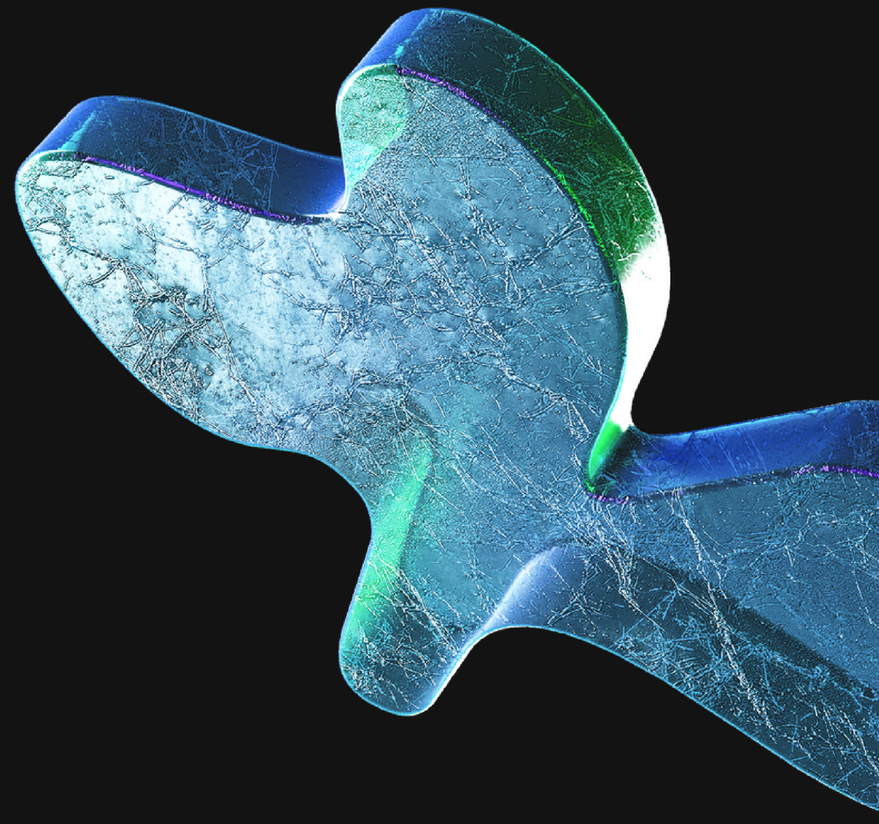
Báo cáo cuối kỳ

Million Songs Dataset Recommendation System



Báo cáo cuối kỳ

Môn học: Hệ thống tư vấn

Giảng viên: Huỳnh Thanh Sơn

Thành viên:

- 20280020 - Huỳnh Việt Dũng
- 20280094 - Lê Hoài Thương
- 20280095 - Nguyễn Ngọc Anh Thy

