

MIT - Data Science, Data Analytics & Machine Learning

Como recomendar filmes com base em classificações?

Final Assignment

Abordagem híbrida para sistema de recomendação

Renan Rocha e Thiago de Carvalho



problem_statement

É possível que um sistema de recomendação de filmes que sugira itens propensos a serem classificados (favoráveis e fora da bolha) a fim de reduzir organicamente a esparsidade dos dados a longo prazo?



print (sincere)

Desenvolver um sistema colaborativo de recomendação de filmes, baseado em classificações que sugiram para os usuários 4 segmentos de filmes:

Tipos de filmes vistos
frequentemente



TRUE LOVE
(zona de conforto)



AFFAIR
(quando o amor vacila)



ONE NIGHT STAND
(por que não tentar a sorte?)



DRUNK FLERT
(cara, eu não me reconheço!)

Tipos de filmes vistos
raramente

```
print(scope, len(movieLens_dataset_ml-25m))
```

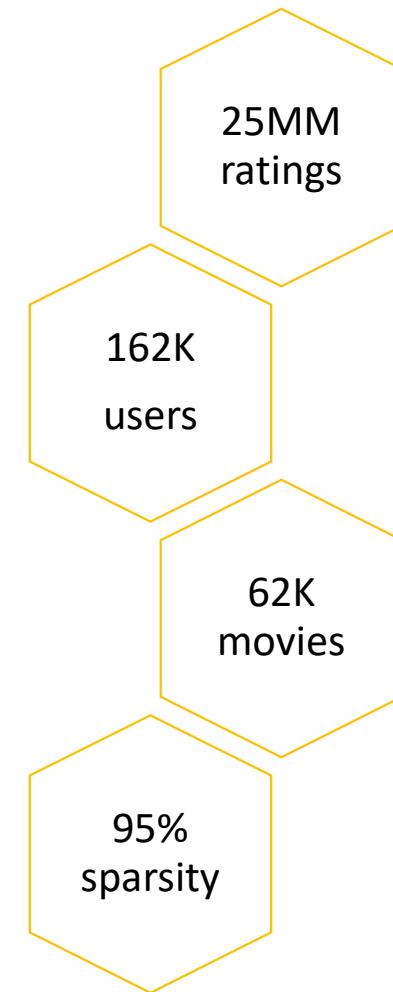
ESCOPO

Desenvolver um sistema colaborativo de recomendação de filmes filtrados com base em classificações.

- abordagem híbrida, aplicando pelo menos 3 algoritmos,
- para tratar de questões de esparsidade (sparsity) de dados

FORA DO ESCOPO

- Outros problemas clássicos (**Scalability, Cold Start**)
- Conta **compartilhada**: Múltiplos usuários/utilização de telas
- Heavy users/ bots



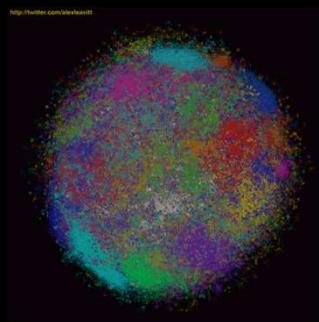
concepts.restore(memory_cards)

Sparsity index

Índice que varia de 1 a 0, quanto mais alto o índice, mais o esparso (com poucas avaliações / total de possíveis avaliações) o item ou usuário é..

$$sparsity = 1 - \frac{count_nonzero(A)}{total_elements_of_A}$$

Fuzzy clustering



Uma forma de agrupamento em que cada ponto de dados pode pertencer a mais de um grupo.

NPS – Net promoter score

Métrica que mostra a probabilidade de os usuários recomendarem uma empresa, produto ou serviço a um amigo ou colega - desenvolvida por Fred Reichheld.



SVD

Algoritmo de fatoração matricial, popularizado por Simon Funk durante o Prêmio Netflix, que é equivalente à fatoração Matricial Probabilística quando as linhas de base não são utilizadas.

Para o Sincere, o hiperparametro de fatores latentes foi definido para 75 vizinhos. Isso significa, extrair características e correlação da matriz de 75 itens de usuário mais próximos.

Benchmarks and references

- Najafabadi et Al. - An Effective Collaborative User Model Using Hybrid Clustering Recommendation Methods
- Mohammed Fadhel Aljunid, Manjaiah DH - An Efficient Deep Learning Approach for Collaborative Filtering Recommender System
- Nicholas Becker - <https://beckernick.github.io/>

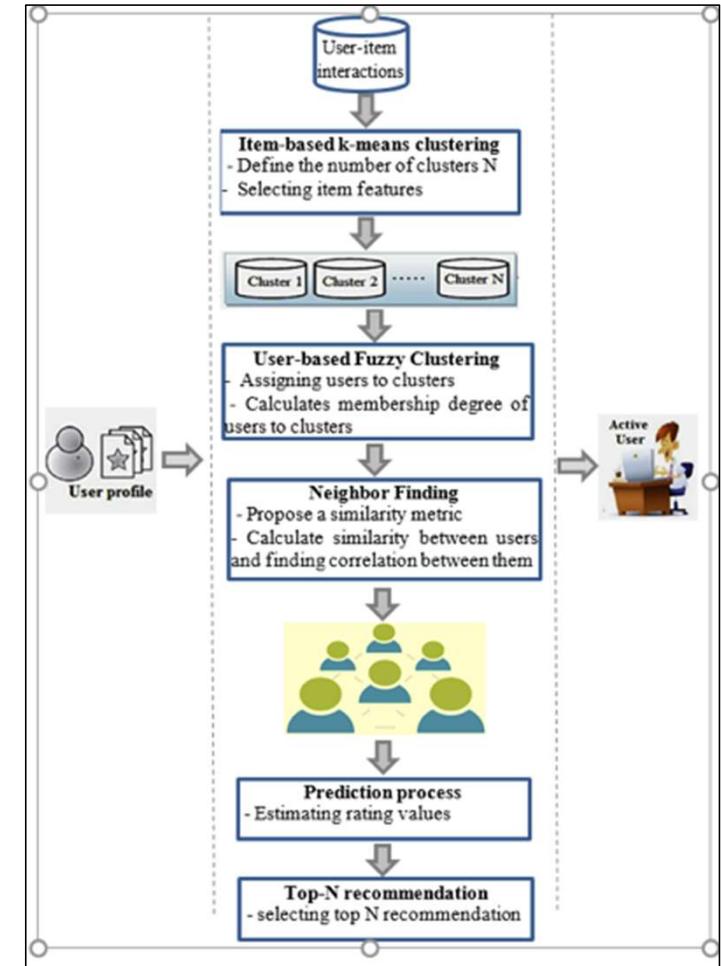
pip update recommender_systems (a)

Aplicação sequencial de dois métodos distintos de agrupamento:

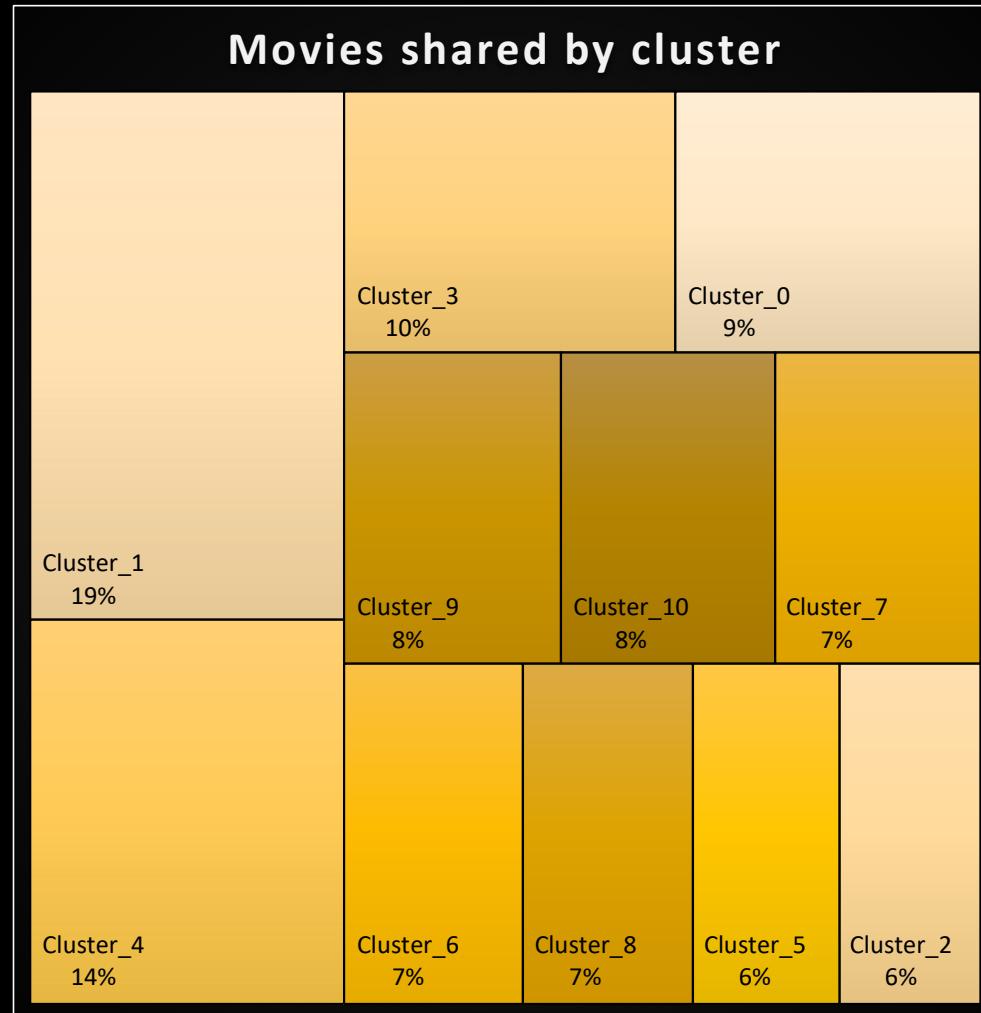
1º (hard clustering) os itens são agrupados por k means para reduzir a disparidade

