

CAP 5768 Introduction to Data Science, Fall 2021

# Recommendation System on OTT Platforms

**Professor:** Dr. Giri Narasimhan

**By:**

Shivani Elakurthy – 6298533

Lakshmi Kolluru – 6266572

Ravi Chandra Madamanchi – 6307383

Santhosh Gadipelly – 6297510

# Table of Contents

I. Introduction .....	2
II. Approach .....	2
III. Data .....	3
III.1. OTT Platforms Datasets .....	3
III.2. User Rating Datasets .....	3
IV. Aim 1: Genre Prediction .....	4
IV.1. Approach.....	4
IV.2. Experiments .....	5
IV.3. Results and Discussions .....	6
IV.1.4. Conclusions .....	8
V. Aim 2: Collaborative Filtering .....	8
V.1. Approach.....	8
V.2. Experiments .....	9
V.3. Results and Discussions .....	10
V.4. Conclusions .....	13
VI. Aim 3: Plot Filtering .....	13
VI.1. Approach.....	13
VI.2. Experiments .....	14
VI.3. Results and Discussions .....	14
VI.4. Conclusions .....	16
VII. Aim 4: Content-based Filtering.....	17
VII.1. Approach.....	17
VII.2. Experiments .....	17
VII.3. Results and Discussions .....	18
VII.4. Conclusions .....	21
VIII. Final Recommendation .....	21
VIII.1. Approach.....	21
VIII.2. Result and Discussion .....	21
VIII.3. Conclusion.....	22
Citations .....	23

# I. Introduction

Our project is on building a hybrid recommendation system based on the content available on various OTT platforms. Movies/series will be recommended to the user by considering the *ratings* provided by the user to different movies/series and also other features of the movie/series like *genres, actors, description*, etc.

Each OTT platform has its recommendation system and recommends its own content to the user. The recommendations to a movie watched or liked by a user on one OTT platform are restricted to that OTT platform. There might be content similar to that movie on other OTT platforms, which the user might like. These will never be known to the user. In order to improve the recommendations for each user, the recommendations will not be restricted to a single OTT platform.

The project aims to first predict all the missing genres in our dataset, followed by collaborative filtering, plot filtering, and content-based filtering on which the final hybrid recommendation system will be built.

# II. Approach

The structure of the project is as follows:

- **Genre Prediction:** To predict the genres using the plot or summary for movies/series that have missing genre information.
- **Collaborative Filtering:** To predict how a user would rate a movie/series that the user has not yet watched.
- **Plot Filtering:** Finds content similar to a given movie/series based on its plot or summary.
- **Content-based Filtering:** Finds content similar to a given movie/series based on features like *genres, directors, actors, content ratings*.
- **Final Recommendation:** *Collaborative, Plot, and Content – based* filtering are combined to generate the final recommendation system or the *Hybrid* recommendation system.

*Scikit – learn* library is used for preprocessing tools (like *MultiLabelBinarizer*), machine learning models (like *LogisticRegression*), evaluation metrics (like *hamming\_loss*), and feature extraction (like *TfidfVectorizer*). *NLTK* package (Natural Language Toolkit) is used for getting the list of *stopwords*. *Seaborn* and *Matplotlib* packages are used for data visualizations.

# III. Data

## III.1. OTT PLATFORMS DATASETS

---

We have gathered datasets for 4 OTT platforms: *Netflix, Prime, Hulu, and Disney +*.

### NETFLIX

[Netflix Dataset](#) contains a total of 9425 records, each record for a movie or series. 29 features are present in this dataset, out of which only 10 are used.

### AMAZON PRIME

[Amazon Prime Dataset](#) contains a total of 9668 records, each record for a movie or series. 12 features are present in this dataset, out of which only 10 are used.

### HULU

[Hulu Dataset](#) contains a total of 3073 records, each record for a movie or series. 12 features are present in this dataset, out of which only 10 are used. 90% of the records have no information (missing values) for the features *directors, actors*.

### DISNEY+

[Disney+ Dataset](#) contains a total of 1450 records, each record for a movie or series. 12 features are present in this dataset, out of which only 10 are used.

All the 10 features in these 4 datasets are:

*title, genres, content\_type, directors, actors, content\_ratings, release\_year, date\_added, and summary.*

All these 4 datasets are appended to 1 single dataset as saved as a *.csv* file. Data cleaning and transformation are done before saving it to a file, such as filling in missing values, renaming the incorrect values, changing data types, etc. There are now 23533 records in this combined *OTT dataset*.

## III.2. USER RATING DATASETS

---

The user ratings are gathered from 2 sources: the *IMDB Review Dataset* found on Kaggle, *IMDB API*.

### IMDB REVIEW DATASET

[IMDB Review Dataset](#) contains a total of 5,571,499 records, and each record is a review by a user for a movie/series. Out of these 9 features, only 3 features are used

## IV. Aim 1: Genre Prediction

(*reviewer, title, rating*). Out of 23533 titles present in the *OTT dataset*, only 9001 titles have ratings in this *IMDB Review Dataset*.

### IMDB API

The missing values for the *directors and actors* in the *OTT dataset* and user ratings for the remaining 14532 titles in the *OTT dataset* are required. [IMDB-API](#) is used for this purpose.

*SearchTitle*, *Review*, and *FullCast* API from the *IMDB – API* are used. *SearchTitle* API is used to find the *IMDB title\_id* for any given *title* (passing the '*title year*' format helps get better results). This is required because *Review* and *FullCast* API takes input for *title\_id*, not the *title* itself. First, for the *titles* that require either information on *directors, actors*, or *user ratings*, obtaining the *title\_id* is required. Once the *title\_ids* are gathered, we can obtain *directors', actors'* information from the *FullCast* API and user ratings from the *Review* API. The gathered *title\_ids* are saved as a separate column in the *OTT* dataset. The *directors and actors* are updated in the *OTT* dataset, and the results are saved to the same *.csv* file. The user ratings are saved in separate *.csv* files, which consists of features *username, rating, title\_id*.

All the *IMDB* files generated to store the user ratings and *IMDB Review Dataset* downloaded from Kaggle are concatenated into one single *.csv* file. This file consists of around 1,600,000 records. This new *User Ratings* CSV file consists of features *reviewer, rating, title\_index* where *title\_index* is the row index in which that *title* in *OTT dataset* is located.

The *OTT dataset* and *User Rating* dataset are used from this point throughout the project.

## IV. Aim 1: Genre Prediction

### IV.1. APPROACH

---

There are some movies/series for which the genres are missing or unknown in the dataset. We aim to predict these missing genres for the movies/series based on their summary. Genre Prediction is a **multi-class multi-label** classification problem. We built a Genre Prediction model using Natural Language Processing (*NLP*). For training and testing the machine learning models, movies/series that have missing genres are separated from the dataset temporarily.

