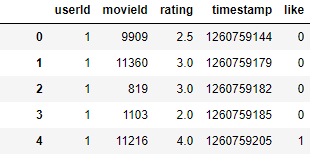
Based on the 2 engineered features `description` and `information`, we chose to utilise TF-IDF for `Movie Description` and CountVectorizer for `Movie Information`, as TF-IDF is well-suited for processing longer, more descriptive content while CountVectorizer is computationally efficient in handling the large number of independent words. This resulted in the construction of 2 separate recommenders based on each representation method.

We ultimately combined both recommenders through a grid search algorithm that took a weighted average of each cosine similarity matrix. This new weighted cosine similarity matrix can also be considered as a new feature constructed with both recommenders.

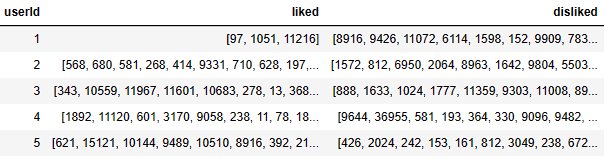
### `Liked` and `Disliked` creation with chronological ordering

We introduced a personalised evaluation system for CBF models by creating liked and disliked movies based on user ratings, whereby a user likes the movie if their rating is more than or equal to 4.0.

Since CBF models lack standard evaluation metrics, we wanted to mimic how modern day recommendation systems generate recommendations across time. By arranging likes and dislikes in chronological order, we can generate recommendations based on the earliest review, and evaluate our predictions.



*Figure 11: User Ratings with `Likes` and `Dislikes`*

**

*Figure 12: Each User’s `Likes` and `Dislikes` in Chronological Order.*

### Bayesian Average

We came across the Bayesian Average model on a website2 proposed for e-commerce. We then repurposed it for movies. This involved introducing a new column through the utilisation of Baysian average, combining popularity and ratings to enhance demographic filtering.

The formula we use is: Weighted Rating (WR) = (𝑣/(𝑣+𝑚)) \* 𝑅+(𝑚/(𝑣+𝑚)) \* 𝐶 where,