2º (soft clustering) agrupamento de usuários usando fuzzy c means.

[!] O output do k means é usado como input do fuzzy, quantidade de filmes de cada cluster vistos por usuário atributo diferencia-los quanto ao uso.



```
print(movie_clustering_results)
```



	Movie_cluster_0	Movie_cluster_1	Movie_cluster_2	Movie_cluster_3
movie age	-	-	old	youth
popularity ML	very rated	few ratings	very rated	few ratings
popularity IMDB	-	awarded	-	higher rates
NPS ML	-	higher satisfaction	-	high satisfaction
Genres	Crime Film-Noir Mystery Thriller	Biography Sport War Drama	Animation Family Fantasy Musical Short	Biography Documentary Music News Sport
	Movie_cluster_4	Movie_cluster_5	Movie_cluster_6	Movie_cluster_7
movie age	old	youth	-	-
popularity ML	-	-	-	-
popularity IMDB	-	-	-	lower rating
NPS ML	-	-	low satisfaction	low satisfaction
Genres	Comedy Musical Reality-TV Talk-Show Western	Mystery Thriller	Comedy	Adult Horror Sci-Fi
	Movie_cluster_8	Movie_cluster_9	Movie_cluster_10	
movie age	-	old	-	
popularity ML	lots of ratings	-	-	
popularity IMDB	-	awarded	-	
NPS ML	lower satisfaction	-	low satisfaction	
Genres	Action Adventure Sci-Fi	Romance	Romance Musical	

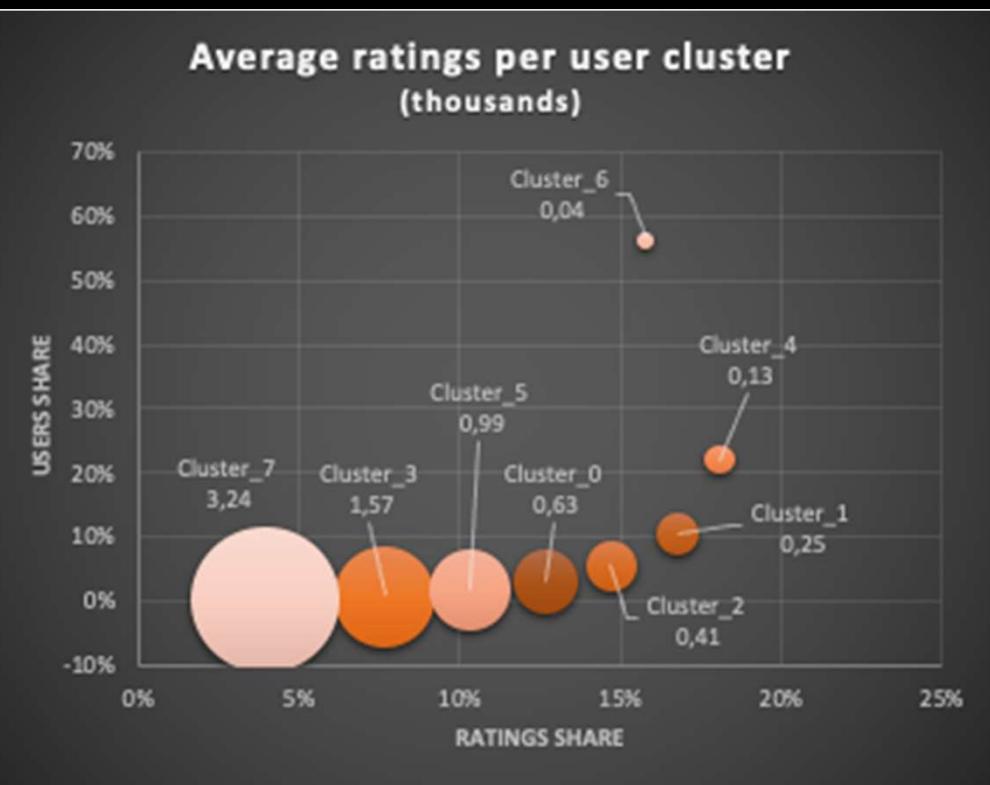
Clustering metrics used to define k=11: Elbow, Silhouette, David Boudin and Dendrogram

Fuzzy C-means

	movie_cluster_0.0	movie_cluster_1.0	movie_cluster_2.0	movie_cluster_3.0	movie_cluster_4.0	movie_cluster_5.0	movie_cluster_6.0	movie_cluster_7.0	movie_cluster_8.0	movie_cluster_9.0	movie_cluster_10.0
0	6.0	21.0	2.0	1.0	7.0	3.0	8.0	0.0	2.0	13.0	7.0
1	16.0	47.0	21.0	0.0	16.0	9.0	8.0	1.0	43.0	9.0	14.0
2	92.0	63.0	49.0	1.0	62.0	61.0	22.0	22.0	237.0	16.0	31.0
3	18.0	15.0	31.0	5.0	35.0	17.0	1.0	4.0	110.0	1.0	5.0
4	14.0	18.0	10.0	0.0	16.0	4.0	11.0	1.0	12.0	6.0	9.0
...
162536	6.0	11.0	8.0	0.0	17.0	4.0	10.0	2.0	9.0	10.0	24.0
162537	6.0	14.0	14.0	0.0	21.0	7.0	11.0	2.0	14.0	30.0	35.0
162538	3.0	10.0	2.0	0.0	3.0	4.0	4.0	0.0	12.0	7.0	2.0
162539	5.0	10.0	20.0	0.0	2.0	7.0	3.0	6.0	17.0	8.0	10.0
162540	15.0	26.0	17.0	1.0	31.0	6.0	14.0	4.0	40.0	14.0	14.0

Fuzzy Matrix		User Fuzzy C Means Clustering							
		0	1	2	3	4	5	6	7
User_cluster_Label	0	45%	9%	22%	2%	5%	12%	4%	0%
	1	2%	51%	14%	0%	22%	0%	9%	0%
	2	14%	22%	47%	0%	9%	2%	5%	0%
	3	9%	4%	6%	49%	3%	21%	3%	5%
	4	0%	13%	2%	0%	61%	0%	23%	0%
	5	20%	6%	9%	12%	4%	46%	3%	0%
	6	0%	1%	0%	0%	10%	0%	89%	0%
	7	6%	5%	5%	17%	4%	9%	4%	50%

```
print(user_clustering_results)
```