## IV. Aim 1: Genre Prediction

There are 93 unique genres present in our dataset. These genres are reduced to 10 genre categories: *Thriller, Documentary, Fiction, Comedy, Family, Horror, Drama, Action, Arts & Culture, and Other*. `MultiLabelBinarizer`<sup>[5]</sup> class from the package `sklearn.preprocessing` generates a binary matrix for these 10 genres. These 10 genres represent the *classes* for the classification model.

The summaries for all the movies/series present in the dataset are stripped of symbols, stop words and are tokenized into words. The *TF – IDF* scores<sup>[4]</sup> (citation here maybe) are calculated for all the unique words present in these summaries using the `TfidfVectorizer` from the `sklearn.feature_extraction.text` package.

## IV.2. EXPERIMENTS

---

The *features* and *classes* are split into training and testing sets using `train_test_split` from the `sklearn` package. The size of the test set is 20% of the whole *features* and *classes*.

***OneVsRestClassifier*** (*OvR*) from the `sklearn` package is used for genre prediction. This classifier takes a parameter *estimator*, which implements the *fit* operation.

5 machine learning models are passed as *estimator* for the *OneVsRestClassifier*: Logistic Regression, Bernoulli Naïve Bayes (*BernoulliNB*), Multinomial Naïve Bayes (*MultinomialNB*), Stochastic Gradient Descent (*SGDClassifier*) and Linear Support Vector Classifier (*LinearSVC*).

The parameters for each of these machine learning models are shown in *Figure IV.2.1*:

```
models=[OneVsRestClassifier(LogisticRegression(random_state=random_state_value, solver='liblinear', \
                                              multi_class="ovr")),
        OneVsRestClassifier(BernoulliNB()),
        OneVsRestClassifier(MultinomialNB()),
        OneVsRestClassifier(SGDClassifier(random_state=random_state_value)),
        OneVsRestClassifier(LinearSVC(multi_class="ovr", random_state=random_state_value))]
```

*Figure IV.2.1*

These models are trained and tested using the training and testing sets. Precision scores and hamming loss measures are calculated for the predictions made by each model. Precision scores are also calculated genre-wise for each model. The time taken to run all these models is around 20 minutes.

Once a model is chosen, we predict the genres for movies/series with the missing genres.

## IV. Aim 1: Genre Prediction

The scripts are written in *Python 3.9.7* version and ran on *Jupyter Notebook 6.0.1* version. The machine configuration on which the experiments were run on is shown in *Figure IV.2.2*.

Processor	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Installed RAM	8.00 GB (7.88 GB usable)

Figure IV.2.2

## IV.3. RESULTS AND DISCUSSIONS

### IV.3.1. RESULTS 1

The precision scores and hamming loss values for each model used as *an estimator* for *OneVsRestClassifier* are shown in *Figure IV.3.1*.

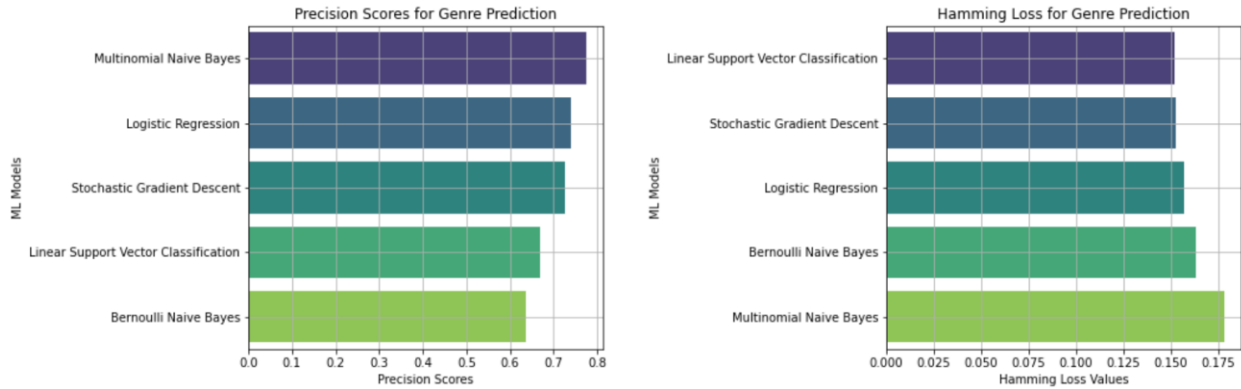


Figure IV.3.1

*Multinomial Naïve Bayes* has the highest precision score but also has the highest hamming loss. The next best model according to the precision scores is *Logistic Regression* which has a decent hamming loss value (3<sup>rd</sup> place from the top). The next best model according to the hamming loss values is *Stochastic Gradient Descent* which has a decent precision score (3<sup>rd</sup> place from the top). *Logistic Regression* or *Stochastic Gradient Descent* can be used for predicting the missing genres.

### IV.3.2. RESULTS 2

The genre-wise precision scores for each model are shown in *Figure IV.3.2*.

*The multinomial Naïve Bayes* model has the best precision score for 2 genres, but it also has the worst precision score for *Arts and Culture* genre. The precision scores for the *Bernoulli Naïve Bayes* model for genres *Arts and Culture* and *Horror* are pretty

## IV. Aim 1: Genre Prediction

low compared to other models. The models with better precision scores genre-wise are *Logistic Regression* and *Stochastic Gradient Descent*.