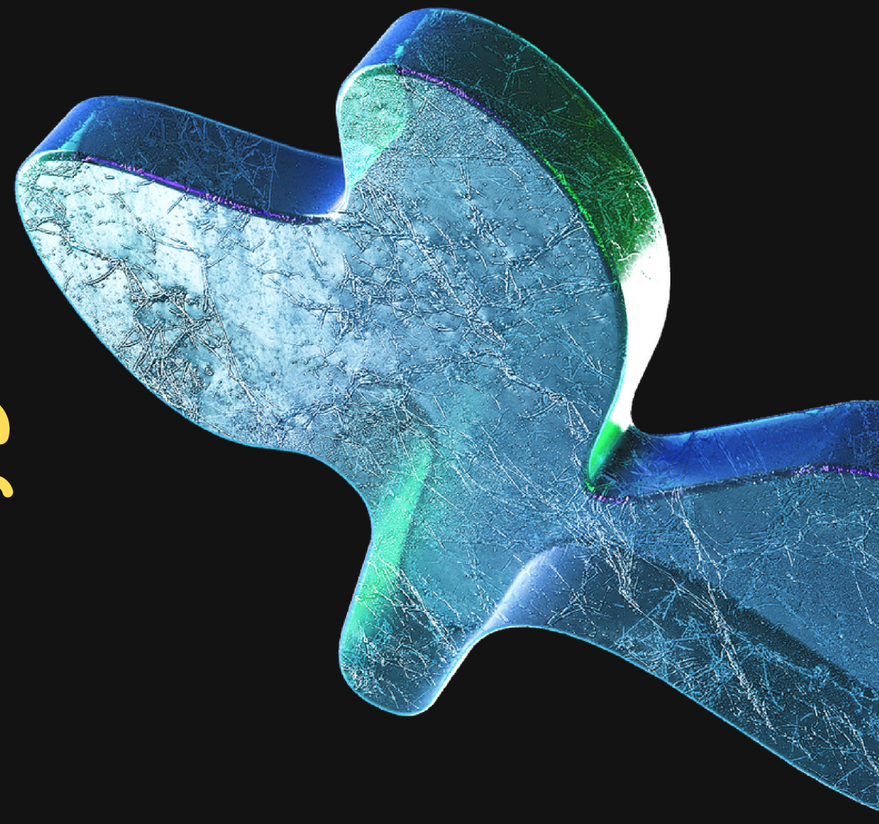
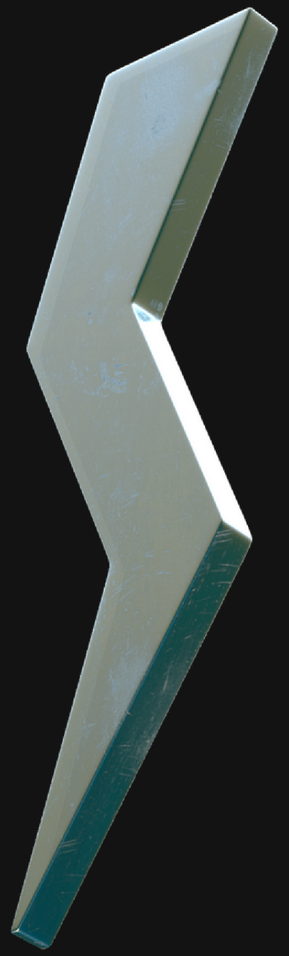


Table of Contents



1. Introduction

PROBLEM

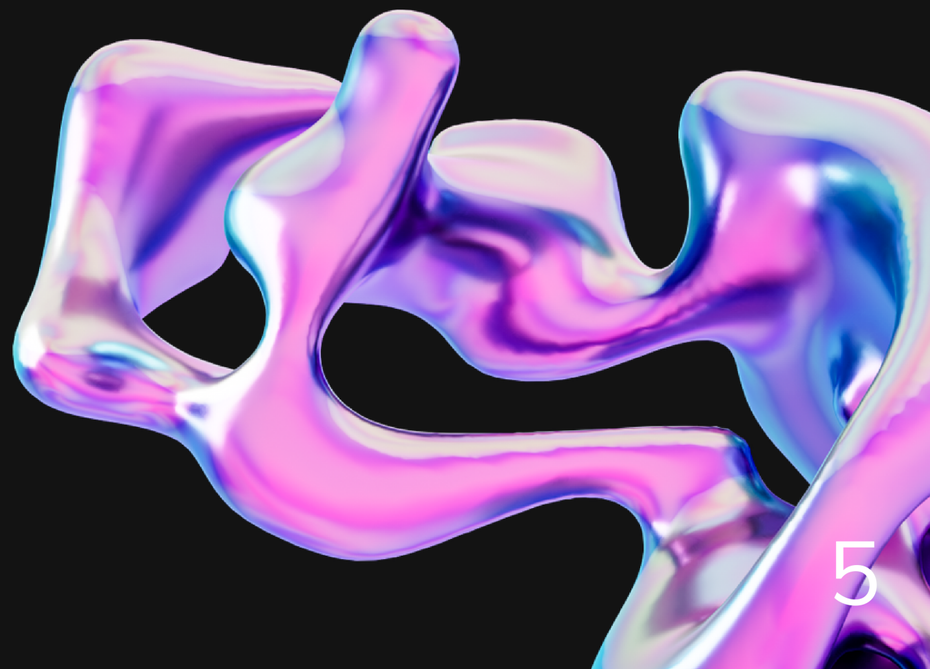
Today we live in a world of rapid technological advancements. Distractions hence limited time to consume good content.