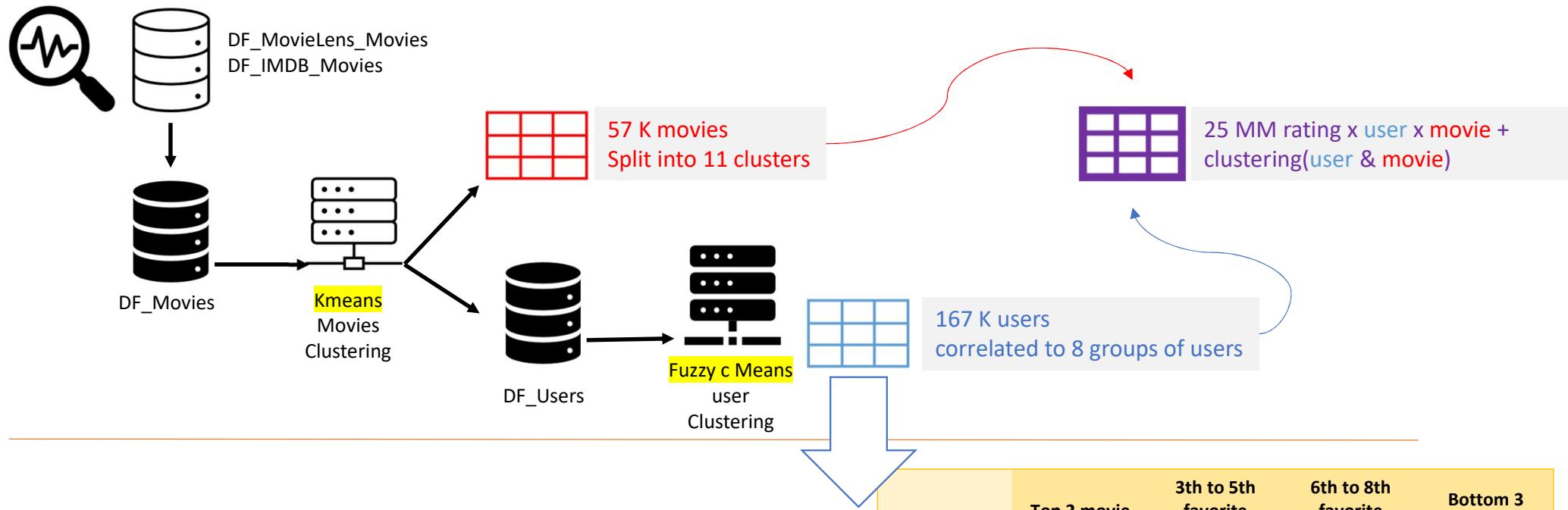


Movie Clusters	User_cluster_0	User_cluster_1	User_cluster_2	User_cluster_3
Favorite day	Monday	Saturday	Saturday	Tuesday
Favorite time	Wednesday	Monday	Monday	-
Voter profile	afternoon	morning	morning	evening
True love movie clusters	8, 0	8, 4	0, 1	8, 1
Affair movie clusters	2, 1, 9	1, 0, 10	8, 4, 10	0, 5, 7
1 night stand movie clusters	4, 10, 5	2, 7, 6	5, 6, 2	2, 4, 9
Drunk flirt movie clusters	6, 7, 3	5, 9, 3	9, 7, 3	10, 6, 3

Movie Clusters	User_cluster_4	User_cluster_5	User_cluster_6	User_cluster_7
Favorite day	not Monday	Wednesday	Thursday	Saturday
Favorite time	afternoon	morning	night	evening
Voter profile	promoter	detractor	promoter	neutral
True love movie clusters	0, 8	8, 1	8, 4	0, 1
Affair movie clusters	1, 10, 2	10, 2, 4	0, 1, 2	8, 4, 10
1 night stand movie clusters	4, 5, 9	9, 0, 5	10, 6, 5	5, 6, 9
Drunk flirt movie clusters	6, 7, 3	6, 7, 3	7, 3, 9	3, 2, 7

plt.sincere_setup.show()

Setup



Output

User Cluster	Top 2 movie clusters	3th to 5th favorite movie clusters	6th to 8th favorite movie clusters	Bottom 3 movie clusters
1	1, 2	4,5,6	7,8,9	3,10,11
2	2,5	1,3,9	4,8,10	6, 7,11
...
n	6,9	7,8,10	2,3,11	1,4,5

pip update recommender_systems (b)

Um sistema híbrido de recomendação que é estabelecido com base em de fatoração de matrizes (...) que trabalha sobre os feedbacks implícitos dos usuários e também informações auxiliares tanto dos usuários quanto dos itens.

Available online at www.sciencedirect.com

ScienceDirect

Procedia
Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)
An Efficient Deep Learning Approach for Collaborative Filtering
Recommender System

Mohammed Fadhel Aljunid^a, Manjaiah DH^b

^aDepartment of Computer Science, Mangalore University, India
^bDepartment of Computer Science, Mangalore University, India

Abstract

Owing to the enormous growth in information over the past few decades, the world has become a global village. The recommendation system remains the most widely used type of commercial websites. The personalized recommender system is of paramount importance in modeling user's preference on items based on their past interactions (e.g., ratings and clicks), known as collaborative filtering (CF) technique. Although CF is very important among the algorithms used in recommendation systems, it suffers some setbacks such as the sparsity of matrix ratings, scalability, and integrals nature of data. Several research studies have shown that the above-mentioned obstacle could be tackled with the help of matrix factorization (MF) techniques. In spite of the fact that the technique is likely to suffer from lack of some meaningful signals by using a low ranked approximation as well as lack of sparsity in times of denser singular vectors. Recently, deep learning techniques have proven to learn good representation in natural language processing, image classification, and so on. In this work, we propose a deep learning method of collaborative recommender systems (DLCRS). We have made a comparative study of the proposed method and existing methods. Experimental results demonstrate that our approach gives improved results compared to already existing methods. We empirically evaluate DLCRS on two famous datasets: 100K and IM MovieLens.

© 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nd/4.0/>)
Peer-review under responsibility of the scientific committee of the Third International Conference on Computing and Network Communications (CoCoNet'19).

Keywords: Recommender System , Collaborative Filtering , Matrix Factorization , Deep Learning , MovieLens Datasets

1. Introduction

The advancement of artificial intelligence and machine learning technologies has brought intelligent products that are essential in providing access to various endeavors of people's day-to-day life. Effective and useful information

* Corresponding author. Tel.: +917304385644.
E-mail address: ngn505@yahoo.com

© 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nd/4.0/>)
Peer-review under responsibility of the scientific committee of the Third International Conference on Computing and Network Communications (CoCoNet'19).
10.1016/j.procs.2020.041090

```
plt.sincere_schema.show()
```

32 MATRIZES FATORADAS VIA SVD

8 CLUSTERS DE USUÁRIOS

*

4 SEGMENTOS

TRUE LOVE

AFFAIR

1 NIGHT STAND

DRUNK FLERT

User Cluster	Top 2 movie clusters	3th to 5th favorite movie clusters	6th to 8th favorite movie clusters	Bottom 3 movie clusters
2	2,5	1,3,9	4,8,10	6,7,11

Filter 25 MM ratings database by user cluster and movie cluster



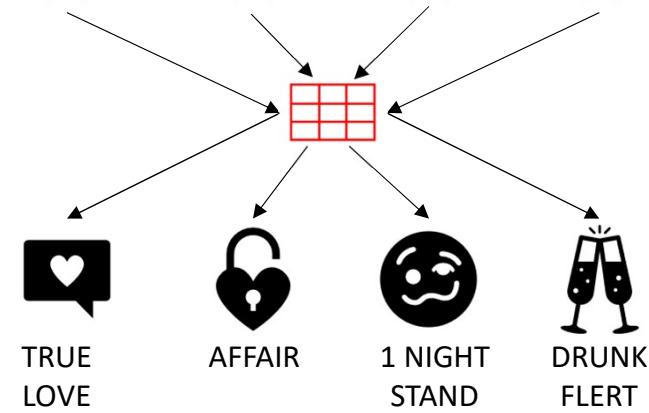
Factorization Matrix via (SVD)