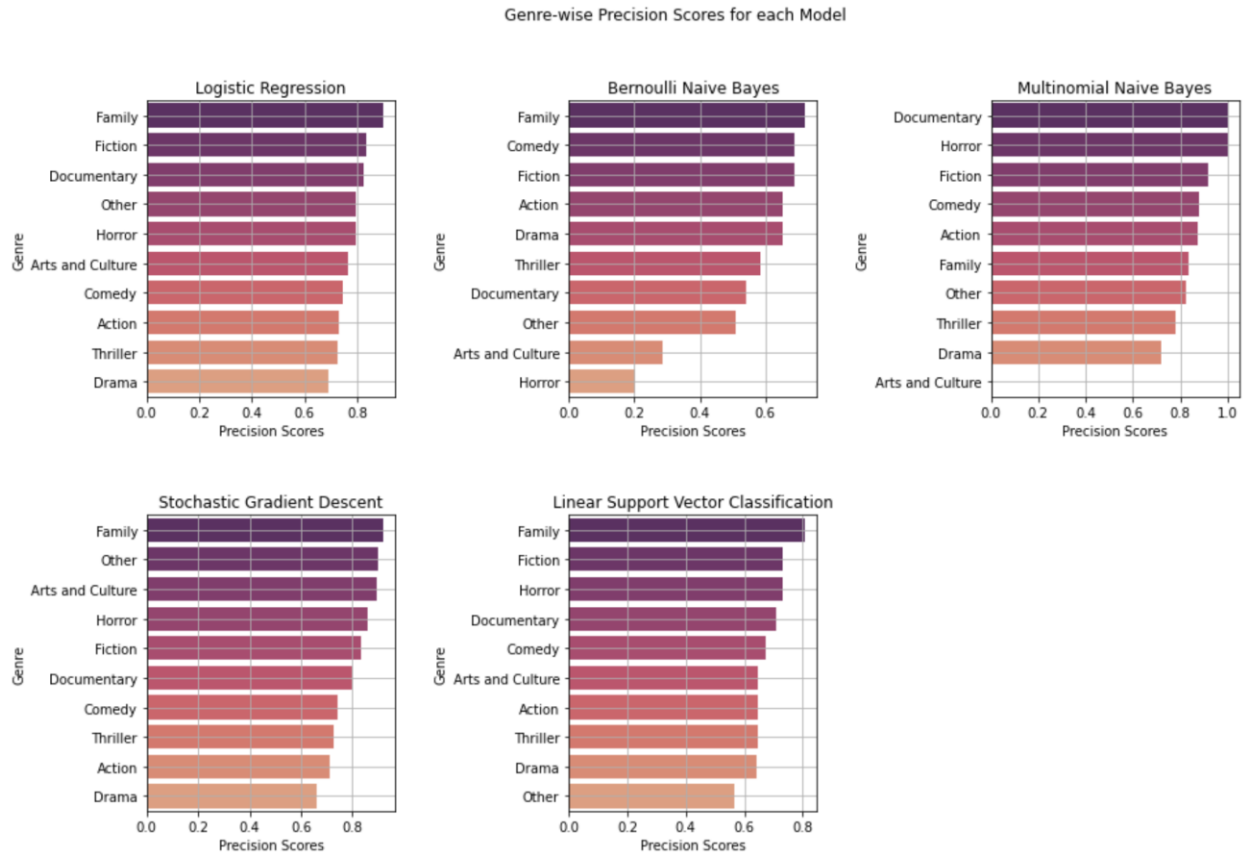


Figure IV.3.2

*Logistic Regression* precision scores are better than that of *Stochastic Gradient Descent* by a small difference, we've chosen ***Logistic Regression*** to predict the missing genres.

### IV.3.3. RESULTS 3

```

Action          1
Arts and Culture 0
Comedy          0
Documentary     0
Drama          0
Family          0
Fiction         0
Horror          0
Other           0
Thriller        1
title_index     169
title           Partners: The Movie IV
release_year    2017
content_type    Movie
Name: 0, dtype: object

```



#### Partners: The Movie IV

A mysterious international crime organization from the Japanese government for the release targets a victory parade of a world sports game

Also Known As: 相棒 劇場版IV / 相棒 劇場版IV 係 最後の決断 / Aibou Gekijo-ban IV / Aibou: T

Director: Hashimoto Hajime [橋本一]

Country: Japanese

Status: Completed

Genres: Action; Crime; Mystery; Suspense;

Airs: 2017

Watch Now

Figure IV.3.3: The image on the left shows the genres predicted for 'Partners: The Movie IV' by the *Logistic Regression* model, and the image on the right shows the genres listed for the same movie on the Internet.



## V. Aim 2: Collaborative Filtering

The missing genres are predicted by the *Logistic Regression* model. One of the movies that had missing genres from our dataset is '*Partners:The Movie IV.*' Our model predicted the genres for this movie as *Action,Thriller*. The genres displayed for this movie on the Internet<sup>[1]</sup> are *Action,Mystery, Crime,Suspense*. Since we've categorized *Crime,Mystery,Suspense* into *thrillers*, the genres for this movie have been predicted accurately (*Figure IV.3.3*).

### IV.1.4. CONCLUSIONS

---

The *features* (words in summary) are extracted from the content (movies/series) with missing genres, and *Logistic Regression* is used to predict the genres for these movies/series.

## V. Aim 2: Collaborative Filtering

### V.1. APPROACH

---

*Collaborative filtering* predicts what the user might like according to the user preferences, such as **ratings**, from many users. Due to the lack of user preference data apart from the user ratings, we'll consider that if a user has rated a movie or series, then that user has watched this movie or series and vice versa. So, from the ratings provided by the user for different content, we'll try to predict the ratings for each user for the content the user has not watched yet. The goal for performing collaborative filtering is to predict the missing ratings.

***Surprise*** package<sup>[2]</sup> from the *scikit – learn* library is specially designed for collaborative filtering and is used in this project. This package has to be installed either using *pip* or *conda* before use (*Figure V.1.1*).

```
import sys
!{sys.executable} -m pip install scikit-surprise
```

*Figure V.1.1*

The *surprise* package has 2 classes *Reader* and *Dataset*. The *Reader* takes the range of values the ratings can have. The dataset containing the  $\langle user\_id, title\_id, rating \rangle$  has to be loaded using the *dataset.load\_from\_df* method. This returns the *Dataset* object that can be passed to the machine learning models for collaborative filtering.

## V.2. EXPERIMENTS

---

From the *Surprise* package, 5 machine learning models are used for collaborative filtering: *SVD* (Singular Value Decomposition), *SVD++* (Singular Value Decomposition++), *NMF* (Non-negative Matrix Factorization), *SlopeOne*, *Co-Clustering*. We use *RMSE* (Root Mean Square Error), and *MAE* (Mean Absolute Error) measures to evaluate these models.

Most of the machine learning algorithms for the collaborative filtering take parameters such as

- *n\_factors*: This parameter exists for matrix factorization techniques<sup>[3]</sup> which is the number of latent factors to consider.
- *n\_epochs*: The number of iterations the machine learning algorithm should perform.
- *random\_state*: Random state is to reproduce the results.

### V.2.1. EXPERIMENTS 1

The parameters for each of the 5 machine learning models are shown in *Figure V.2.1*:

```
models=[SVD(n_epochs=10, random_state=random_state_value),
        SVDpp(n_epochs=10, random_state=random_state_value),
        NMF(n_epochs=10, random_state=random_state_value),
        SlopeOne(),
        CoClustering(n_epochs=10, random_state=random_state_value)]
```

*Figure V.2.1*

10 – fold cross – validation is performed for each of these algorithms, and *RMSE* and *MAE* scores are calculated for each model for each validation.

The outputs are the *RMSE*, and *MAE* scores calculated for each model for each validation. The time to run each of these models is around 10 minutes each except for *SVD++*, which takes around 1 hour 45 minutes to run.

### V.2.2. EXPERIMENTS 2

The *Dataset* object is now split into train and test sets using the *train\_test\_split* (present in the *Surprise* package). The size of the test set is 20% of the *Dataset* object.

Next, the *SVD* model is run several times for different parameter values, as shown in *Figure V.2.2*. For each of these *SVD* models, the train set is trained using *fit()*, and predictions are made using a *test()*. The *RMSE* and *MAE* scores are calculated for the predictions and test set for each model. The time to run these *SVD* models is around 15 minutes.

