PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Marcela Coury Pinto

SISTEMA DE RECOMENDAÇÃO DE FILMES

Belo Horizonte
Agosto de 2023

Marcela Coury Pinto

SISTEMA DE RECOMENDAÇÃO DE FILMES

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Belo Horizonte
Agosto de 2023

SUMÁRIO

1. Introdução

Nas últimas décadas foi notável a crescente oportunidade que a web proporcionou para a personalização de serviços, pois é possível coletar dados de uma maneira mais fácil e menos agressiva para o usuário. Devido a isso o tópico de sistemas de recomendação veio ganhando importância nas transações de negócio e e-commerce. (AGGARWAL,2004)

Um sistema de recomendação é um tipo de algoritmo que analisa dados de preferências, comportamentos de usuários e de histórico de visualizações para sugerir produtos, serviços ou informações que podem ser interessantes para o usuário. Esse tipo de sistema pode ser utilizado em diversos contextos diferentes como e-commerce, plataforma de streaming de vídeo e música, redes sociais, entre outros.

Os sistemas de recomendações de filmes, por exemplo, geralmente funcionam através da coleta de informações do usuário, como idade, sexo, preferências, gênero e histórico de visualizações. Com base nesses dados, eles constroem um modelo capaz de sugerir vários filmes que sejam relevantes para o usuário. Existem no mercado diversas plataformas que utilizam esse tipo de sistema, como Netflix, Amazon Prime Vídeo, Disney+, Star+ e HBO Max.

De acordo com uma pesquisa analisada no artigo publicado no Convibra (SILVA et al., 2020), o uso de plataformas de streaming cresceu durante a pandemia e 73% dos usuários de internet disseram que aumentaram esse tipo de consumo. Isso indica também uma oportunidade de aumentar a demanda por sistemas mais personalizados e atrativos aos usuários.

2. Descrição do Problema e da Solução Proposta

Com a grande variedade de filmes disponíveis nas plataformas de streaming, pode ser difícil para os usuários encontrar novos títulos que sejam do seu interesse, e um sistema que sugere os mais relevantes, facilita a descoberta de novos filmes e engajam os usuários na plataforma.

Outra vantagem é que quando os usuários encontram filmes que lhes interessam, eles se sentem mais satisfeitos com a plataforma. Isso pode fazer com que utilizem mais

a plataforma e indiquem para amigos e familiares, gerando mais usuários para a empresa.

No contexto da crescente demanda por sistemas de recomendação para melhor atender ao usuário, este trabalho tem como objetivo desenvolver um sistema de recomendação de filmes, utilizando três tipos de recomendação personalizada: recomendação baseada em conhecimento, recomendação baseada em conteúdo e recomendação baseada em filtragem colaborativa. E também dois sistemas não personalizados: mais assistidos e mais bem avaliados.

3. Coleta de Dados

Para o desenvolvimento do projeto foram utilizados os conjuntos de dados *(datasets)* de filmes da *MovieLens* do *GroupLens Research Lab* da Universidade do Minnesota. Esse grupo utiliza tecnologia de filtragem colaborativa para a recomendação de filmes e disponibiliza no site do grupo vários *datasets*.

Os conjuntos de dados utilizados contêm 9742 filmes, 100836 avaliações (*ratings*) que foram criadas por 610 usuários em um período entre 29 de março de 1996 e 24 de setembro de 2018.

Duas planilhas foram utilizadas para esse projeto, uma com as informações dos filmes (Tab. 1) e outra com as informações das avaliações que os usuários deram para os filmes (Tab. 2). Nas tabelas estão apresentadas apenas uma amostra dos dados, para ter acesso ao conteúdo completo há um link no final do trabalho.

Tabela 1 - Conjunto de dados sobre filmes.

movieId (int64)	**title (object)**	**genres (object)**
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance

A tabela de filmes apresenta informações como “movieId”: número de identidade para aquele filme, “title”: título do filme e “genres”: gênero do filme. O gênero pode ser classificado em 20 tipos diferentes, sendo eles: ação, aventura, animação, criança, comédia, crime, documentário, drama, fantasia, noir, terror, musical, mistério, romance, ficção científica, suspense, guerra, faroeste, IMAX e não listados.

Tabela 2 - Conjunto de dados sobre avaliações dos usuários.

userId (int64)	movieId (int64)	rating (float64)	timestamp (int64)
1	1	4.0	964982703
1	3	4.0	964981247
1	6	4.0	964982224
1	47	5.0	964983815
1	50	5.0	964982931

Já a tabela de avaliações contém: “userId”: usuário, “movieId”: mesmo número de identificação do filme utilizado na Tab.1, “rating”: nota da avaliação para aquele filme que pode variar de 0-5, “timestamp”: representa o momento que o usuário fez a avaliação.