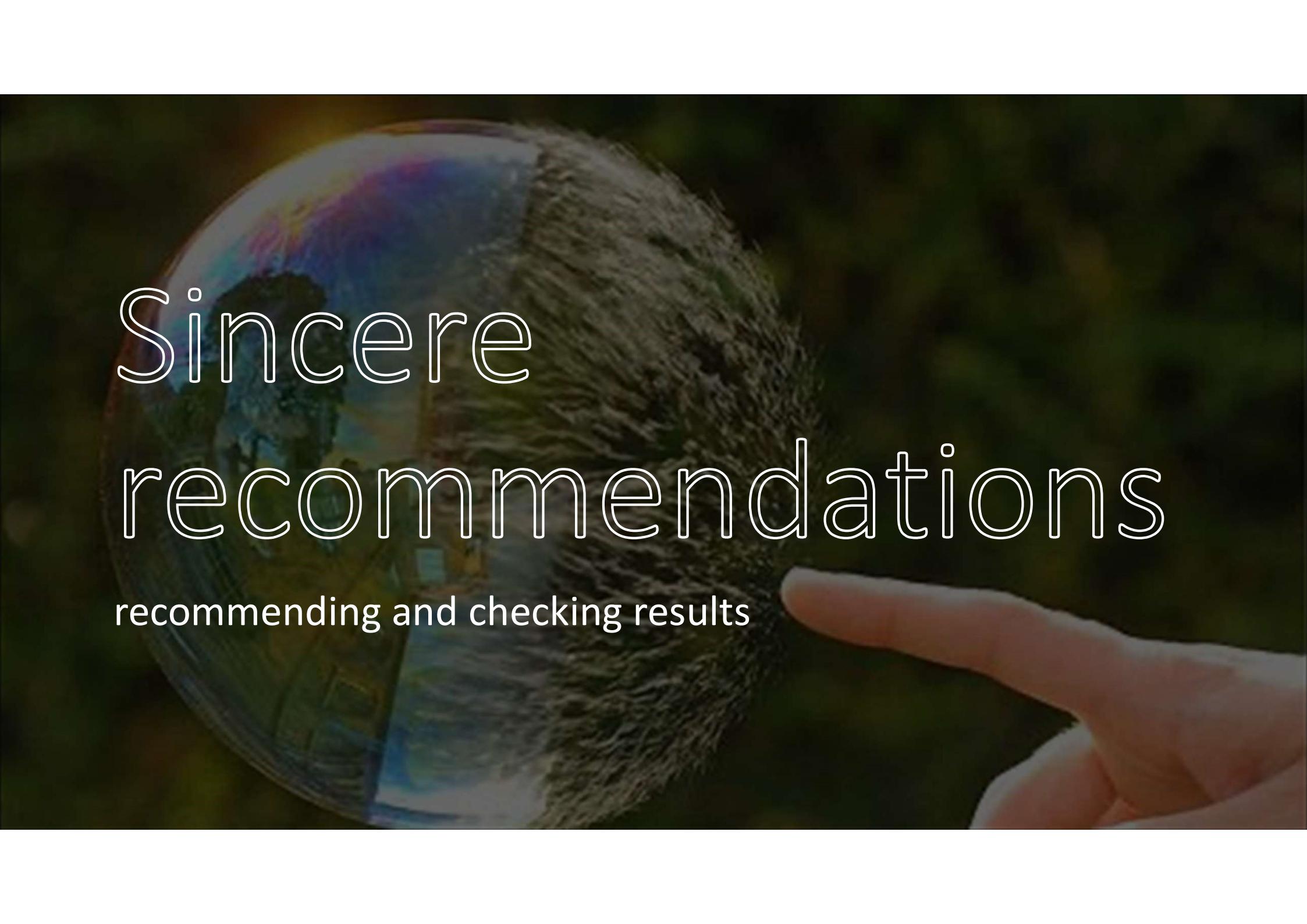
Making Predictions from the Decomposed Matrices



Making Movie Recommendations



```
print(sincere)
```



Sincere
recommendations

recommending and checking results

tests.describe()

Sampling



Recommending



Checking results



Target



1

Amostra de 80 usuários. 10 de cada cluster com índice fuzzy alto.

Recomendação de 20 filmes (5 para cada segmento de afinidade) usando Sincere.

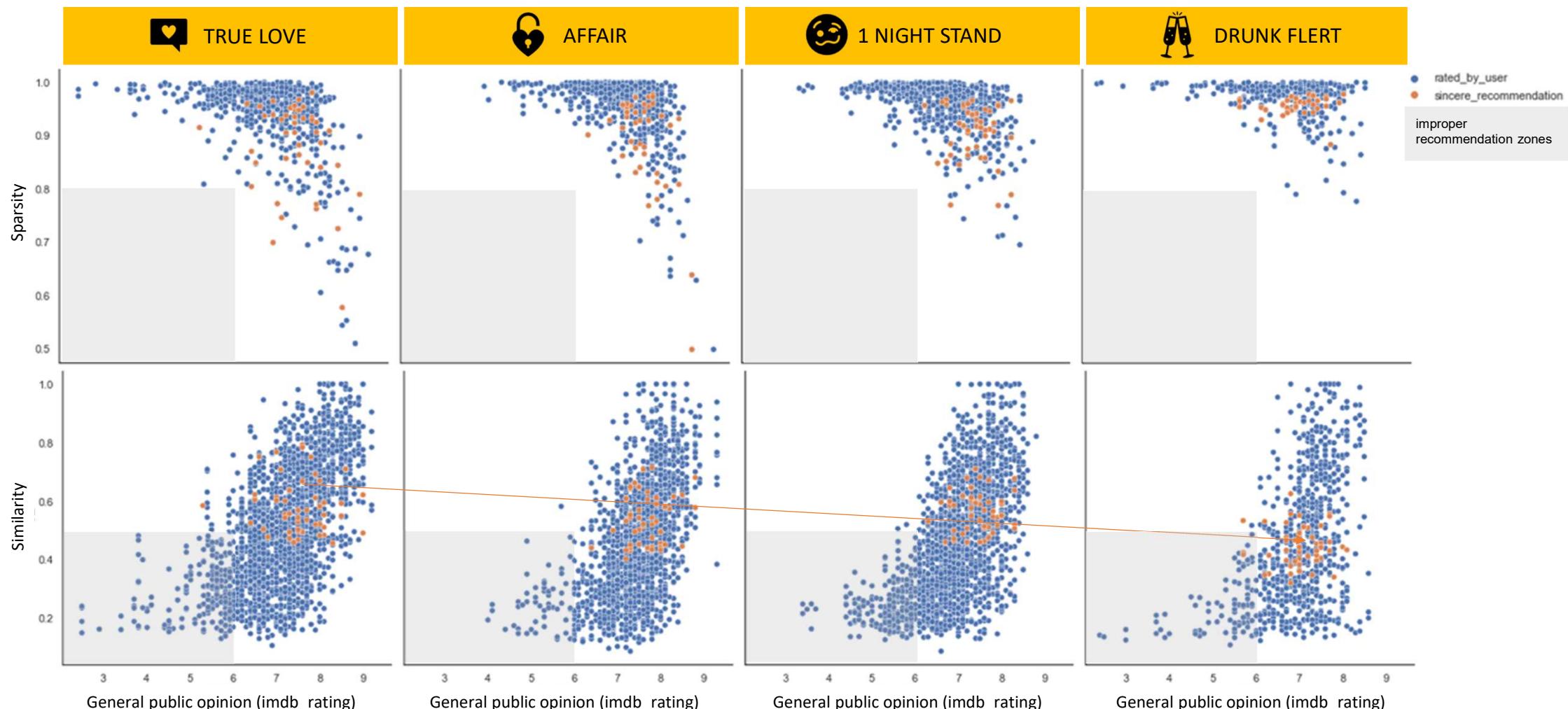
Analizar a similaridade e esparsidade das recomendações

Sincere recomendou filmes com avaliações esparsas?

Os filmes recomendados foram relevantes para o público em geral?