## V. Aim 2: Collaborative Filtering

```
svd_models=[SVD(random_state=random_state_value), # Default - factors=100, epochs=20
            SVD(n_epochs=10, random_state=random_state_value), # Factors=100, epochs=10
            SVD(n_factors=50, random_state=random_state_value), # Factors=50, epochs=20
            SVD(n_factors=50, n_epochs=10, random_state=random_state_value), # Factors=50, epochs=10
            SVD(n_factors=250, n_epochs=10, random_state=random_state_value), # Factors=250, epochs=10
            SVD(n_factors=250, n_epochs=20, random_state=random_state_value), # Factors=250, epochs=20
            SVD(n_epochs=50, random_state=random_state_value), # Factors=100, epochs=50
            SVD(n_factors=250, n_epochs=50, random_state=random_state_value)] # Factors=100, epochs=50
```

Figure V.2.2

The scripts are written in *Python 3.9.7* version and ran on *Jupyter Lab 3.2.1* version. The machine configuration on which the experiments were run on is shown in *Figure V.2.3*.

Processor	Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz
Installed RAM	16.0 GB (15.6 GB usable)

Figure V.2.3

## V.3. RESULTS AND DISCUSSIONS

### V.3.1. RESULTS 1

For the machine learning models run in the *Section V.2.1.*, the *RMSE* scores for all 5 machine learning models are as shown in *Figure V.3.1*.

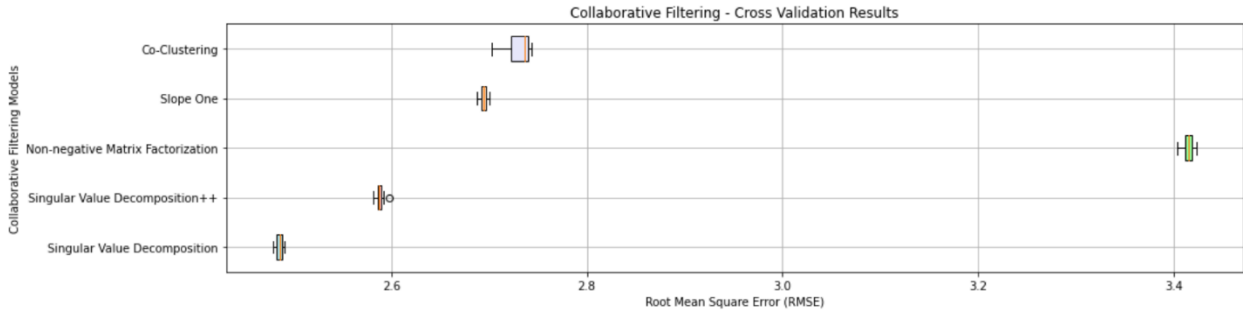


Figure V.3.1

*RMSE* values can range from 0 to *infinity*, and the lower the score, the better is the model. We can see that Singular Value Decomposition (*SVD*) has the lowest *RMSE* score among all the 5 models (a lower score is better). Non-negative Matrix Factorization (*NMF*) has the highest *RMSE* score among these 5 models.

The *MAE* scores for all 5 machine learning models are as shown in *Figure V.3.2*. *MAE* values also range from 0 to infinity, and the lower the score, the better is the model. The *MAE* score is low for the *SVD* model (a lower score is better) and high for the *NMF* model.

## V. Aim 2: Collaborative Filtering

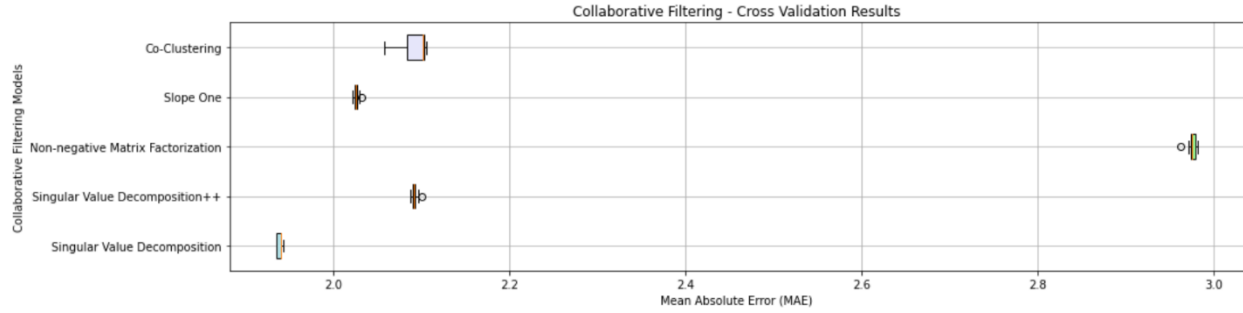


Figure V.3.2

Since *SVD* has the lowest score for both *RMSE* and *MAE* measures, ***SVD*** is chosen for further experimentation.

### V.3.2. RESULTS 2

For the *SVD* models run with different parameters in Section V.2.2., the *RMSE* scores for each *SVD* model are shown in Figure V.3.3.