Como o objetivo do trabalho é o desenvolvimento do sistema de recomendação e aplicação de inteligência artificial, um conjunto de dados com filmes lançados há mais de 5 anos não interferiu nos resultados.

4. Processamento/Tratamento de Dados

Analisando o conjunto de dados de filmes tem-se a coluna de gênero com todos os tipos de gêneros envolvidos em cada filme, porém o texto separado com “|” dificulta a análise da relação entre os gêneros e os filmes. Portanto, foi construída uma matriz esparsa com todos os filmes e os gêneros relacionados, como na figura abaixo.

	movieId	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	...	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller
0	1	0	1	1	1	1	0	0	0	1	...	0	0	0	0	0	0
1	2	0	1	0	1	0	0	0	0	1	...	0	0	0	0	0	0
2	3	0	0	0	0	1	0	0	0	0	...	0	0	0	1	0	0
3	4	0	0	0	0	1	0	0	1	0	...	0	0	0	1	0	0
4	5	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9737	193581	1	0	1	0	1	0	0	0	1	...	0	0	0	0	0	0
9738	193583	0	0	1	0	1	0	0	0	1	...	0	0	0	0	0	0
9739	193585	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0
9740	193587	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0
9741	193609	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0

9742 rows × 21 columns

Figura 1 - Matriz de gênero de filmes.

Para a construção dessa matriz foram necessárias algumas adequações. Primeiramente foi repartida cada linha da coluna gênero em uma lista (Fig. 2) e depois foram criados alguns laços para identificar quais gêneros pertencem à quais filmes (Fig. 3).

	movieId	title	genres
0	1	Toy Story (1995)	[Adventure, Animation, Children, Comedy, Fantasy]
1	2	Jumanji (1995)	[Adventure, Children, Fantasy]

Figura 2 - Lista de gêneros.

```
for index in range(df_movies_modif['genres'].shape[0]):
    for nb_genres in range(len(df_movies_modif['genres'][index])):
        movie_list['movieId'] = df_movies_modif['movieId'][index]
        for item in genres_list:
            if df_movies_modif['genres'][index][nb_genres] == item:
                movie_list[item] = 1

    genres_matrix.loc[index] = movie_list
    movie_list = zerar_lista()

genres_matrix
```

Figura 3 - Trecho de código para matriz esparsa.

A partir dessa matriz foi possível obter a soma de filmes de cada tipo de gênero e construir um gráfico com os gêneros mais assistidos que poderá ser analisado no capítulo seguinte.

Outro pré-processamento foi o da separação do ano de lançamento dos filmes com título. Como pode ser observado na tabela de filmes, o título vem acompanhado do ano de lançamento, e para que possa ser feita uma análise do período que mais lançou filmes ou da tendência de lançamentos, é necessário realizar essa separação.

Em seguida, foi feita a separação do título com a data em formato “(xxxx)” e depois a substituição dos parênteses e possíveis espaçamentos. Após isso, verificou se havia títulos sem datas compatíveis com números e caso houvesse, esses seriam excluídos. Não houve muitos casos sem datas compatíveis e essa exclusão não afetou significativamente o tamanho do *dataset*.

5. Análise e Exploração dos Dados

Primeiramente foi feito uma análise exploratória dos dados para identificar as variáveis, verificar medidas descritivas e esparsidade das matrizes.

Como a tabela de filmes tem informações textuais, as análises descritivas foram feitas na tabela de avaliações (*ratings*) e com isso obteve-se a figura abaixo do boxplot.

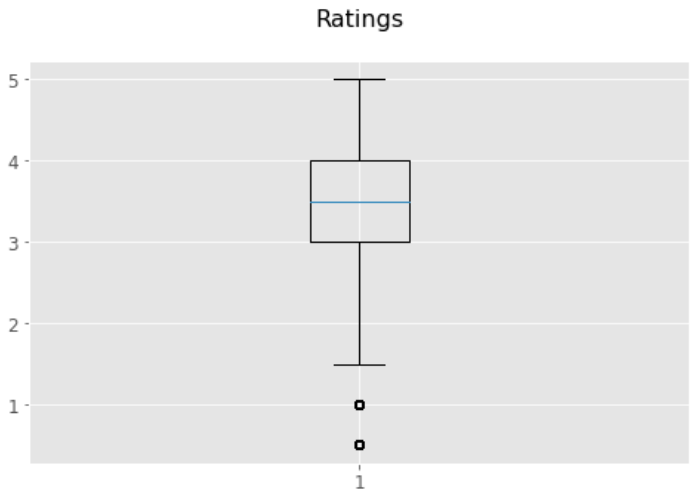

Figura 4 - Boxplot de avaliações.