Most people enjoy listening to music and almost every internet-based company's revenue relies on the time consumers spend on it's platform. These companies need to be able to figure out what kind of content is needed in order to increase customer time spent and make their experience better.

1. Introduction

WHAT IS RECOMMENDATION SYSTEMS?

Recommendation systems are a way of modeling and rearranging information available about user preferences and then using this information to provide informed recommendations on the basis of that information.



2. Preparing environment and uploading data

Language: python
Library: Numpy, Pandas, Matplotlib, Seaborn, Sklearn, Collections, Surprise, Nltk, Re

DATASET

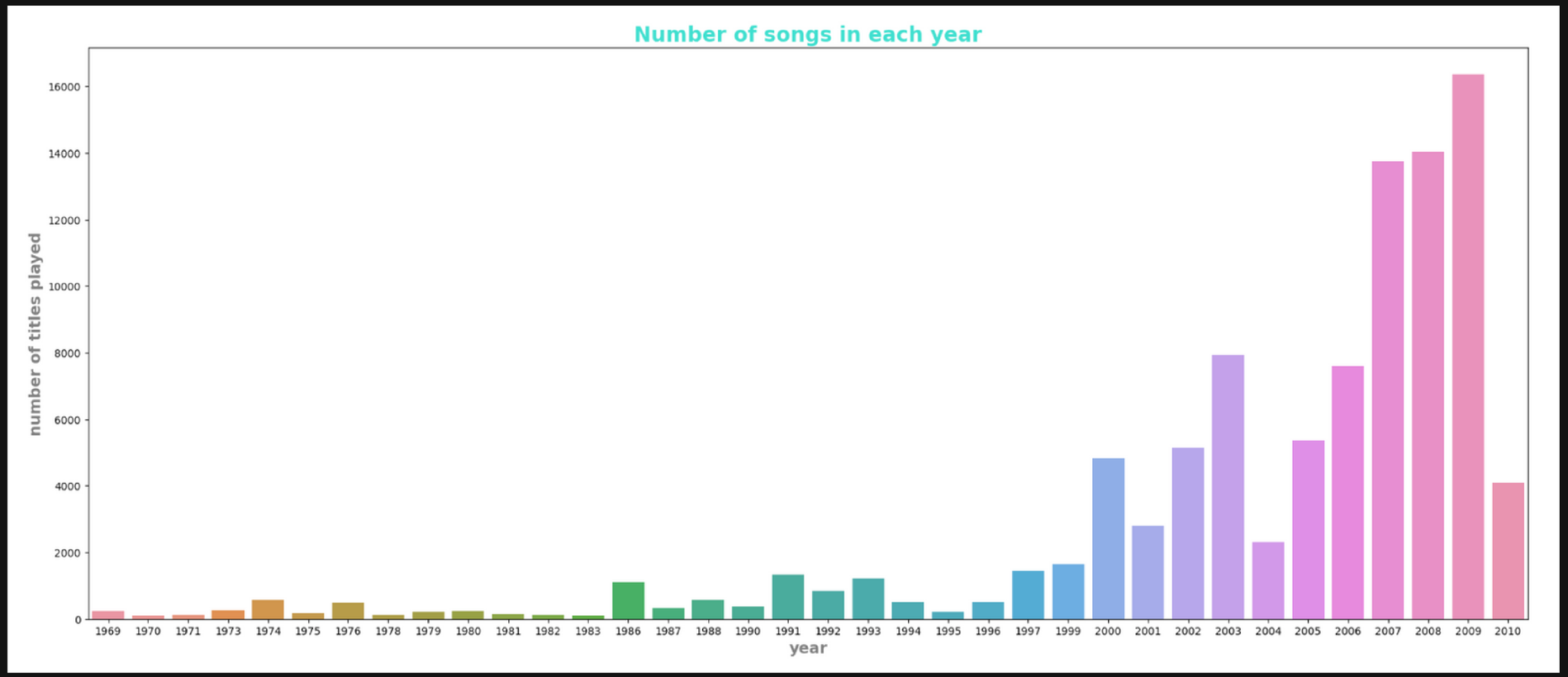
song_data : (1,000,000 records)
song_id - A unique id given to every song
title - Title of the song
Release - Name of the released album
Artist_name - Name of the artist
year - Year of release

count_data : (2,000,000 records)
user _id - A unique id given to the user
song_id - A unique id given to the song
play_count - Number of times was played