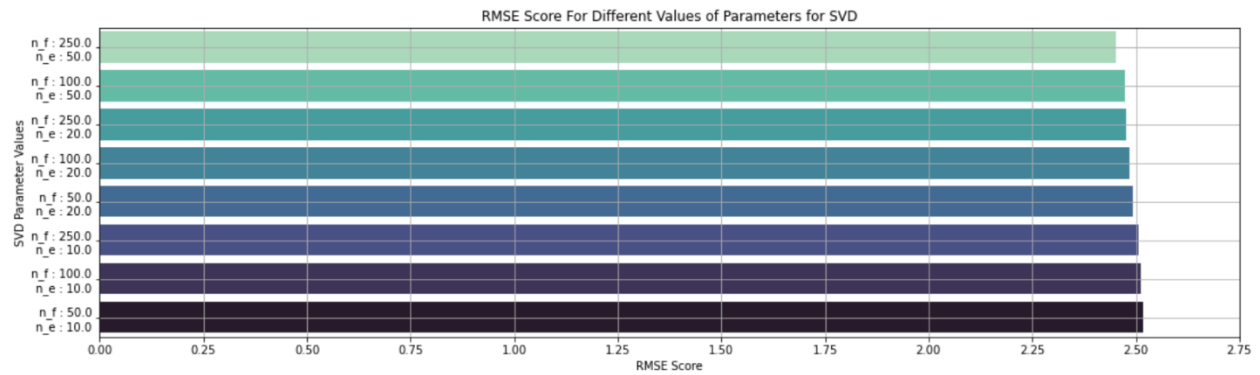


Figure V.3.3

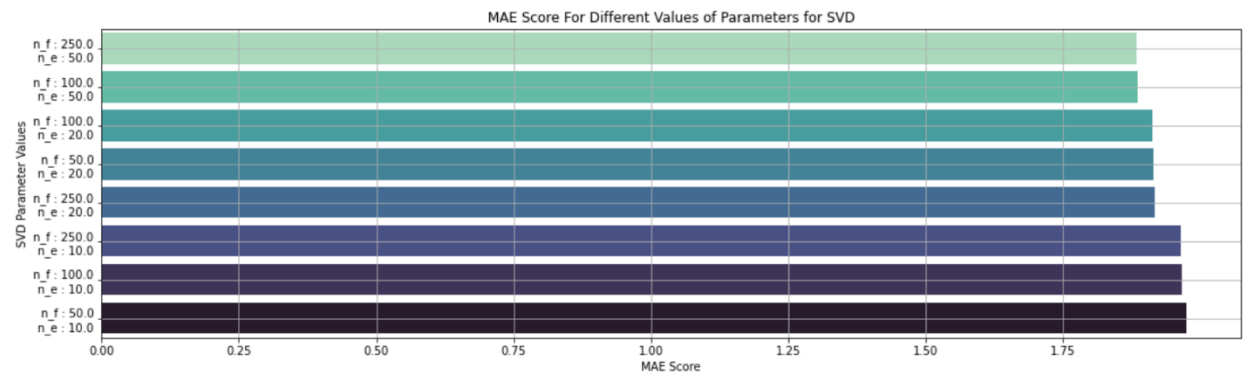


Figure V.3.4

$n_f$  represents a number of factors, and  $n_e$  represents a number of epochs/iterations. The lowest *RMSE* score is for the *SVD* model run for with a number of latent factors equal to 250 and the number of iterations equal to 50. The next best model would be

## V. Aim 2: Collaborative Filtering

the model with a number of factors as 100 and the number of iterations as 50. The difference between the *RMSE* scores for these 2 models is very less. The *MAE* scores for each *SVD* model are shown in *Figure V.3.4*. The lowest *MAE* score is for the *SVD* model run for with a number of latent factors equal to 250 and number of iterations equal to 50, and the model with a number of factors as 100 and number of iterations as 50.

From these results, both models ( $n_f=250$ ,  $n_e=50$  and  $n_f=100$ ,  $n_e=50$ ) can be used on this dataset. In order to not overfit the data, we chose the **SVD** model with *a number of latent factors* = 100 and *number of iterations* = 50.

### V.3.3. RESULTS 3

Now that we have chosen the model to predict the missing ratings between a user and a movie/series, *Figure V.3.5* is an example table that can be constructed. The column names are the movie/series *titles*, and the row indices are the *user\_ids* from the dataset. The cells values that do not start with 'P:' are the ratings given by the user to that content. Those cells whose values start with 'P:' are the predicted ratings by the SVD model with  $n_f=100$  and  $n_e=50$ .

	The Godfather	Inception	Stranger Things	Joker	Twilight
3	10.0	8.0	7.0	8.0	4.0
14	10.0	8.0	10.0	10.0	P: 7.127766626068944
21	10.0	10.0	9.0	P: 10	7.0
22	P: 8.455436414551684	8.0	6.0	4.0	P: 5.853456751175114
289	10.0	P: 7.084465962334621	8.0	P: 9.588707228358931	P: 6.12414000775816

Figure V.3.5

From the above table, we can infer that user 3 and user 289 have rated the titles pretty similarly. Even the predicted ratings came out similar. User 21 seems like an easy rater, and user 22 seems like a tough rater.

### V.4.4. RESULTS 4

	title	content_type	release_year	platforms	predicted_rating
12992	#Unfit: The Psychology of Donald Trump	Movie	2020	Prime	10.000000
1210	Eurovision Song Contest: The Story of Fire Saga	Movie	2020	Netflix	10.000000
9038	The Hangover	Movie	2009	Netflix	9.988466
20585	Maxxx	Movie	2020	Hulu	9.939500
6720	Hum Aapke Hain Koun	Movie	1994	Netflix	9.938368

Figure V.3.6

## VI. Aim 3: Plot Filtering

There is a user ‘*Leofwine\_draca*’ from the dataset. According to the ratings this user has provided, the next 5 movies which the user could enjoy (sorted by predicted ratings) are shown in *Figure V.3.6*.

## V.4. CONCLUSIONS

---

The *SVD* model with a number of latent factors equal to 100 and the number of iterations equal to 50 works well on this dataset. Providing recommendations simply based on user ratings is not enough. Therefore, plot and content-based filtering are applied along with collaborative filtering.

# VI. Aim 3: Plot Filtering

## VI.1. APPROACH

---

Plot filtering recommends content based on the ***plot*** or ***summary*** of the movie/series. Preferences of other users are not considered in this filtering. Since *plots* are in *text* format, we can generate a matrix of ***TF – IDF*** scores and predict similar content using the similarity measures.

Term frequency gives the importance of the term in a single document and scales it by its importance across all documents using IDF. The mathematical equation of ***TF – IDF*** is

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \text{ where, } t = \text{term and } d = \text{document and } D = \text{set of documents.}$$

Before computing the ***TF – IDF*** scores, *stopwords* should be removed from the *plots* of the movies/series present in the dataset. The stop words are removed using the *NLTK* (Natural Language Toolkit) package and should be downloaded from this package before use (*Figure VI.1.1*).

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

*Figure VI.1.1*

## VI.2. EXPERIMENTS

---

### VI.2.1. EXPERIMENTS 1 – DATA ANALYSIS

The top 10 words with the highest mean of the  $TF - IDF$  scores are calculated and visualized in the *Results* section.


### VI.2.2. EXPERIMENTS 2

First, the summaries for all the movies/series present in the dataset are stripped of symbols, stop words and are tokenized into words. The  $TF - IDF$  scores (citation here maybe) are calculated for all the unique words present in these summaries using the *TfidfVectorizer* from the *sklearn.feature\_extraction.text* package. These scores are used to *fit* and *transform* the words to a document-term matrix.

$TF - IDF$  has in-built stop words, but it has several known issues with '*english*,' and the documentation for  $TF - IDF$  suggests us to consider an alternative<sup>[4]</sup>, which is why we are using the *NLTK* package.

*Pairwise cosine similarity* from the *sklearn* package is used to calculate the similarities between each movie/series with other movie/series based on the  $TF - IDF$  scores. Computation of  $TF - IDF$  matrix and cosine similarities take around 8 minutes.

The scripts are written in *Python* 3.8.8 version and ran on *Jupyter Notebook* 6.3.0 version. The machine configuration on which the experiments were run on is shown in *Figure VI.2.1*.



MacBook Air (M1, 2020)  
Chip Apple M1  
Memory 8 GB

*Figure VI.2.1*

## VI.3. RESULTS AND DISCUSSIONS

---

### VI.3.1. RESULTS 1 – DATA ANALYSIS

*Figure VI.3.1* shows the top 10 words with the highest mean of the  $TF - IDF$  scores calculated for each word. Most of the top 10 words look like words that are usually important in one's life.