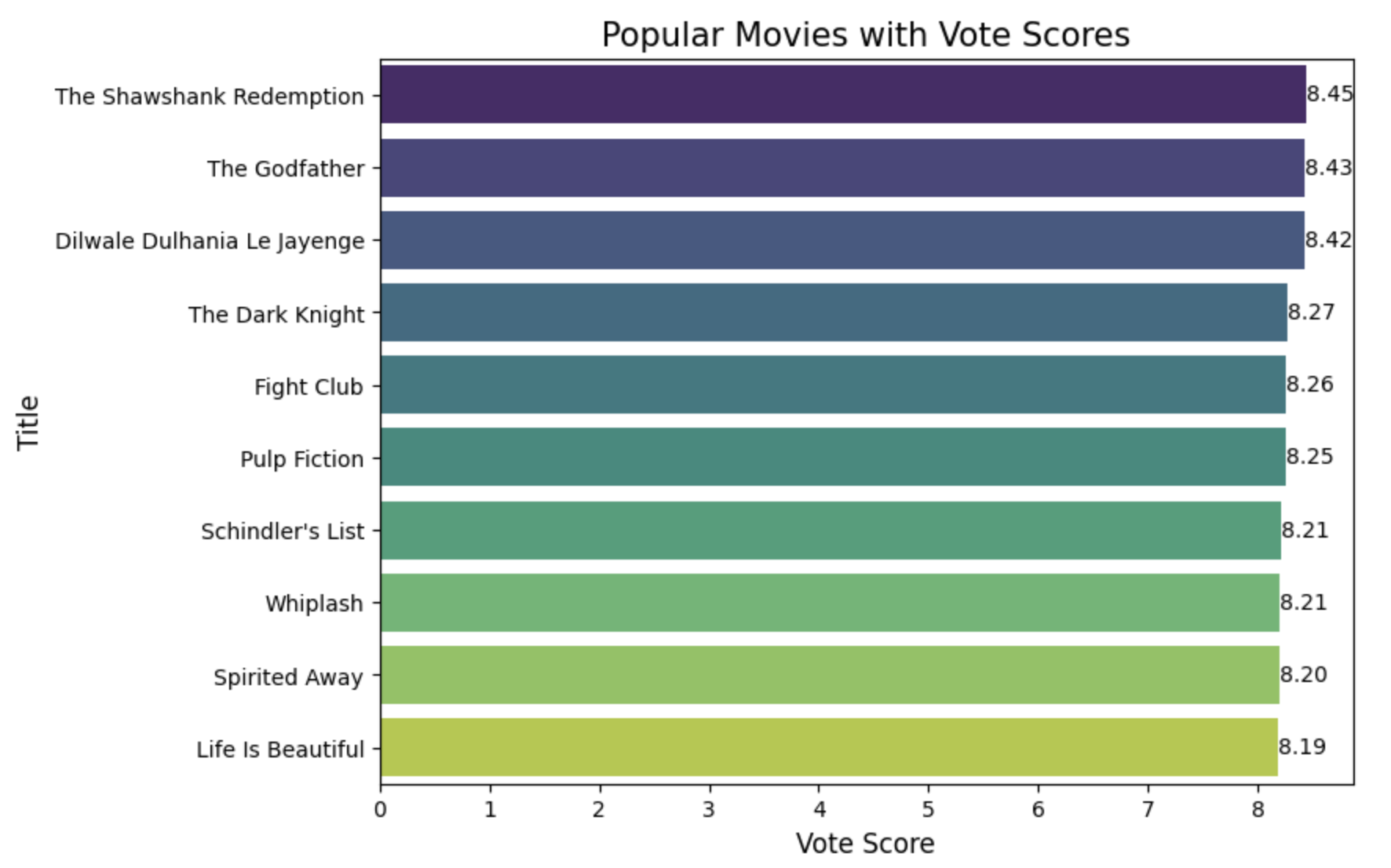
v: the number of votes for the movie

m: the minimum votes required to be listed in the chart

R: the average rating of the movie

C: the mean vote across the whole report

Certain highly rated movies with low number of ratings may be a niche movie not enjoyed by the general public. Thus, by computing the Bayesian Average for each movie, each rating will be adjusted in accordance to its deviation from the mean rating and the rating count threshold, hence providing a more accurate metric for gauging each movies’ popularity.

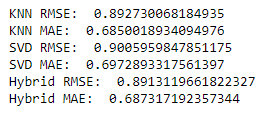


*Figure 13: Graph of popularity based on Bayesian Averages.*

## Adoption of new ML Methods / Architecture

### Hybrid Collaborative Filtering Model

Here, we attempt to make a hybrid collaborative filtering recommender to produce a better model for improved recommendations. We used the rating predictions from each of the models and combined the result with static weightings of 0.5 each. This resulted in lower RMSE and we will use this hybrid-based collaborative model for our final model.



*Figure 14: Accuracy of all 3 types of models*

Memory-based CF is good at capturing user preferences based on historical interactions, while model-based CF can handle sparse data and provide more accurate predictions in scenarios with limited user-item interactions. The combined model produced reduced RMSE and enhanced results, demonstrating the effectiveness of combining the two.

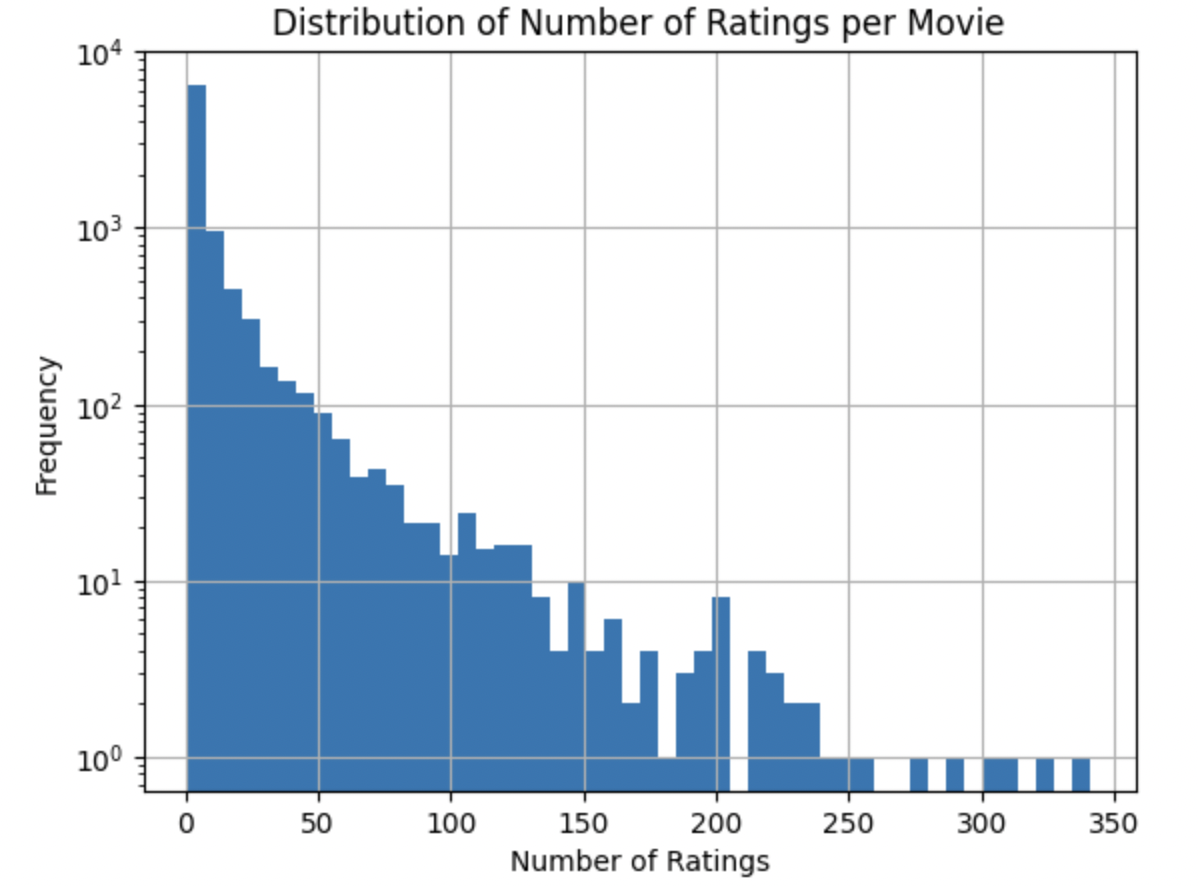
### Improved CF-CBF Hybrid Model with Threshold