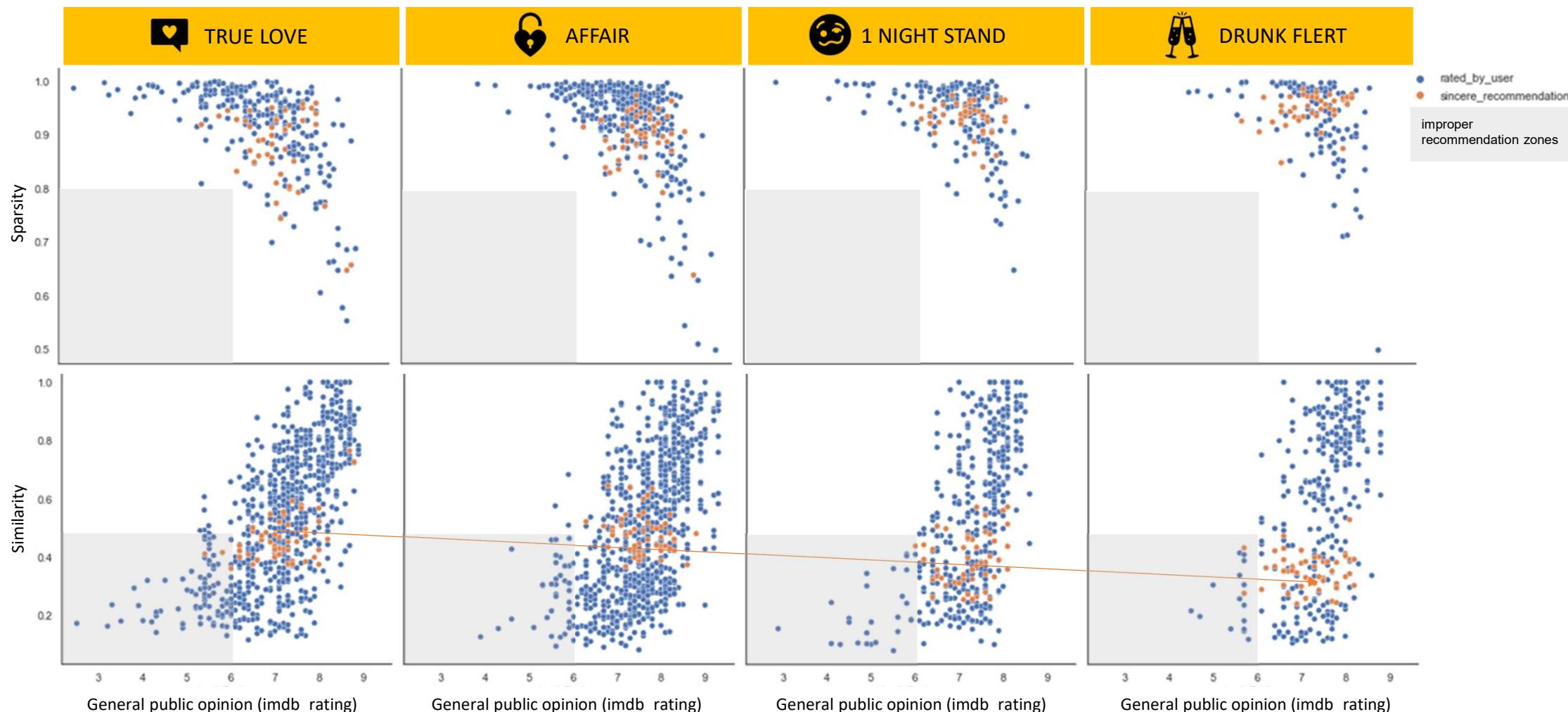
plt.sincere_results(Cluster_0)

Para os grupos de usuários mais densos (0, 1, 2, 3, 5 e 7), Sincere recomendou filmes com alta esparsidade e com classificação geral (IMDB) superior a 6. A semelhança relativa com o interesse comum do usuário está diminuindo de acordo com o segmento de recomendação, como esperado.



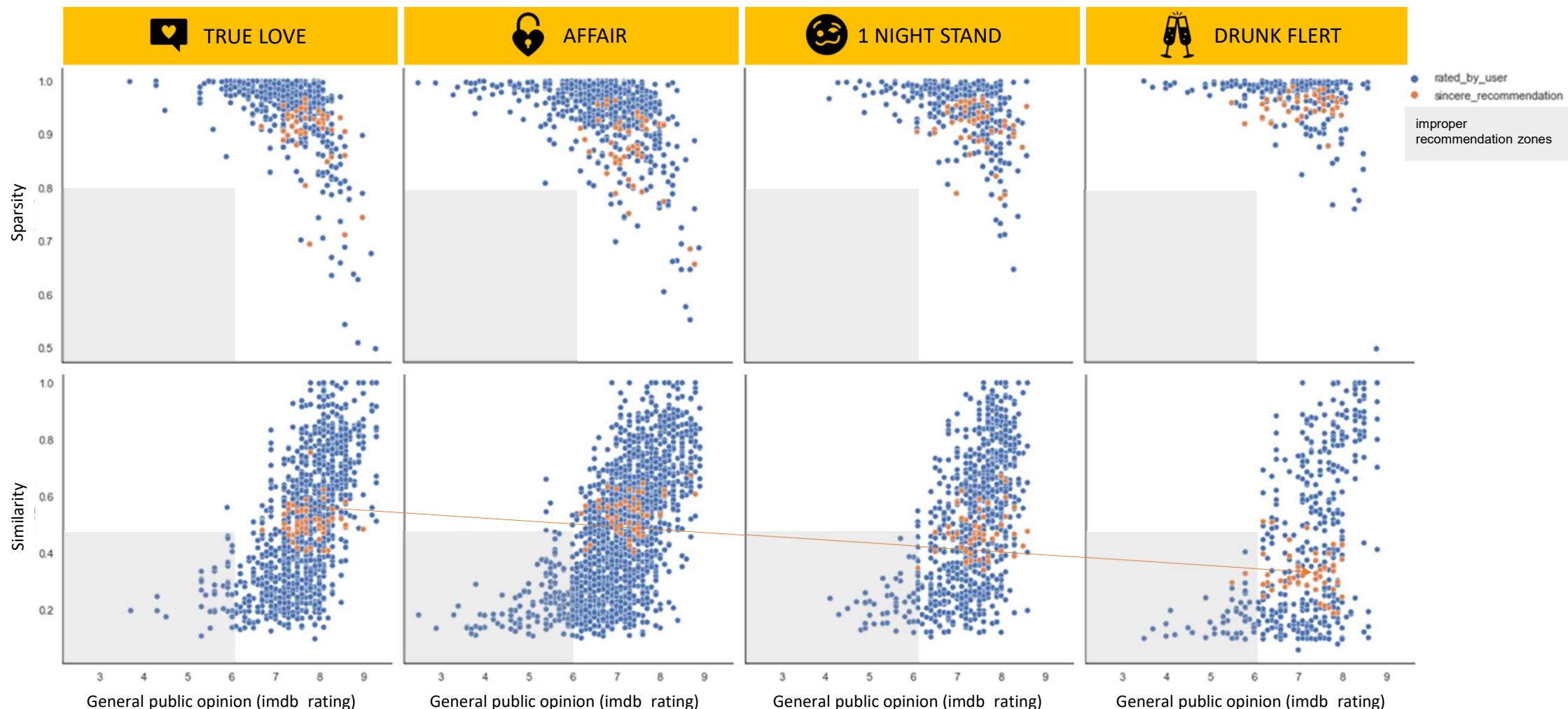
plt.sincere_results(Cluster_1)

Para os grupos de usuários mais densos (0, 1, 2, 3, 5 e 7), Sincere recomendou filmes com alta esparsidade e com classificação geral (IMDB) superior a 6. A semelhança relativa com o interesse comum do usuário está diminuindo de acordo com o segmento de recomendação, como esperado.