## VI. Aim 3: Plot Filtering

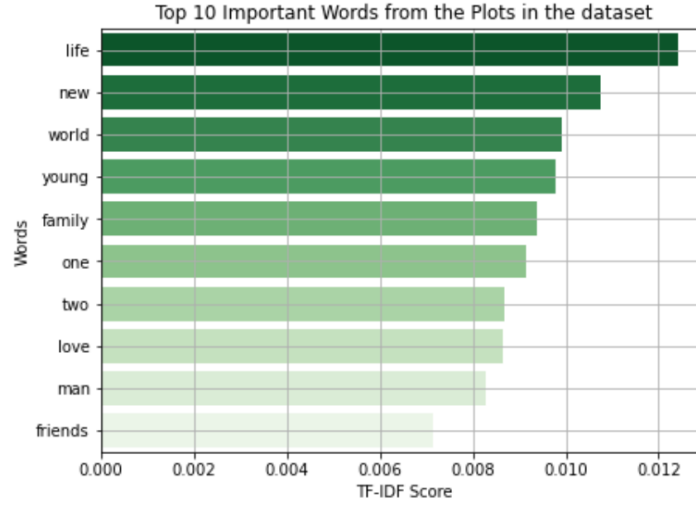


Figure VI.3.1

### VI.3.2. RESULTS 2

The matrix in Figure VI.3.2 consisting of  $TF - IDF$  scores generated for the top 10 important words for the first 5 movies/series from our dataset.

	life	new	world	young	family	one	two	love	man	friends
title										
Lets Fight Ghost	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000
HOW TO BUILD A GIRL	0.000000	0.000000	0.0	0.000000	0.131769	0.0	0.000000	0.000000	0.0	0.000000
The Con-Heartist	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000
Gleboka woda	0.000000	0.138662	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000
Only a Mother	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.148718	0.0	0.000000

Figure VI.3.2 <sup>(i)</sup>

Figure VI.3.3 is a small part of the cosine similarity matrix generated based on the  $TF - IDF$  matrix generated for the plots from our dataset. Each index and column name represents the movie/series present in that row index. The  $cell(i, j)$  contains the cosine similarity between movie/series  $i$  and movie/series  $j$ .

<sup>i</sup> \*This represents a small part of the  $TF - IDF$  matrix generated for the plots from our dataset



## VI. Aim 3: Plot Filtering

	13675	2323	15155	15246	2280	2257	2251	15451	2239	15506
13675	1.000000	0.000000	0.001120	0.002918	0.002631	0.000000	0.105952	0.000000	0.000000	0.000000
2323	0.000000	1.000000	0.000000	0.092450	0.087706	0.000000	0.280837	0.000000	0.087706	0.104828
15155	0.001120	0.000000	1.000000	0.000000	0.001162	0.000000	0.000000	0.004119	0.000898	0.000883
15246	0.002918	0.092450	0.000000	1.000000	0.105409	0.000000	0.087722	0.100546	0.000000	0.003949
2280	0.002631	0.087706	0.001162	0.105409	1.000000	0.145442	0.237171	0.091287	0.104995	0.119523

Figure VI.3.3

### VI.3.3. RESULTS 3

Figure VI.3.4 contains some details for the movie *"The Shawshank Redemption"* from the dataset.

	title	plot	platforms
8649	The Shawshank Redemption	Framed for murder, upstanding banker Andy Dufresne begins a new life at the Shawshank prison and gradually forms a close bond with older inmate Red.	Netflix

Figure VI.3.4

Plot-based filtering recommends the movies/series shown in Figure VI.3.5 for *"The Shawshank Redemption."* We can see some keywords like *murder*, *life*, *prison*, and *banker* are in the plot of the given movie and are also present in the plots of the recommended content.

	title	plot	platforms
6016	Battles Without Honor and Humanity	A former Japanese soldier goes to prison for murder and becomes sworn brothers with a yakuza inmate. Soon, he's swept into a violent street war.	Netflix
417	Me & You vs The World	When a young man launches an outdoor adventure program, he forms a bond with a student that opens both of them up to new possibilities.	Netflix
3183	Chameli	While taking shelter from a Mumbai monsoon, an investment banker forms an unlikely connection with a naïve prostitute over the course of the night.	Netflix
21801	A Man Alone	Framed for a robbery, a gunman hides out in the home of the local sheriff and gradually wins his trust and respect. Now they must face the vengeful townspeople in a bloody war.	Hulu
6872	Cell 211	Knocked out during his first day on the job, a prison guard awakens to find that his co-workers have abandoned him in the midst of an inmate uprising.	Netflix

Figure VI.3.5

## VI.4. CONCLUSIONS

Using *TF – IDF* scores after removing the *stopwords* from the *plots* does recommend good results for a given movie/series. Applying plot filtering along with collaborative and content-based filtering will produce results specific to each user.

# VII. Aim 4: Content-based Filtering

## VII.1. APPROACH

---

Content-based filtering predicts similar content for a given movie/series. Preferences of other users are not involved in this filtering.

We will try to predict similar content from the features of a movie/series like ***genres,actors,directors***, and ***content\_ratings***. Since these features are non-numerical, we need to convert them into a standard mathematical representation, such as **binary** representation, and predict similar movies using the similarity functions.

## VII.2. EXPERIMENTS

---

### VII.2.1. EXPERIMENTS 1 – DATA ANALYSIS

The values for *genres,directors*, and *actors* in our dataset are comma-separated strings. Extracting the unique values in each feature took around 1 minute in total. There are 93 *genres*, 10 *content ratings*, 13500 directors, and 53266 actors present in our dataset.

The top 10 values for each feature (based on the number of movies/series) are visualized and shown in the *Results* section.

### VII.2.2. EXPERIMENTS 2

We have generated the binary matrices for *genres,directors,actors*, and *content rating* features using the **Multi – Label Binarizer**<sup>[5]</sup> from the *sklearn* package.

The size of the binary matrix for each feature is *a number of movies/series × number of unique values in the feature*, i.e., the size of the binary matrix generated by the *genre* feature is  $23533 \times 93$ . These matrices are merged into 1 single matrix of size  $23533 \times (93 + 10 + 13500 + 53266) = 23533 \times 66869$ .

*Pairwise cosine similarity* from the *sklearn* package is used to calculate the similarities between each movie/series with other movie/series based the merged binary matrix. Computation of binary matrices and cosine similarities takes around 10 minutes (*Figure VI.2.1*).

## VII. Aim 4: Content-based Filtering

```
def generateMatrices(data):  
    binary=pd.concat([makeBinaryDFs(data, 'genres'), makeBinaryDFs(data, 'directors'), \  
        makeBinaryDFs(data, 'actors'), makeBinaryDFs(data, 'content_rating')], axis=1)  
    return pd.DataFrame(cosine_similarity(binary), index=binary.index, columns=binary.index)
```

Figure VI.2.1

The scripts are written in *Python 3.8.8* version and ran on *Jupyter Notebook 6.3.0* version. The machine configuration on which the experiments were run on is shown in *Figure VI.2.2*.

MacBook Air (M1, 2020)  
Chip Apple M1  
Memory 8 GB

Figure VI.2.2

## VII.3. RESULTS AND DISCUSSIONS

### VII.3.1. RESULTS 1 – DATA ANALYSIS

There are 93 genres in our dataset, with the *Drama* genre at the top with the most content (*Figure VII.3.1*). There are 10 content ratings in our dataset, with an *Unrated* rating at the top with the most content (*Figure VII.3.1*).