A partir do boxplot pode ser observado que a média das avaliações foram de 3.5, a nota mínima foi próxima dos 0,5 e 50% dos filmes foram avaliados com notas entre 3 e 4, sendo nota 1 avaliação ruim e 5 muito boa. Além disso, temos alguns valores discrepantes (*outliers*), mas considerando que existem poucas classificações, esses *outliers* não são muito significativos para o trabalho. Para a melhor análise dos ratings, um histograma de frequência foi feito (Fig. 5).

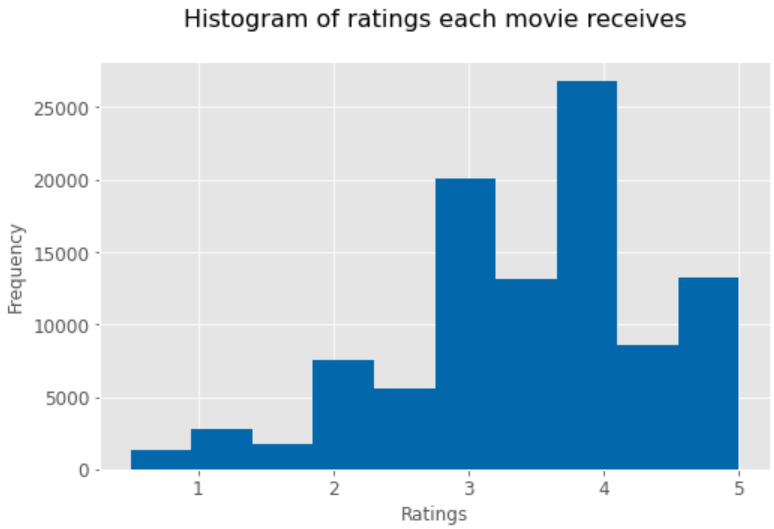

Figura 5 - Histograma de frequência de avaliações.

A partir do histograma é possível observar que a maioria das avaliações foram entre 3 e 4, retificando o que já havia mostrado no boxplot. Além disso obteve-se poucas notas abaixo de 3.

Uma matriz é esparsa quando ela é composta por uma grande quantidade de elementos com valor zero. Para um sistema de recomendação uma matriz muita esparsa é ruim para o sistema, pois este tem uma dificuldade maior de identificar usuários semelhantes. Por exemplo, transformando o histórico de visualização de um usuário em uma matriz, quanto maior a quantidade de elementos preenchidos, melhor é aquela matriz para um sistema de recomendação, identificará mais facilmente com outros usuários e indicar filmes semelhantes.

O cálculo da esparsidade da matriz de avaliações pode ser obtido calculando quantas avaliações faltaram para que todos os usuários avaliassem todos os filmes.

```
n_users = len(df_ratings['userId'].unique())
n_movies = len(df_ratings['movieId'].unique())
n_ratings = len(df_ratings['rating'])

sparsity = 1 - (n_ratings)/float(n_users * n_movies)

print("N° of users:", n_users)
print("N° of movies:", n_movies)
print("N° of ratings:", n_ratings)
print("Data sparsity:", sparsity)

N° of users: 610
N° of movies: 9724
N° of ratings: 100836
Data sparsity: 0.9830003169443864
```

Figura 6 - Cálculo da esparsidade das avaliações.

De acordo com a figura 6, considerando “n_users” o número de usuários do sistema, “n_movies” a quantidade de filmes, “n_ratings” a quantidade de avaliações e “sparsity” a esparsidade das avaliações, obteve-se uma esparsidade de 98%. O número foi muito alto o que significa que haverá dificuldades para identificar usuários com avaliações semelhantes.

Partindo para uma análise dos filmes, foi possível identificar uma tendência da quantidade de filmes assistidos por usuários (Fig. 7), verificando que grande parte dos usuários assistiram poucos filmes. Com esse gráfico obteve-se média de 165 filmes por usuário, mediana de 70,5 e desvio padrão de 265,26. O desvio padrão foi muito alto e a discrepância dos usuários é perceptível.