Due to the limitations other individual models posed such as CBF is limited in recommending items outside a user's known preferences, CF struggles with cold start problems for new users or new movies which do not have enough rating, and Demographic Model does not take into account user sentiments.

Here we calculate a threshold for the number of ratings a movie must have in order to determine which recommendation system it should run under. To solve the cold start issue associated with CF, where a new movie with not enough ratings might not be chosen as a recommendation, for the movies with number of ratings above a certain threshold, we use both CF and CBF recommenders whereas for the movies with ratings lower than that threshold, we only use CBF.

CF might not work well for movies with very few ratings, so relying on the CBF recommender is a reasonable alternative for this hybrid model.

As such, we implemented the hybrid recommender. Firstly, we calculate the 90th percentile of ratings, then created a list of movies in the top 10% based on the number of ratings



*Figure 15: Distribution of number of ratings per movie*

If a recommendation is asked for a movie that lies within the top 10% based on number of ratings, we use CBF + CF. If the movie has a number of ratings lower than the threshold, we only used CBF.

## 

## 

## 

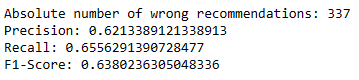
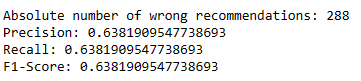
## Understanding Prediction Results

### Content-Based Filtering Evaluation

We introduced Precision, Recall, F1-score as new evaluation metrics for CBF models. For our optimization methods, we chose to focus on F1-score, since it helps to capture wrongly made predictions (FN and FP).

As compared to traditional classification problems, CBF evaluation is challenging as finding the intersection between each user’s watched and recommended movies is often rare. However, through rigorous tests, F1-score offers the best predictive balance across classes.

By first observing the distribution of user ratings, we set a threshold of 4.0 for liking a movie, and make predictions on each user’s first liked and disliked movie. Here, we were trying to mimic how modern day recommendation systems recommend movies across time. Based on this, we could compute the F1-Score of our recommenders.



Figures 16 and 17: Evaluation metrics of our Description Recommender (left) and Information Recommender (right).

As we refer to our evaluation metrics that was mentioned before, our individual CBF recommenders have about the same F1-score performance. Precision here is the probability of making a right prediction on liked movies while recall is the probability of a liked movie being correctly predicted as liked. F1-score is hence the harmonic mean of these probabilities. However, in reality, every piece of information regarding a movie influences our preferences towards it, and hence the reason why we decided to combine both recommenders through a grid search algorithm.

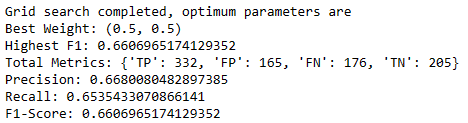
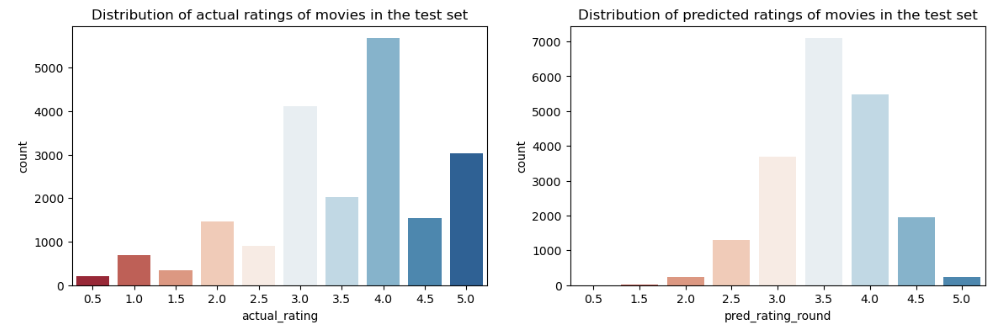