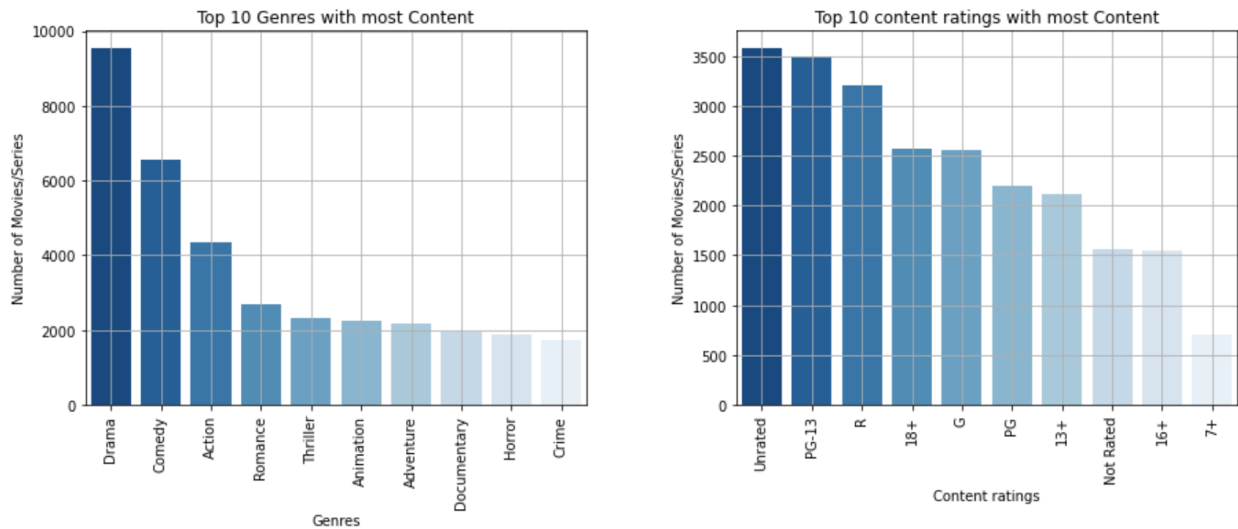


Figure VII.3.1

There are 13500 directors present in our dataset. In *Figure VII.3.2*, the director at the top with the most content is '*Mark Knight*.' There is something interesting about this. *Mark Knight* is actually not a regular film or TV series director but has actually directed a lot of health and fitness videos which are available on the *Amazon Prime* platform, which fall under *Fitness* and *Special Interest* genres.

## VII. Aim 4: Content-based Filtering

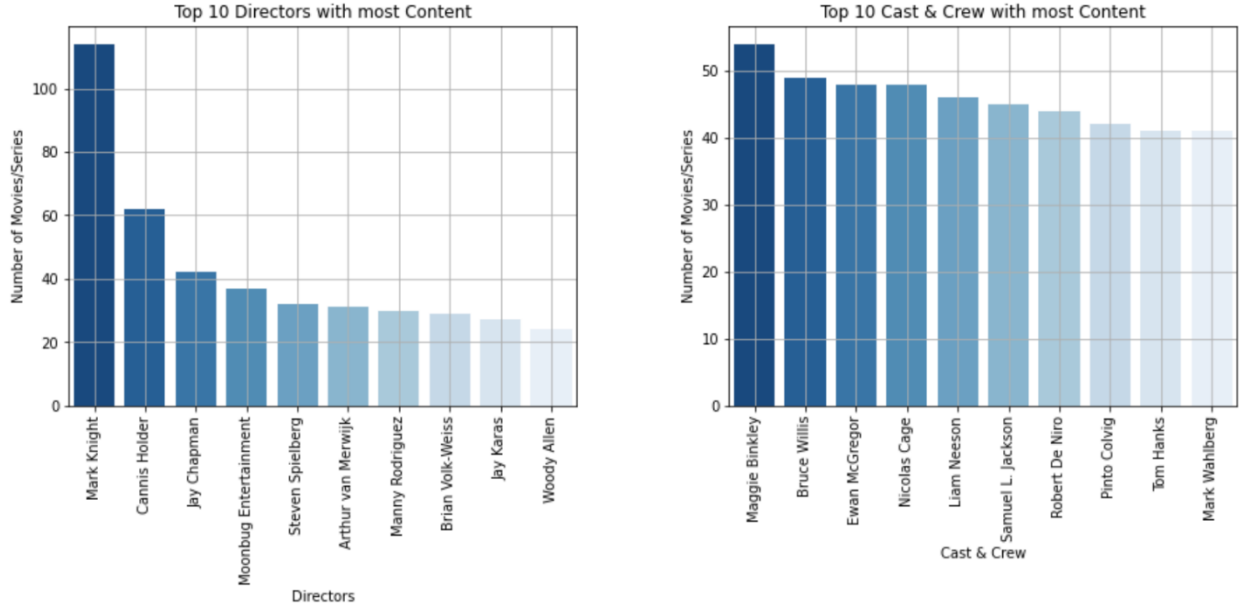


Figure VII.3.2

There are 53266 actors present in our dataset. In *Figure VII.3.2*, the top result for *Cast & Crew* is '*Maggie Binkley*', who is actually not an actress but a fitness expert. There are a number of videos of *Maggie Binkley* on the *Amazon Prime* platform categorized under the *Fitness* genre. The top 8<sup>th</sup> result is '*Pinto Colvig*,' who's an American voice actor. He has appeared in many children's animation movies and has played famous roles like *Pluto* and *Goofy*.

### VII.3.2. RESULTS 2

*Figure VII.3.3* is a table generated from the binary matrix generated for the *genres* present in our dataset. The columns are some of the *genres* present in our dataset, and the indices are the *row ids* of movies/series present in our dataset.

	Action	Drama	Thriller	Variety	Comedy
2	0	0	0	0	1
4	0	1	0	0	0
10	1	0	0	0	0
256	0	1	1	0	1
3822	0	0	0	0	0

Figure VII.3.3

Similar binary matrices are produced for the other features. *Figure VII.3.4* is a table generated from the cosine similarities computed between each movie/series with

## VII. Aim 4: Content-based Filtering

another movie/series. Each index and column name represents the movie/series present in that row index. The  $cell(i,j)$  contains the cosine similarity between movie/series  $i$  and movie/series  $j$ .