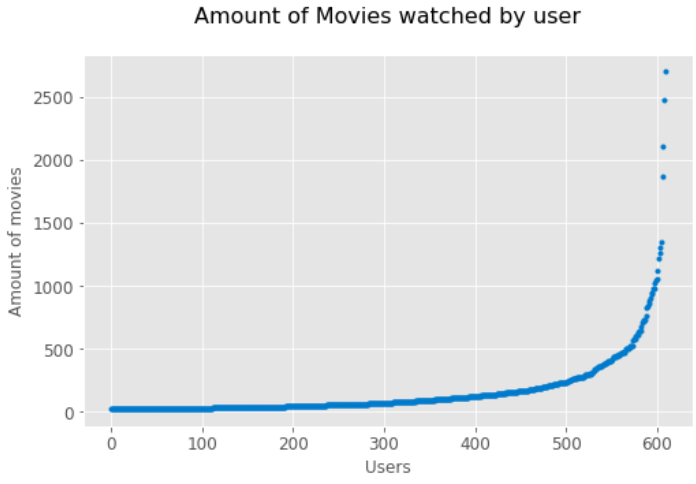

Figura 7 - Quantidade de filmes assistidos por usuário.

Outra análise feita foi a de popularidade dos filmes (Fig. 8). Essa análise foi possível por meio das avaliações dos usuários, portanto filmes com poucas avaliações terão pouca popularidade, o que aconteceu com a maioria dos filmes no conjunto de dados do projeto.

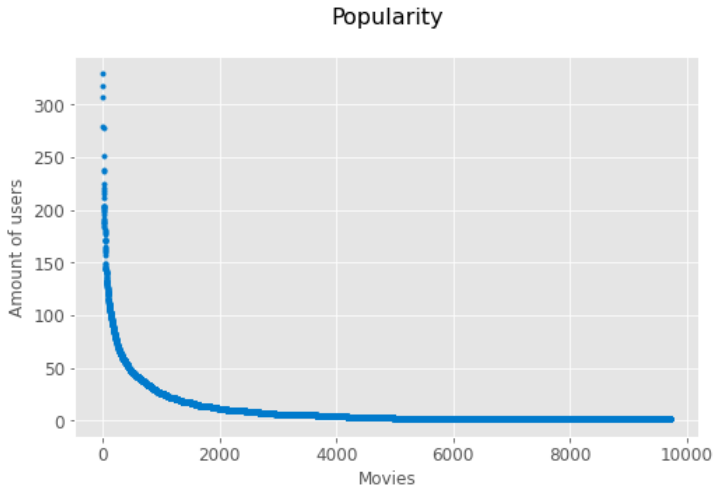

Figura 8 - Popularidade dos filmes.

Também foi feita uma comparação dos gêneros dos filmes e o mais assistido/avaliado foi o filme do tipo drama com mais de 4 mil títulos, seguido pelo de comédia, como pode ser observado na figura abaixo.

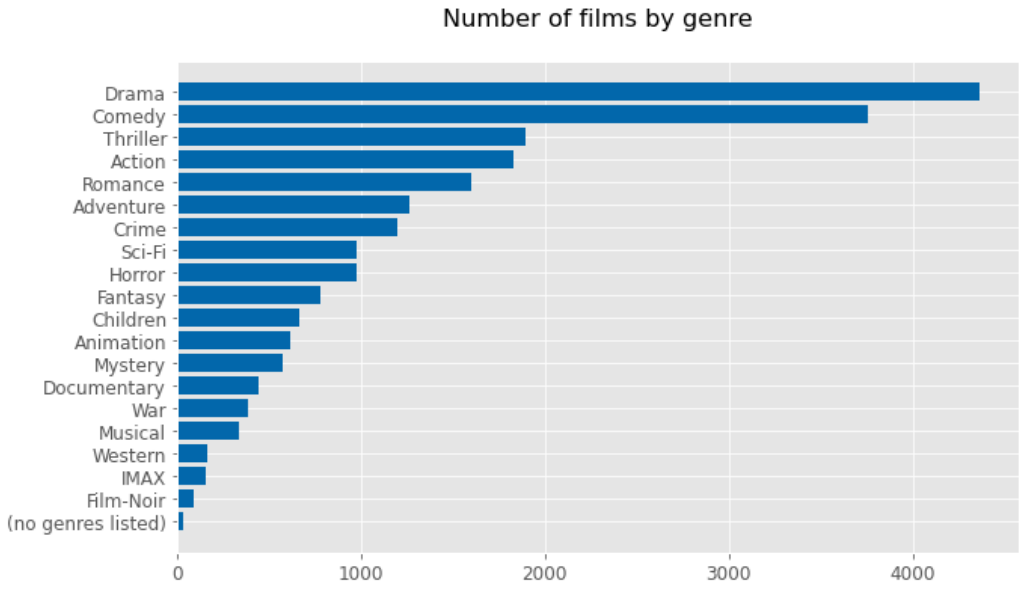

Figura 9 - Número de filmes por gênero.