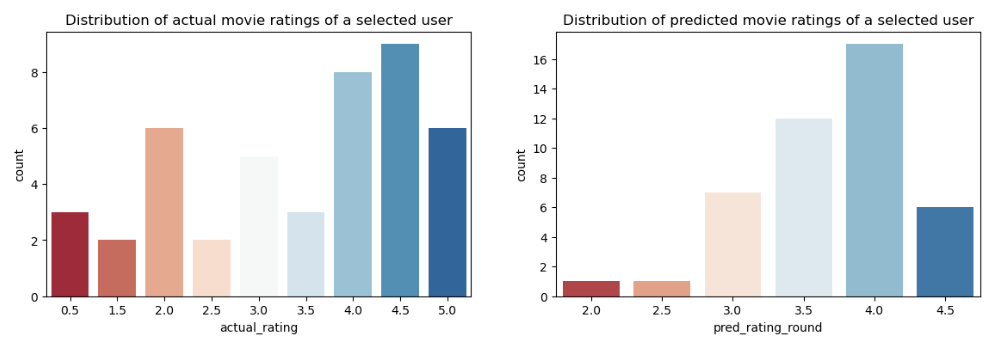
Figure 18: Evaluation metrics for the optimal combined CBF recommender

From our grid search algorithm, we found that the optimal recommender with the highest F1-score utilises 0.5 of each individual CBFs cosine similarity matrix. This means that the most effective recommendations are a result from combining information from movie descriptions and movie information in equal proportions.

### Collaborative Filtering Evaluation

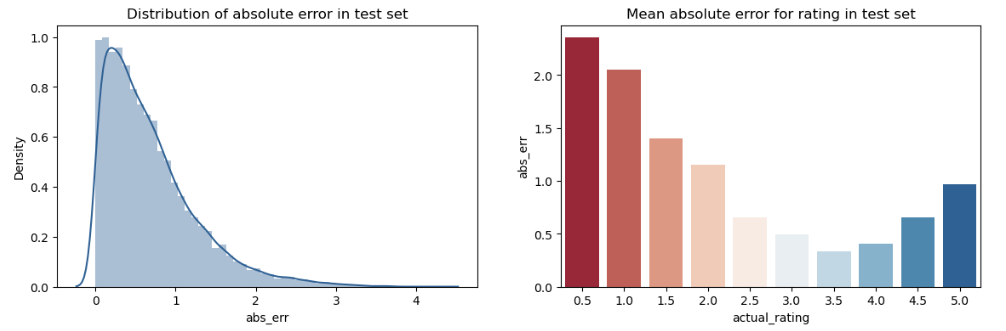
Only the memory-based model here is shown, but both types of models perform similarly. The provided histograms compare the distribution of actual ratings versus predicted ratings for movies in a test set and for a selected user. From the histograms, it appears that the collaborative filtering model is underpredicting the number of 5.0 ratings.





*Figure 19,20,21,22 shows the distribution of the movies actual and predict ratings for test set and an user*

One of the reasons could be that users who rate movies may tend to reserve 5.0 ratings for exceptional movies. The model might be picking up on the fact that users are more conservative with perfect scores, and thus it predicts fewer 5.0 ratings.



*Figure 23, 24 shows the distribution of absolute error (left) and mean absolute error (right) in test set*

Looking at the absolute errors of the test set, we can see errors are higher for extreme ratings (0.5 and 5) and lower for mid-range ratings. This could be due to a tendency to avoid extreme predictions where models may be less likely to predict extreme values to minimise the potential for large errors, which could be particularly penalised by certain accuracy metrics like RMSE. One way to solve this could be to design a loss function that penalises the underprediction of 5.0 ratings more heavily.

# References

1. **Rounak Banik, The Movies Dataset, 2017, Retrieved October 23, 2023,** <https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv>
2. **Using the Bayesian Average in Custom Ranking, October 23, 2023, Retrieved November 2, 2023** <https://www.algolia.com/doc/guides/managing-results/must-do/custom-ranking/how-to/bayesian-average/>
3. **Soumil, Pythonist: Computing Similarity on Images using Machine Learning and Cosine Similarity, July 10, 2020, Retrieved November 10, 2023** <https://soumilshah1995.blogspot.com/2020/07/computing-similarity-on-images-using.html>
4. **OMDb API, The Open Movie Database, Retrieved November 11, 2023** <https://www.omdbapi.com/>