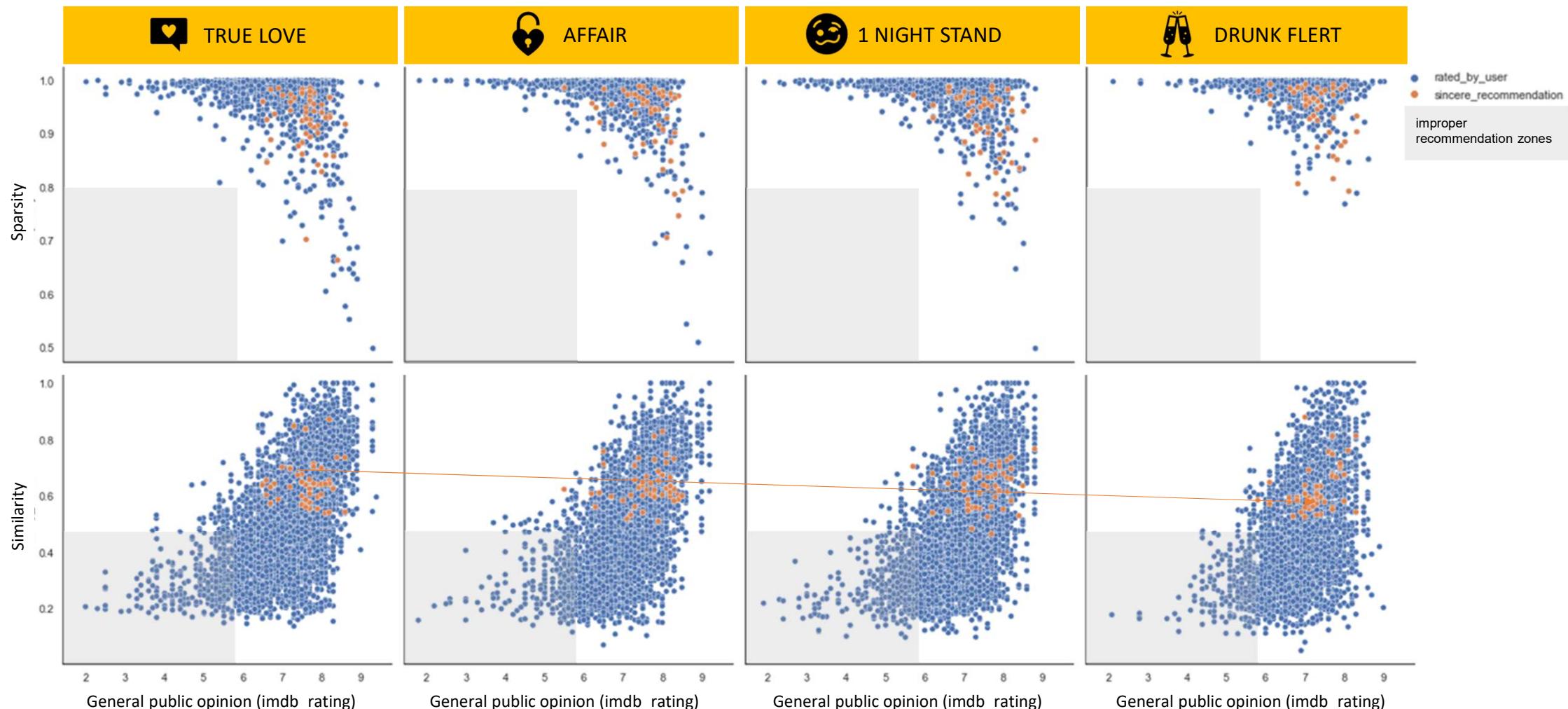
plt.sincere_results(Cluster_2)

Para os grupos de usuários mais densos (0, 1, 2, 3, 5 e 7), Sincere recomendou filmes com alta esparsidade e com classificação geral (IMDB) superior a 6. A semelhança relativa com o interesse comum do usuário está diminuindo de acordo com o segmento de recomendação, como esperado.



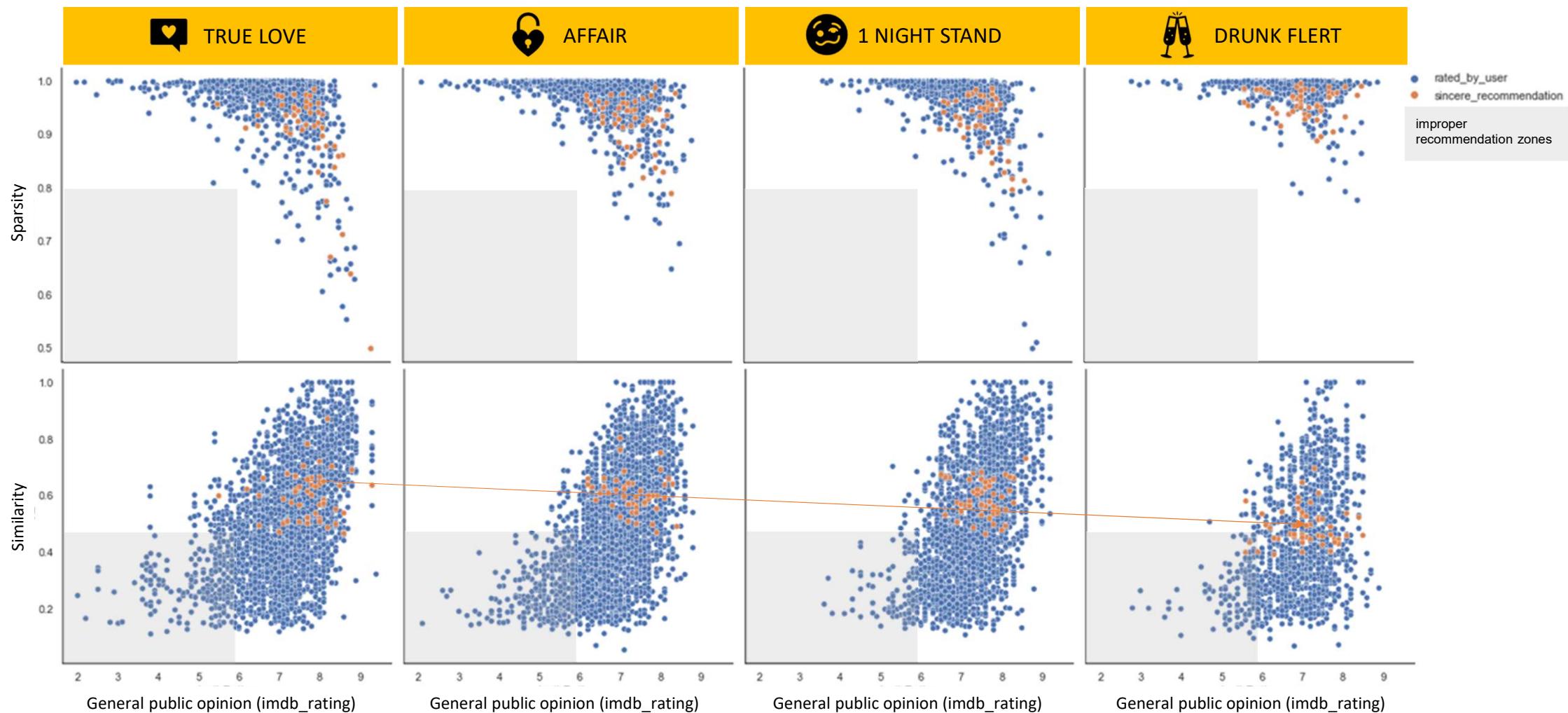
plt.sincere_results(Cluster_3)

Para os grupos de usuários mais densos (0, 1, 2, 3, 5 e 7), Sincere recomendou filmes com alta esparsidade e com classificação geral (IMDB) superior a 6. A semelhança relativa com o interesse comum do usuário está diminuindo de acordo com o segmento de recomendação, como esperado.



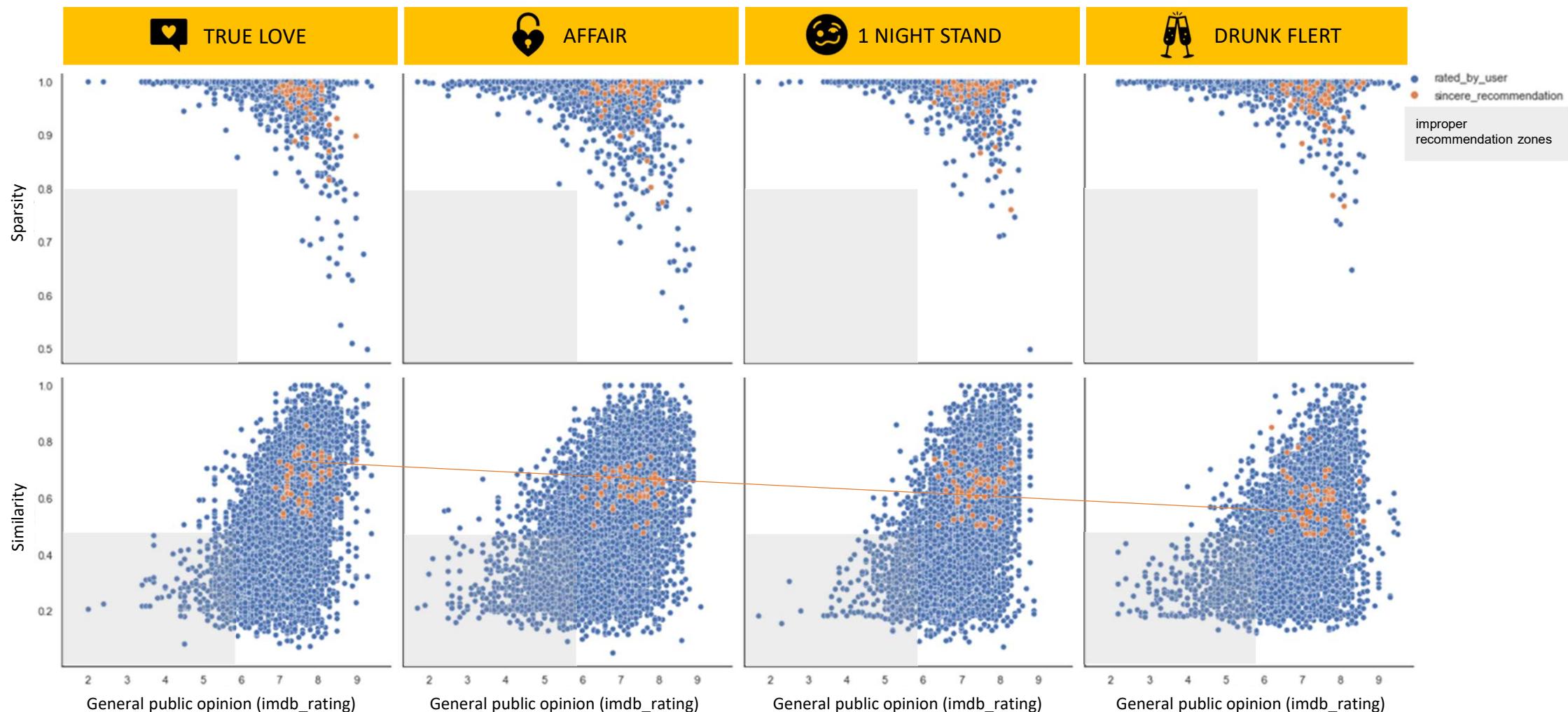
plt.sincere_results(Cluster_5)