Para finalizar, foi construído um gráfico com o número de filmes lançados por ano (Fig. 10), onde é possível observar que houve um crescente aumento nos lançamentos de filmes nos últimos 20 anos. O ano que teve mais filmes lançados foi o ano de 2002 com 311 filmes.

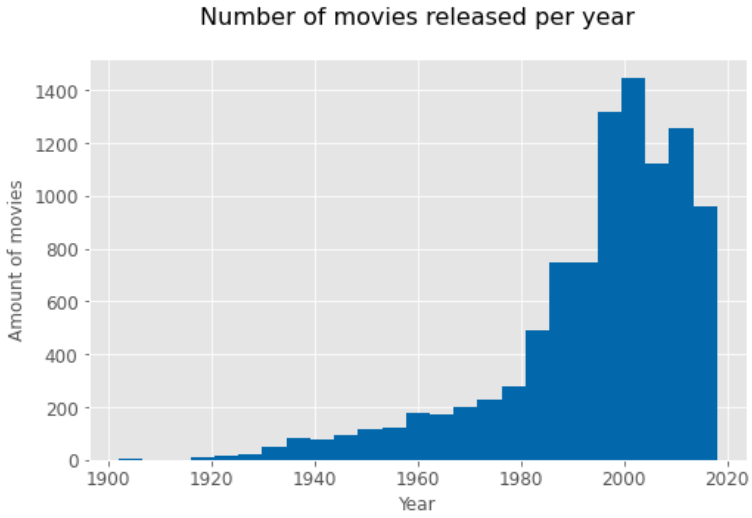

Figura 10 - Número de filmes lançados por ano.

6. Preparação dos Dados para os Modelos de Aprendizado de Máquina

Com o intuito de aplicar um modelo de aprendizado de máquina que faça uma recomendação de filmes de forma personalizada foi necessário fazer algumas preparações dos dados. Primeiramente a base de dados foi dividida em 80% treino e 20% teste para que o modelo possa ser avaliado. Para isso foi utilizado o *train_test_split* da biblioteca *sklearn*. O resultado foi de 322672 dados de treino e 80672 de teste. Para o projeto de recomendação de filmes não foi preciso fazer nenhum acréscimo de dados e nem o balanceamento dele.

Posteriormente, criou-se as matrizes esparsas de avaliações, gêneros e usuários, essas matrizes são importantes para que se possam fazer todas as recomendações de filmes que serão apresentadas nesse projeto.

Outros tipos de mudanças na base de dados foram feitas na aba de tratamento de dados.

7. Aplicação de Modelos de Aprendizado de Máquina

Para a recomendação foram desenvolvidos dois sistemas não personalizados: mais assistidos e mais bem avaliados. E três outros sistemas personalizados para cada usuário: recomendação baseada em conhecimento, conteúdo e filtragem colaborativa.

7.1 Sistemas Não Personalizados

Para os filmes mais assistidos foi feita uma contagem dos itens que foram vistos/avaliados pelos usuários e assim ordenados de forma decrescente. Com a lista dos filmes assistidos foi feita uma comparação com aqueles que o usuário não assistiu para que seja mostrada a recomendação sem repetição de títulos. A Fig.11 mostra um trecho do código utilizado para gerar essa recomendação com a recomendação para alguns usuários.

```
# Setting the recommendations for each user (not to show repeated movies):
recommendationPopular = {}
for user in range(ratings_matrix.shape[0]):
    recommendationPopular[user] = []
    cont = 0
    for item in most_popular:
            if (cont<top_qt):
                if(ratings_matrix[user,item]==0):
                    recommendationPopular[user].append(item)
                    cont+=1
            else:
                break;
```

```
print("Recommendations for userId 10:",recommendationPopular[10])
print("Recommendations for userId 356:",recommendationPopular[356])
print("Recommendations for userId 605:",recommendationPopular[605])
```

```
Recommendations for userId 10: [318, 593, 260, 480, 110, 1, 1198, 527, 589, 1196]
Recommendations for userId 356: [356, 318, 296, 593, 2571, 260, 480, 110, 1, 527]
Recommendations for userId 605: [318, 593, 2571, 1198, 527, 589, 2959, 1196, 50, 2858]
```

Figura 11 - Filmes mais populares.