Data Source*:
http://millionsongdataset.com/

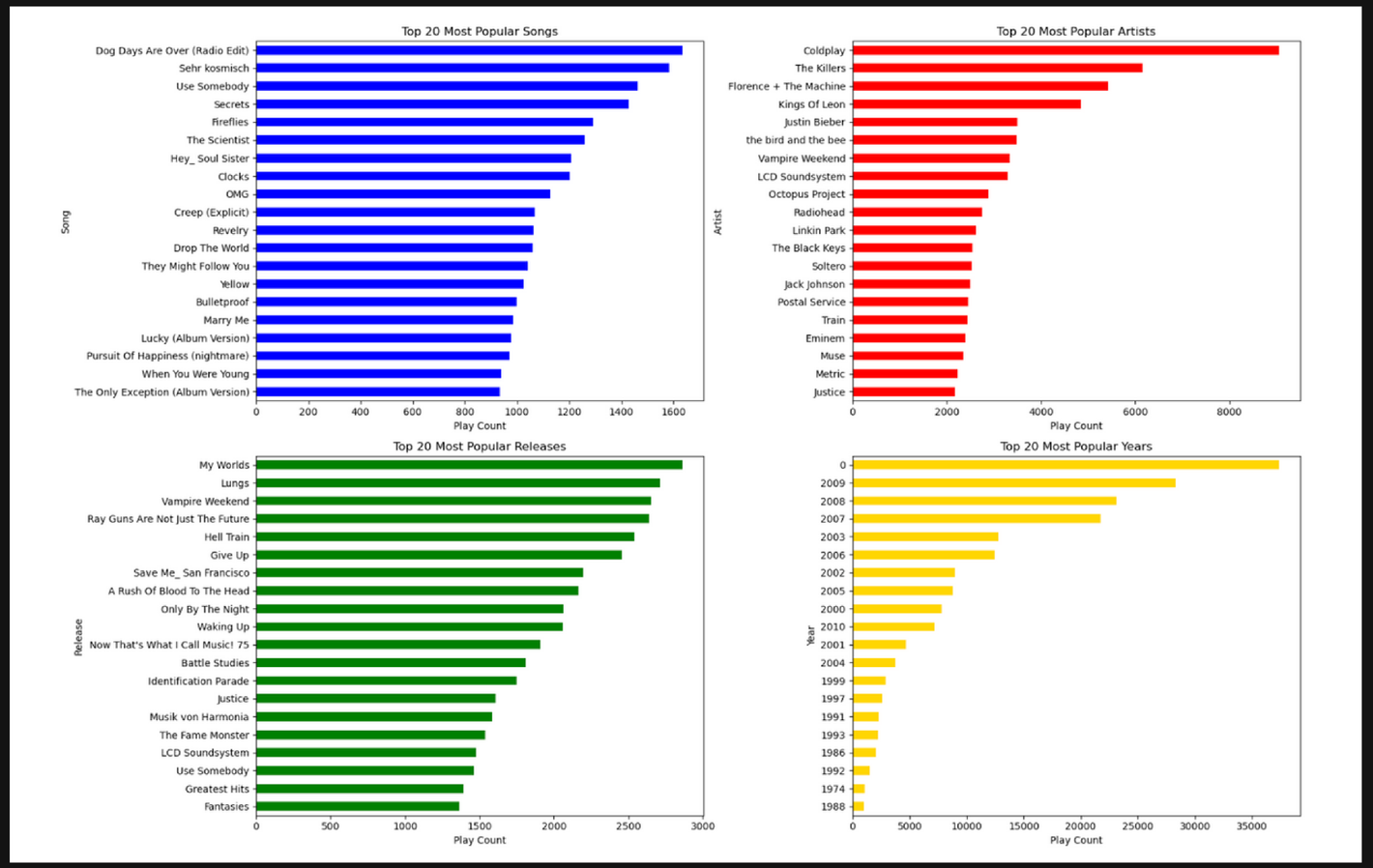
3. Exploratory Data Analysis

A plot for the number of songs in each year

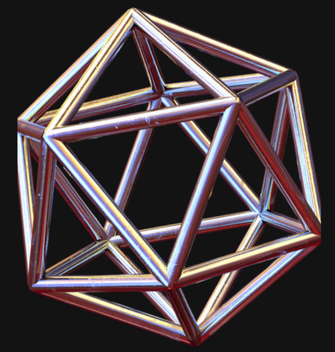


3. Exploratory Data Analysis

The top 20 in the different features

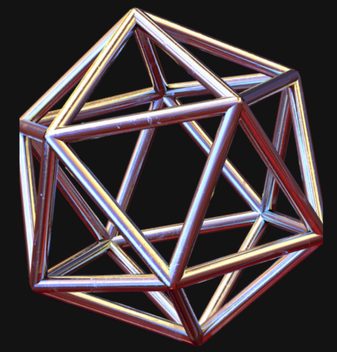


4. Recommend Alogirthms & Evaluate Metrices



4.1 Rank / Popularity - Based Recommendation System
4.2 User - User similarity - Based Collaborative Filtering
4.3 Item - Item similarity - Based Collaborative Filtering
4.4 Model Based Collaborative Filtering / Matrix factorization
4.5 Clustering - Based Recommendation System
4.6 Content Based Recommendation System

4. Recommend Alogirthms & Evaluate Metrices



Performance Metrics:

- Precision @ k : The fraction of recommended items that are relevant in top k predictions
- Recall @ k: The fraction of relevant items that are recommended to the user in top k predictions
- RMSE: Checks how far the overall predicted ratings are from the actual ratings
- F1_Score @ k: The harmonic mean of Precision @ k and Recall @ k

4.1 Popularity-Based Recommendation System

Taking the count and sum of play counts of the songs to build the popularity recommendation systems based on the sum of play counts.

song_id	avg_count	play_freq
21	1.622642	265
22	1.492424	132
52	1.729216	421
62	1.728070	114
93	1.452174	115
...	...	...