Para os grupos de usuários mais densos (0, 1, 2, 3, 5 e 7), Sincere recomendou filmes com alta esparsidade e com classificação geral (IMDB) superior a 6. A semelhança relativa com o interesse comum do usuário está diminuindo de acordo com o segmento de recomendação, como esperado.



plt.sincere_results(Cluster_7)

Para os grupos de usuários mais densos (0, 1, 2, 3, 5 e 7), Sincere recomendou filmes com alta esparsidade e com classificação geral (IMDB) superior a 6. A semelhança relativa com o interesse comum do usuário está diminuindo de acordo com o segmento de recomendação, como esperado.

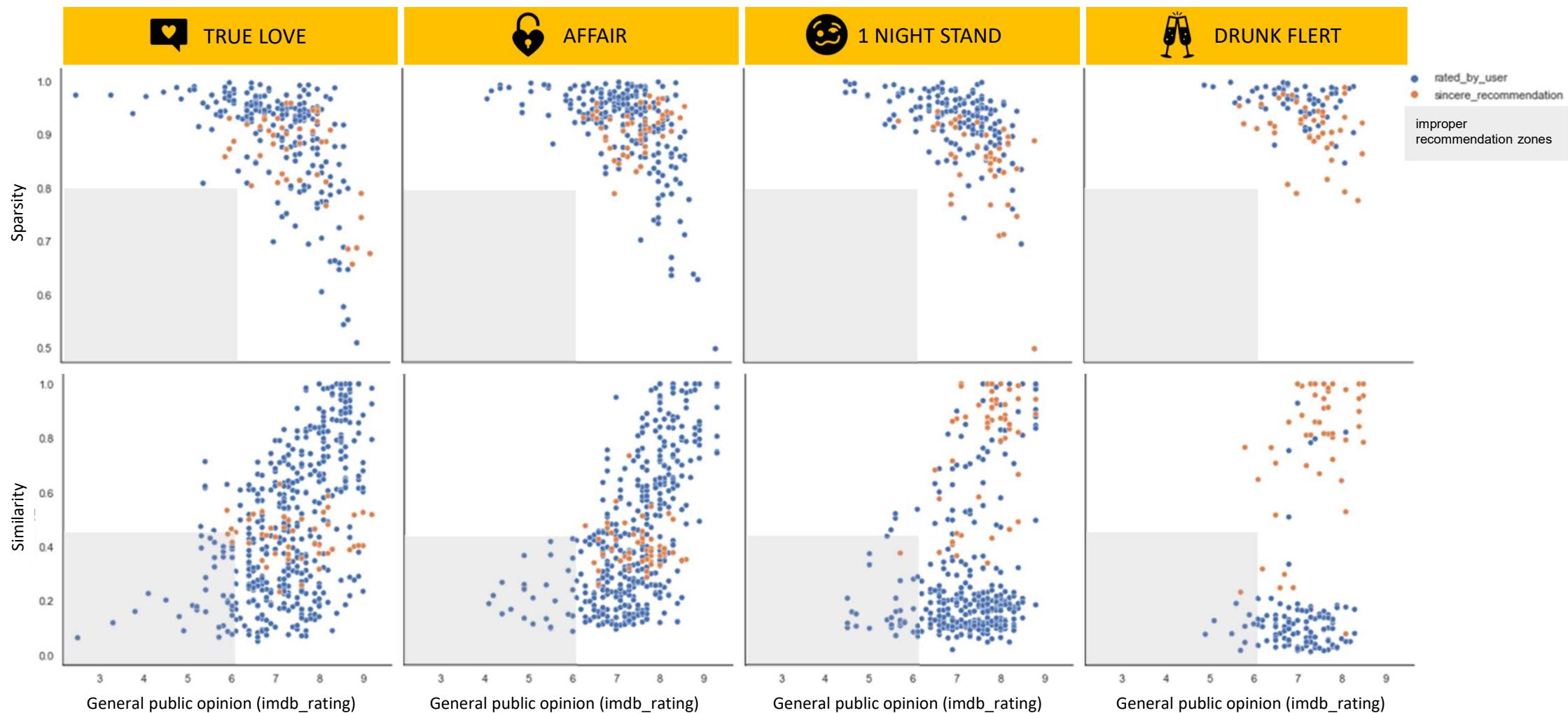


Example. ({userId:130311, cluster:0})

	TRUE LOVE	AFFAIR	1 NIGHT STAND	DRUNK FLERT
Rated by user				
Imdb	9.0	9.3	8.6	8.4
Sparsity	0.99	0.99	0.99	0.99
Sincere				
Imdb	8.1	7.3	8.3	7.7
Sparsity	0.96	0.94	0.91	0.97
Similarity	0.61	0.43	0.56	0.55

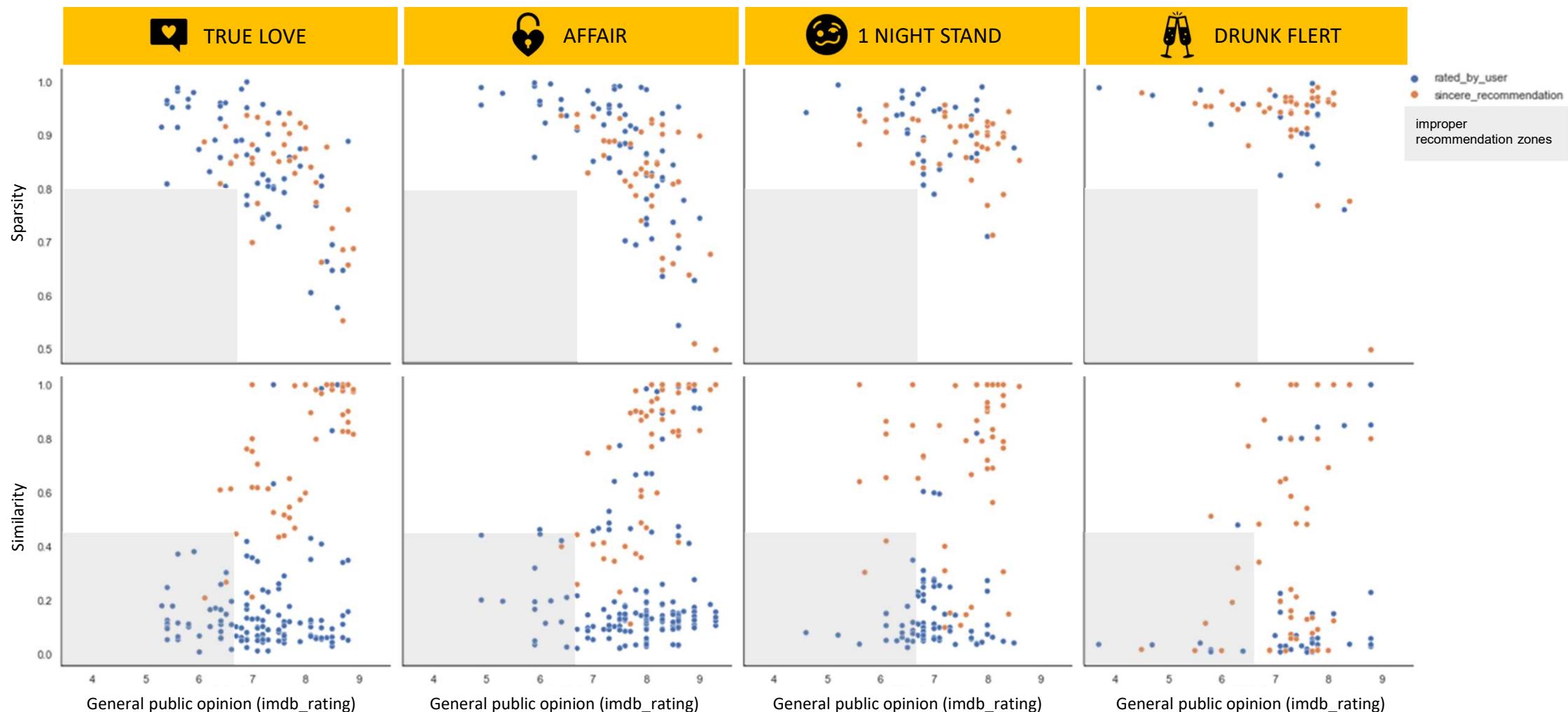
plt.sincere_results(Cluster_4)