Assim como os filmes mais assistidos, para recomendação dos bem avaliados foi feita uma ordenação com a média das avaliações para obter uma lista com os itens mais bem avaliados e após isso uma seleção para não mostrar filmes que o usuário já assistiu, como mostra a figura a seguir.

```
# Setting the recommendations for each user (not to show repeated movies):
recommendationBest = {}
for user in range(ratings_matrix.shape[0]):
    recommendationBest[user] = []
    cont = 0
    for item in best_rated:
            if (cont<top_qt):
                if(ratings_matrix[user,item]==0):
                    recommendationBest[user].append(item)
                    cont+=1
            else:
                break;
```

```
print("Recommendations for userId 10:",recommendationBest[10])
print("Recommendations for userId 356:",recommendationBest[356])
print("Recommendations for userId 605:",recommendationBest[605])
```

```
Recommendations for userId 10: [318, 593, 260, 110, 527, 1198, 480, 1196, 50, 1]
Recommendations for userId 356: [318, 356, 296, 593, 2571, 260, 110, 527, 480, 1196]
Recommendations for userId 605: [318, 593, 2571, 527, 1198, 2959, 1196, 50, 589, 858]
```

Figura 12 - Filmes bem avaliados.

Como pode ser observado, a maioria dos filmes populares também estão com as melhores avaliações.

7.2 Recomendação Baseada em Conteúdo

Os sistemas baseados em conteúdo indicam aos usuários, itens com características semelhantes a outros que o mesmo usuário já consumiu. Exemplo: se o consumidor X deu uma ótima avaliação ao filme de criança Toy Story 1, é bem provável que ele queira assistir outros da mesma coleção ou do mesmo gênero de criança.

Para desenvolver a recomendação baseada nesse tipo de sistema foi utilizado o algoritmo de Rocchio. Esse algoritmo ajuda a refinar pesquisas e é muito utilizado em recuperação de informação (WANG C. et al, 2013). Com ele a predição dos dados é baseada na similaridade entre os itens e os usuários e nesse trabalho será feita similaridade por cossenos. Para o algoritmo de Rocchio foi utilizada a seguinte equação:

$$\vec{u} = \frac{1}{|R_u|} \sum_{j \in R_u} r_{uj} \vec{j} \qquad (1)$$

Onde R_u são os itens consumidos pelo usuário “u” e r_{uj} é a avaliação do usuário ao item j. A aplicação da equação pode ser analisada na figura 13.

```
# Normalizing them by the size of user historic
aux = np.dot(ratings_matrix, features_matrix)

for u in range(ratings_matrix.shape[0]):
    # measuring the items nonzero
    nb_nonzero = len(np.nonzero(ratings_matrix[u,:])[1])
    # multiplying this
    if (nb_nonzero != 0):
        users_matrix[u,:] = np.dot(1/float(nb_nonzero), aux[u,:])

users_matrix.shape
```

```
(611, 21)
```

Appling the cossine similarity

```
# consine similarity between each item
prediction_matrix = cosine_similarity(users_matrix, features_matrix)
prediction_matrix.shape
```

Figura 13 - Algoritmo de Rocchio e similaridade de cosseno.

Após a aplicação do modelo foi gerado um conjunto de vetores em que cada um representa a recomendação para cada usuário. A figura 14 apresenta algumas dessas recomendações para o usuário número 10, 356 e 605.

```
print("Recommendations for userId 10:",recommendationContent[10])
print("Recommendations for userId 356:",recommendationContent[356])
print("Recommendations for userId 605:",recommendationContent[605])
```

```
Recommendations for userId 10: [852, 49130, 1457, 3536, 52668, 44864, 5617, 281, 3531, 6218]
Recommendations for userId 356: [72142, 3531, 224, 232, 95199, 106594, 838, 3099, 3097, 84847]
Recommendations for userId 605: [3440, 79139, 45672, 3740, 485, 59103, 87529, 65588, 258, 546]
```

Figura 14 - Recomendação baseada em conteúdo.

7.3 Recomendação Colaborativa

Um sistema baseado em filtragem colaborativa se baseia em comportamentos passados de outros usuários, como avaliações anteriores, para tentar prever qual será a avaliação de um usuário para um item que ele ainda não avaliou. Existem dois tipos de filtros colaborativos: um baseado em memória e outro baseado em modelo. Para esse projeto foi utilizado o baseado em memória. Ele usa as avaliações dos usuários para encontrar similaridades entre os itens e usuários. O método mais usado para se encontrar

essas proximidades é o KNN (*K-Nearest Neighbors*), onde são selecionados K vizinhos mais próximos ao usuário.