9939	1.683983	231
9942	2.486667	150
9960	1.597122	139
9981	1.921053	152
9989	1.333333	120

563 rows × 2 columns

Top 10 recommendations Results:

[7224, 8324, 6450, 9942, 5531, 5653, 8483, 2220, 657, 614]

This is helpful in the case of cold start problems (when you don't have data points say for a new user to a platform)

4.2 User - User similarity - Based Collaborative Filtering

Building the default user-user-similarity model

```
sim_options = {'name': 'pearson_baseline',
               'user_based': True}
```

5 nearest neighbors to user_id 0:
[42, 1131, 17, 186, 249]

	song_id	play_freq	predicted_ratings	corrected_ratings
3	7224	107	3.141147	3.044473
1	614	373	2.525000	2.473222
2	5653	108	2.514023	2.417798
0	352	748	2.425000	2.388436
4	6450	102	2.394927	2.295913

```
RMSE: 1.0521
Precision:  0.413
Recall:  0.721
F_1 score:  0.525
```

4.3 Item - Item similarity - Based Collaborative Filtering

Building the default item-item-similarity model

```
sim_options = {'name': 'pearson_baseline',
               'user_based': False}
```

5 most similar to item_id 0:
[124, 523, 173, 205, 65]

	song_id	play_freq	predicted_ratings	corrected_ratings
4	2342	111	2.653903	2.558987
2	5101	130	2.386577	2.298871
3	139	119	2.313727	2.222057
1	7519	168	2.270864	2.193712
0	8099	275	2.212702	2.152399

```
RMSE: 1.0328
Precision:  0.408
Recall:  0.665
F_1 score:  0.506
```