Com o aumento da esparsidade (clusters 4 e 6), nosso modelo tentou recomendar adequadamente filmes seguindo nossas diretrizes de remoção da esparsidade e melhor classificação geral (IMDB), mas encontrou dificuldades na definição dos clusters de usuário em relação aos itens.



plt.sincere_results(Cluster_6)

Com o aumento da esparsidade (clusters 4 e 6), nosso modelo tentou recomendar adequadamente filmes seguindo nossas diretrizes de remoção da esparsidade e melhor classificação geral (IMDB), mas encontrou dificuldades na definição dos clusters de usuário em relação aos itens.



Example. ({userId:135548, cluster:6})

	TRUE LOVE	AFFAIR	1 NIGHT STAND	DRUNK FLERT
Rated by user				
Imdb	8.7	8.8	7.8	8.7
Sparsity	0.96	0.96	0.95	0.95
Similarity	0.16	0.66	0.58	0.84
Sincere				
Imdb	8.1	8.0	8.3	7.7
Sparsity	0.96	0.97	0.91	0.97
Similarity	1	1	1	1

tests.describe()

Sampling



Recommending



Checking results



Target



1

Amostra de 80 usuários. 10 de cada cluster com índice fuzzy alto.

2

Dividir o banco de dados para identificar os 10% de usuários com índice fuzzy mais alto. (Com os 80 da amostra usada no teste 1 incluídos).

Recomendação de 20 filmes (5 para cada segmento de afinidade) usando Sincere.

Analizar a similaridade e esparsidade das recomendações

Recomendação de 20 filmes para os mesmos 80 usuários com o sistema de recomendação tradicional.

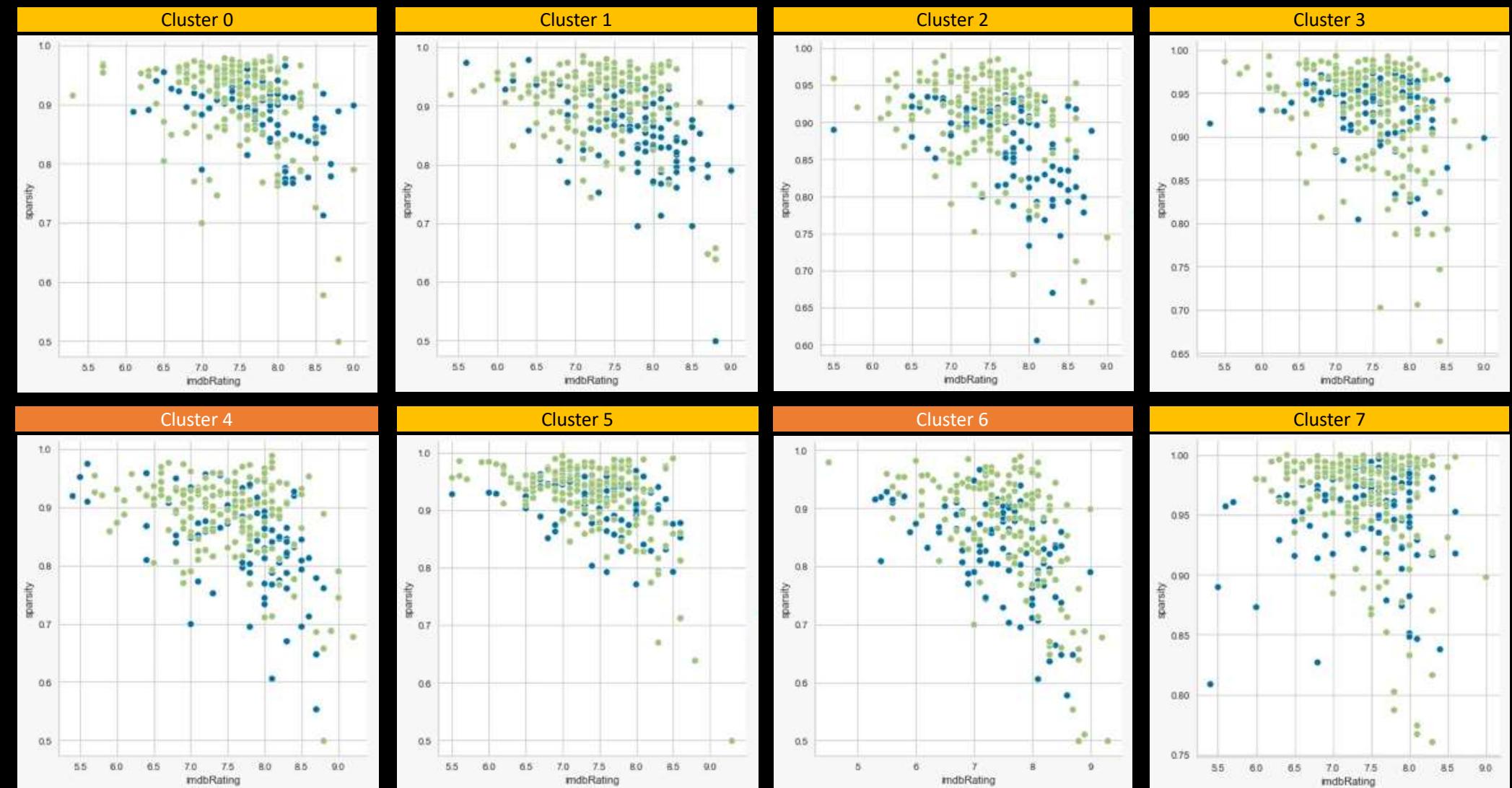
Verificar se os resultados sinceros em comparação com o sistema de recomendação comum.

Sincere recomendou filmes com avaliações esparsas?

Os filmes recomendados foram relevantes para o público em geral?

sparsity_vs_rating.compare()

- ordinary recommendations
- sincere recommendations



```
print(sincere_key_findings)
```



```
print(f'conclusion: {conclusion}')
```

O experimento SINCERE, com os seus resultados preliminares sugere que:

- É possível implantar mecanismos de recomendação que reduzem esparsidade sem afetar a satisfação do usuário.
- Não há bala de prata para recomendar corretamente para todos os usuários, é conveniente misturar técnicas.
- Os sistemas de recomendação também podem ser uma arma para estourar a bolha da segregação cultural nas redes.
- A divisão do conjunto de dados por 4 segmentos de filmes similares também superou algumas limitações computacionais.



- Enriquecer o modelo com dados dos usuários
- Revisar a clusterização de usuários especialmente para clusters 6 e 4, a fim de maximizando os benefícios do fuzzy.
- Testar outros métodos de fatoração: PMF, SVD++, NMF
- Aplicar outros modelos tradicionais de recomendação e comparar com os resultados do Sincere.
- Colocar o modelo em produção, aceitando input de novos usuários e itens

Evoluções do trabalho

- Análise do perfil de votação pode gerar mais engajamento e satisfação ao notificar usuários.
- Campanha promocional para voters pode estimular bons resultados contra esparsidade [exemplo: convites para premières exclusivas]

Business insights



Renan Rocha
renanbdr@hotmail.com
Github: renanbdr



Linkedin

Thiago de Carvalho
thiago_de_carvalho@outlook.com.br
Github: ThiagoCarvalho-81



Linkedin