	0	1	2	3	4	5	6	7	8
0	1.000000	0.113961	0.106600	0.100504	0.113961	0.000000	0.272727	0.113961	0.100504
1	0.113961	1.000000	0.133631	0.000000	0.000000	0.142857	0.000000	0.000000	0.000000
2	0.106600	0.133631	1.000000	0.117851	0.133631	0.267261	0.000000	0.133631	0.117851
3	0.100504	0.000000	0.117851	1.000000	0.251976	0.125988	0.100504	0.251976	0.222222
4	0.113961	0.000000	0.133631	0.251976	1.000000	0.142857	0.113961	0.285714	0.251976

Figure VII.3.4

### VII.3.3. RESULTS 3

Figure VII.3.5 holds some details regarding the movie “Back to the Future” present in our dataset.

	title	genres	content_type	directors	actors	content_rating	release_year	platforms
8767	Back to the Future	Adventure, Comedy, Sci-Fi	Movie	Robert Zemeckis	Michael J. Fox, Lea Thompson, Crispin Glover, Christopher Lloyd	PG	1985	Netflix

Figure VII.3.5

The content-based filtering algorithm recommends the movies shown in Figure VII.3.6.

	title	genres	content_type	directors	actors	content_rating	release_year	plot	platforms	cos_sim
8835	Back to the Future Part II	Adventure, Comedy, Sci-Fi	Movie	Robert Zemeckis	Michael J. Fox, Lea Thompson, Christopher Lloyd, Thomas F. Wilson	PG	1989	Marty and Doc are at it again in this sequel to the 1985 blockbuster as the time-traveling duo head to 2015 to nip some McFly family woes in the bud.	Netflix	0.888889
8820	Back to the Future Part III	Adventure, Comedy, Sci-Fi, Western	Movie	Robert Zemeckis	Michael J. Fox, Mary Steenburgen, Christopher Lloyd, Thomas F. Wilson	PG	1990	The final installment of the trilogy finds Marty digging the trusty DeLorean out of a mineshaft and looking up Doc in the Wild West of 1885.	Netflix	0.737865
7958	Galaxy Quest	Adventure, Comedy, Sci-Fi	Movie	Dean Parisot	Sigourney Weaver, Tony Shalhoub, Tim Allen, Alan Rickman	PG	1999	Decades after the success of a sci-fi series, the shows washed-up stars are recruited by actual aliens to pull off an intergalactic rescue mission.	Netflix	0.444444
7234	Baby Geniuses	Comedy, Crime, Family, Sci-Fi	Movie	Bob Clark	Peter MacNicol, Christopher Lloyd, Kim Cattrall, Kathleen Turner	PG	1999	Two evil adults try to exploit babies who are born with knowledge of the universes secrets -- and can speak to each other in a secret language.	Netflix	0.421637
167	Sonic the Hedgehog	Action, Adventure, Comedy, Sci-Fi	Movie	Jeff Fowler	Jim Carrey, Ben Schwartz, Tika Sumpter, James Marsden	PG	2020	A small-town sheriff helps an alien hedgehog with supersonic speed outrun a mad doctor who wants the creatures special powers to dominate the world.	Netflix	0.421637

Figure VII.3.6

We can see a lot of similarities between the recommended content and the given movie. Genres such as *Adventure, Comedy, Sci – Fi* from the recommended content

## VIII. Final Recommendation

match with the listed genres in the given movie. Similarly, director *Robert Zemeckis*, actors *Michael J. Fox Christopher Lloyd, Thomas F. Wilson*, and content rating *PG* from the recommended content also match with the ones listed in the given movie.

The top 2 results are sequels to the given movie, and the cosine similarity values are high compared to the other recommended movies, which are quite common in recommendation systems.

### VII.4. CONCLUSIONS

---

Using features like *genres, actors, directors*, and *content ratings* does recommend good results for a given movie/series. Applying content-based filtering along with collaborative and plot filtering will produce results specific to each user.

## VIII. Final Recommendation

### VIII.1. APPROACH

---

The top 10000 results recommended by the collaborative filtering are passed down to the plot and content-based filtering. The  $TF - IDF$  scores [\[4\]](#) will be computed for the words in the *plots* of the 10000 results passed to the plot filtering. The binary matrices for *genres, actors, directors*, and *content ratings* will also be computed for the top 5000 results passed to the content-based filtering. The  $TF - IDF$  matrix and all the binary matrices will be merged into 1 matrix, say  $M$ .

Pairwise cosine similarities will be now computed using the matrix  $M$ . The cosine similarities will be sorted in descending order, and the top  $n$  results will be recommended to the user.

### VIII.2. RESULT AND DISCUSSION

---

In *Figure VIII.2.1*, we're trying to recommend movies or series similar to '*Back to the Future*' to the user 'arishsankar.'

```
recommender("arishsankar", "Back to the Future")
```

*Figure VIII.2.1*

## VIII. Final Recommendation

We can see the recommended content in *Figure VIII.2.2*.

	title	genres	content_type	directors	actors	content_rating	release_year	summary	platforms
8835	Back to the Future Part II	Adventure, Comedy, Sci-Fi	Movie	Robert Zemeckis	Michael J. Fox, Lea Thompson, Christopher Lloyd...	PG	1989	Marty and Doc are at it again in this sequel t...	Netflix
7958	Galaxy Quest	Adventure, Comedy, Sci-Fi	Movie	Dean Parisot	Sigourney Weaver, Tony Shalhoub, Tim Allen, Al...	PG	1999	Decades after the success of a sci-fi series, ...	Netflix
167	Sonic the Hedgehog	Action, Adventure, Comedy, Sci-Fi	Movie	Jeff Fowler	Jim Carrey, Ben Schwartz, Tika Sumpter, James ...	PG	2020	A small-town sheriff helps an alien hedgehog w...	Netflix
8350	Stuart Little	Adventure, Comedy, Family, Fantasy	Movie	Rob Minkoff	Michael J. Fox, Hugh Laurie, Jonathan Lipnicki...	PG	1999	Mr. and Mrs. Little want to adopt a brother fo...	Netflix
23496	Who Framed Roger Rabbit	Action-Adventure, Animation, Comedy	Movie	Robert Zemeckis	Bob Hoskins, Christopher Lloyd, Stubby Kaye, J...	PG	1988	A juicy mystery filled with loony fun pairs a ...	Disney

*Figure VIII.2.2*

We can see that *Back to the Future*'s third sequel movie (*Back to the Future Part III*) is not in the top 5 results. This could happen because we've taken user preferences (here, *rating*) in the account. *Back to the Future Part III* movie takes place in the 18<sup>th</sup> century (unlike *Back to the Future* Part 1 and 2), and the user might not be fond of historical movies.

## VIII.3. CONCLUSION

The quality of recommendation has been improved after merging the collaborative, plot, and content-based filtering. The results of our recommendation system are pretty good. With the presence of more user preferences data like *purchases*, *likes*, *bookmarked*, *skipped*, etc., the recommendation system can be improved.

# Citations

- [1] <https://dramacool.news/asian-wiki/partners-the-movie-iv.html>
- [2] <http://surpriselib.com/>
- [3] [https://en.wikipedia.org/wiki/Matrix\\_factorization\\_\(recommender\\_systems\)](https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems))
- [4] [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.TfidfVectorizer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)
- [5] <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html>