4.4 Model Based Collaborative Filtering-Matrix factorization

Building SVD model

```
svd_optimized = SVD(n_epochs = 30, lr_all = 0.01, reg_all = 0.2, random_state = 1)
```

	song_id	play_freq	predicted_ratings	corrected_ratings
2	7224	107	2.601899	2.505225
1	5653	108	2.108728	2.012502
4	8324	96	2.014091	1.912029
0	9942	150	1.940115	1.858465
3	6450	102	1.952493	1.853478

```
RMSE: 1.0141
Precision:  0.415
Recall:  0.635
F_1 score:  0.502
```

4.5 Clustering - Based Recommendation System

Building Clustering model

```
clust_tuned = CoClustering(n_cltr_u = 5,n_cltr_i = 5, n_epochs = 10, random_state = 1)
```

	song_id	play_freq	predicted_ratings	corrected_ratings
4	7224	107	3.711503	3.614829
3	5653	108	2.903883	2.807658
0	6860	169	2.691043	2.614120
1	657	151	2.606354	2.524975
2	8483	123	2.582807	2.492640

```
RMSE: 1.0654
Precision:  0.394
Recall:  0.566
F_1 score:  0.465
```

4.6 Content Based Recommendation System

Create a different column named "text" by concatenating the "title", "release", "artist_name" columns
Create a dataframe with the title column as the index

title	user_id	song_id	play_count	text
Daisy And Prudence	6958	447	1	Daisy And Prudence Distillation Erin McKeown
The Ballad of Michael Valentine	6958	512	1	The Ballad of Michael Valentine Sawdust The Ki...
I Stand Corrected (Album)	6958	549	1	I Stand Corrected (Album) Vampire Weekend Vamp...
They Might Follow You	6958	703	1	They Might Follow You Tiny Vipers Tiny Vipers
Monkey Man	6958	719	1	Monkey Man You Know I'm No Good Amy Winehouse

Creating a function to pre-process the text data:

```
# Function to tokenize the text
def tokenize(text):

    text = re.sub(r"[^a-zA-Z]"," ", text.lower())

    tokens = word_tokenize(text)

    words = [word for word in tokens if word not in stopwords.words('english')]  # Using stopwords of english

    text_lems = [WordNetLemmatizer().lemmatize(lem).strip() for lem in words]

    return text_lems
```

4.6 Content Based Recommendation System

Creating tfidf vectorizer

```
nltk.download('omw-1.4')
tfidf = TfidfVectorizer(tokenizer = tokenize)
```

Recommend for the song with title 'Learn To Fly'

```
[509, 234, 423, 345, 394, 370, 371, 372, 373, 375]

['Everlong',
 'The Pretender',
 'Nothing Better (Album)',
 'From Left To Right',
 'Lifespan Of A Fly',
 'Under The Gun',
 'I Need A Dollar',
 'Feel The Love',
 'All The Pretty Faces',
 'Bones']
```

User - User similarity Based Collaborative Filtering

```
RMSE: 1.0521
Precision:  0.413
Recall:  0.721
F_1 score:  0.525
```

Item - Item similarity Based Collaborative Filtering

```
RMSE: 1.0328
Precision:  0.408
Recall:  0.665
F_1 score:  0.506
```

Model Based Collaborative Filtering-Matrix factorization

```
RMSE: 1.0141
Precision:  0.415
Recall:  0.635
F_1 score:  0.502
```

Clustering - Based Recommendation System

```
RMSE: 1.0654
Precision:  0.394
Recall:  0.566
F_1 score:  0.465
```

5. Conclusion

In conclusion, this system effectively incorporates various recommendation methods, including collaborative filtering, content-based filtering, ranking, SVD, and clustering. The results demonstrate the system's flexibility and diversity in providing personalized suggestions for users.

5. Conclusion

Proposal for the future solution design and outlook

- A robust hybrid recommendation system will be used
- Build an interactive tool showcasing the end to end machine learning process
- Hyperparameter tuning will be done to improve model performance
- Continuous training on new data to improve model

Thank you!