Para a aplicação desse modelo foi preciso criar uma outra tabela associando os itens com as avaliações dos usuários. Assim é possível fazer a comparação entre os usuários com relação aos filmes. As primeiras 5 linhas podem ser observadas na figura abaixo.

userId	132	134	364	54	155	78	580	334	156	200	...	481	154	236	360	163	320	535	459	306	184
title																					
Toy Story (1995)	2.0	3	5	3	3	4.0	3.0	3.5	4.0	3.5	...	0	0.0	0	0	0	0.0	0	0.0	0.0	0.0
Jumanji (1995)	0.0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0	0	0	0.0	0	0.0	0.0	0.0
Grumpier Old Men (1995)	0.0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0	0	0	0.0	0	0.0	0.0	0.0
Waiting to Exhale (1995)	0.0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	...	0	0.0	0	0	0	0.0	0	0.0	0.0	0.0
Father of the Bride Part II (1995)	0.0	0	0	0	0	0.0	0.0	0.0	0.0	4.0	...	0	0.0	0	0	0	0.0	0	0.0	0.0	0.0

5 rows × 610 columns

Figura 15 - Tabela de avaliações dos filmes por usuário.

Para a recomendação baseada no algoritmo KNN foi criado a seguinte função que calcula a relação dos filmes baseado nas avaliações dos usuários:

```
def CollabRecommendation (movie_id):
    movie_id = movie_id-1
    # Applying the model with cosine as metric
    model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
    model_knn.fit(pivot_user_movie_matrix)
    distances, indices = model_knn.kneighbors(pivot_user_movie.iloc[movie_id,:].values.reshape(1,-1), n_neighbors = 10)

    movie = []
    distance = []

    for i in range(0, len(distances.flatten())):
        if i != 0:
            movie.append(pivot_user_movie.index[indices.flatten()[i]])
            distance.append(distances.flatten()[i])

    m=pd.Series(movie,name='title')
    d=pd.Series(distance,name='distance')
    recommendationCollab = pd.concat([m,d], axis=1)
    recommendationCollab = recommendationCollab.sort_values('distance',ascending=True)

    return recommendationCollab
```

Figura 16 - Recomendação colaborativa utilizando KNN.

Para motivos de comparação também foi desenvolvido uma recomendação baseada no algoritmo SVD (*Singular Vector Decomposition*) e utilizando o coeficiente correlação de Pearson. Como a quantidade de dados é muito grande foi utilizado o TruncatedSVD da biblioteca Sklearn para comprimir a matriz de usuários em 12 componentes arbitrariamente, representando uma visão generalizada do sistema. O algoritmo para a recomendação é representado na figura 17.

```
# get the index of the popular movie
movie_names = pivot_user_movie.index
movie_list = list(movie_names)

search_movie = movie_list.index(df_movies[df_movies['movieId'] == movie_id]['title'].item())
corr_search_movie = corr_matrix[search_movie]

print('Recommendations for {0}:\n'.format(pivot_user_movie.index[movie_id-1]))
list(movie_names[(corr_search_movie < 1.0) & (corr_search_movie > 0.95)])
```

Figura 17 - Recomendação colaborativa utilizando SVD.

7.4 Recomendação Baseada em Conhecimento

Para situações que tem uma quantidade baixa de avaliações ou informações dos usuários, as recomendações acima não serão tão eficazes pois elas precisam de muitos dados para fazer as similaridades. Os sistemas de recomendação baseados em conhecimento ajudam a suprir essa área, pois o usuário consegue filtrar o que ele precisa no sistema e a partir desse requerimento, o sistema pode sugerir algumas recomendações similares também. Uma forma de acrescentar mais de um tipo de filtragem.

Primeiramente foi feito um filtro de filmes com as maiores avaliações por gênero que o usuário escolher, como na figura a seguir que utilizou o gênero aventura (*adventure*):

```
search_genre = 'Adventure'
df_search = genres_matrix[['movieId', search_genre]]
df_search = df_search[df_search[search_genre] == 1]
df_search_genre = pd.merge(mean_rating, df_search, how = 'inner', on = 'movieId').sort_values(by='mean',ascending=False)
pd.merge(df_search_genre, df_movies, how = 'inner', on = 'movieId')
```

	movieId	mean	Adventure	title	genres	year
0	102194	5.0	1	Mud (2012)	[Adventure, Crime, Drama]	2012
1	172585	5.0	1	Karlson Returns (1970)	[Adventure, Animation, Children]	1970
2	141816	5.0	1	12 Chairs (1976)	[Adventure, Comedy]	1976
3	91355	5.0	1	Asterix and the Vikings (Astérix et les Viking...	[Adventure, Animation, Children, Comedy, Fantasy]	2006
4	3687	5.0	1	Light Years (Gandahar) (1988)	[Adventure, Animation, Fantasy, Sci-Fi]	1988
...	...	...	...	...	...	...
1192	65350	0.5	1	General Died at Dawn, The (1936)	[Adventure, Crime, Thriller]	1936
1193	135216	0.5	1	The Star Wars Holiday Special (1978)	[Adventure, Children, Comedy, Sci-Fi]	1978
1194	104017	0.5	1	3 dev adam (Three Giant Men) (1973)	[Action, Adventure, Sci-Fi]	1973
1195	152063	0.5	1	Gods of Egypt (2016)	[Adventure, Fantasy]	2016
1196	53453	0.5	1	Starcrash (a.k.a. Star Crash) (1978)	[Action, Adventure, Fantasy, Sci-Fi]	1978

Figura 18 - Filtragem por conhecimento (gênero).

Também foi feito um filtro baseado em filme que o usuário escolher, como por exemplos filmes com avaliações parecidos com Toy Story 2 (Fig. 19).

```
search_movie = 'Toy Story 2 (1999)'
```

```
item_movieId = df_movies[df_movies['title']==search_movie].movieId.item()
CollabRecommendation(item_movieId)
```

	title	distance
0	Shattered Glass (2003)	0.462885
1	Adventures of Baron Munchausen, The (1988)	0.496648
2	Lantana (2001)	0.523475
3	Pieces of April (2003)	0.537013
4	Husbands and Wives (1992)	0.537018
5	Nine to Five (a.k.a. 9 to 5) (1980)	0.548803
6	Michael Collins (1996)	0.562868
7	Igby Goes Down (2002)	0.565791
8	Breakfast at Tiffany's (1961)	0.571683

Figura 19 - Filtragem por filmes.

8. Avaliação dos Modelos de Aprendizado de Máquina e Discussão dos Resultados

A avaliação do modelo de aprendizado de máquina é importante para que possa determinar o desempenho do modelo na previsão dos resultados. Para fazer essa avaliação é necessário que os dados sejam diferentes dos dados usados no treinamento do modelo, pois modelos podem “memorizar” os dados e não os generalizar (Amazon, 2023).

Para a avaliação dos modelos foram utilizadas métricas de Hit-Rate, precisão e revocação e MRR (*Mean Reciprocal Rank*). O Hit-Rate representa a média do número de itens recomendados que estão no conjunto de teste de cada usuário, quanto maior o valor, melhor. A precisão representa quantas avaliações foram acertadas dentre todas as classificações positivas do modelo. A revocação indica, das amostras existentes, quantas o modelo conseguiu acertar corretamente. E o MRR avalia a qualidade do modelo, verifica a relevância dos itens, quanto mais próximo de 1, melhor.

Tabela 3 - Resultado das avaliações dos modelos.

	Hit-Rate	**Precisão**	**Revocação**	**MRR**
Populares	1,6059	0,1182	0,0926	0,3083
Mais bem avaliados	1,6420	0,1236	0,0976	0,3149
Recomendação baseada em conteúdo	0,0853	0,0049	0,0045	0,0099
Colaborativa (KNN)	0,3284	0,0237	0,0125	0,0448
Colaborativa (SVD)	0,0755	0,0089	0,0042	0,0199

Devido ao fato de recomendação dos filmes mais populares, mais bem avaliados e baseada em conteúdo ter sido feita usando somente um método, não tem como fazer uma comparação entre as avaliações dos algoritmos. Porém para fazer a recomendação colaborativa foi usado dois tipos de métodos diferentes, KNN e SVD, fazendo a comparação dos dois e identificando deles se comportou melhor com os dados do trabalho em questão.

Comparando a avaliação do método KNN com o SVD pode-se identificar que o KNN teve melhores resultados e, portanto, será utilizado no sistema de recomendação de filmes.

9. Conclusão

Este trabalho propôs o desenvolvimento de um sistema de recomendação de filmes utilizando diferentes tipos de métodos, com o objetivo de disponibilizar para o usuário um sistema personalizado e com filmes relevantes.

Os objetivos do trabalho foram cumpridos. Foram utilizadas recomendações baseadas em conteúdo, conhecimento e colaboração e para tal fim utilizou-se modelos de aprendizado de máquina. Utilizou-se dois tipos de algoritmos diferentes para a recomendação baseada em colaboração e a partir dos resultados o algoritmo KNN apresentou melhor performance com os dados do sistema.

Para um trabalho futuro será desenvolvida uma interface gráfica para possibilitar ao usuário fazer as pesquisas de filmes e ter as opções mais relevantes e baseadas em escolhas passadas. Além da pesquisa de filmes e recomendações, também será possível fazer avaliações de novos títulos.

10. Links

Link do repositório do GitHub onde pode ser encontrado o código desenvolvido para o projeto: https://github.com/marcelacoury/Movie-Recommendation-System.