Métodos Numéricos
Projeto da unidade 2

Equipe: Daniel Lima Neto , Eduardo Henrique Lima de Morais e Igor Kádson de Souza Oliveira

1. INTRODUÇÃO

Neste projeto será abordado o desenvolvimento de um sistema de recomendação de filmes utilizando técnicas de aprendizado de máquina e análise de dados. Um sistema de recomendação tem como objetivo sugerir itens relevantes ao usuário com base em suas preferências e decisões passadas, ele é bastante Utilizado em plataformas como Amazon, Netflix e Spotify. Técnicas de filtragem colaborativa e fatoração de matriz, como o método Singular Value Decomposition (SVD), serão exploradas para gerar recomendações personalizadas. Além disso, a visualização de dados e a criação de uma interface interativa permitirão uma experiência mais intuitiva para os usuários. O foco deste projeto é integrar essas técnicas em uma solução completa que possa ser aplicada ao conjunto de dados MovieLens, amplamente utilizado para estudos sobre sistemas de recomendação.

2. Descrição do problema

O problema central deste projeto é ajudar os usuários a descobrir novos filmes que provavelmente gostarão, com base em suas classificações anteriores e no comportamento de outros usuários com preferências semelhantes. Com o aumento do catálogo de filmes disponíveis em plataformas de streaming, encontrar filmes relevantes pode ser um desafio. Portanto, a tarefa é construir um modelo capaz de prever as classificações dos filmes que o usuário ainda não viu, e sugerir os mais bem avaliados.

As principais características do problema incluem:

- Alto volume de dados de classificações.
- Ausência de classificação para a maioria dos filmes (sparsidade dos dados).
- A necessidade de personalização de recomendações.
- A dificuldade de melhorar a experiência do usuário ao filtrar um grande número de opções de filmes.

Esses problemas são vastamente encontrados em sistemas de recomendação de e-commerce, mídia digital e até na criação de playlists personalizadas.

3. MÉTODOS APLICADOS À SOLUÇÃO

Para solucionar o problema, usaremos o métodos:

1. **Filtragem colaborativa:** Esse método utiliza a interação de outros usuários com características semelhantes para recomendar ao usuário, usamos uma variante chamada de fatoração de matriz.
2. **Singular Value Decomposition (SVD)**: O SVD é uma técnica de fatoração de matriz que decompõe a matriz de classificações (usuários x filmes) em componentes menores, ele permite trabalhar com classificações ausentes. Ele é usado para lidar com sparsidade dos dados uma vez que gera uma representação compacta dos padrões de comportamento dos usuários.
3. **Similaridade de Cosseno:** Usaremos a similaridade de cossenos para calcular a similaridade entre usuários e então medir o grau de semelhança entre dois usuários com base em suas classificações de filmes. isso vai permitir que seja sugerido filmes para usuários semelhantes tenham gostado.

Esses métodos serão úteis para resolver os problemas citados, pois nos permite lidar com alto volumes de dados, sparsidade dos dados e melhora nas recomendações que serão feitas para os usuários.

4. IMPLEMENTAÇÃO

a. Carregamento e Preparação dos Dados

Iniciamos o projeto carregando os arquivos `ratings.csv` e `movies.csv`, que contém informações sobre as avaliações dos usuários e os filmes disponíveis, respectivamente. Durante este processo, duplicatas são removidas para garantir a integridade dos dados.

```
ratings = pd.read_csv('/home/suporte/1234/ratings.csv')
movies = pd.read_csv('/home/suporte/1234/movies.csv')
ratings = ratings.drop_duplicates()
ratings['userId'] = ratings['userId'].astype(int)
ratings['movieId'] = ratings['movieId'].astype(int)
ratings['rating'] = ratings['rating'].astype(float)
movies = movies.drop_duplicates()
movies['movieId'] = movies['movieId'].astype(int)
movies['title'] = movies['title'].astype(str)
```

b. Amostragem de Dados

Para otimizar o processamento, uma amostra de 70% dos dados de avaliações e filmes foi selecionada. Além disso, 100 usuários e 100 filmes foram escolhidos aleatoriamente para compor a base de dados reduzida.

```
ratings_sample = ratings.sample(frac=0.7, random_state=42)
movies_sample = movies.sample(frac=0.7, random_state=42)
unique_users = ratings['userId'].drop_duplicates().sample(n=100, random_state=42)
unique_movies = ratings['movieId'].drop_duplicates().sample(n=100, random_state=42)
```

c. Criação da Matriz de Usuários e Filmes

A matriz de avaliações foi criada utilizando o método `pivot`, que organiza as avaliações em um formato onde cada linha representa um usuário e cada coluna representa um filme. As avaliações ausentes foram preenchidas com zeros.

```
ratings_matrix = ratings_sample.pivot(index='userId', columns='movieId', values='rating').fillna(0)
```

d. Aplicação de SVD

A técnica de Singular Value Decomposition (SVD) foi aplicada à matriz de usuários e filmes. Esta técnica permite reduzir a dimensionalidade dos dados e extrair características principais, utilizando apenas um componente.

```
svd = TruncatedSVD(n_components=1, random_state=42)
matrix_svd = svd.fit_transform(ratings_matrix)
print("Forma da matriz SVD:", matrix_svd.shape)
```

e. Calcular a Média de Avaliações por Filme

A média das avaliações foi calculada para cada filme, permitindo a identificação dos 10 filmes mais bem avaliados. Isso oferece uma visão geral dos filmes com melhor desempenho.

```
ratings_with_titles = ratings.merge(movies[['movieId', 'title']], on='movieId')
average_ratings = ratings_with_titles.groupby('title')['rating'].mean()
top_movies = average_ratings.nlargest(10).reset_index()
```

f. Filmes Mais Populares

A popularidade dos filmes foi analisada com base no número de avaliações recebidas, e os 10 filmes mais populares foram destacados.

```
popularity = ratings_with_titles.groupby('title')['rating'].count()
top_popular_movies = popularity.nlargest(10)
for title, count in top_popular_movies.items():
    print(f"{title}: {count} ratings")
```

g. Filtragem de Filmes por Gênero

Um filme específico, como "Toy Story (1995)", foi escolhido para filtrar outros filmes com gêneros semelhantes. A matriz de gêneros foi criada e SVD foi aplicada, reduzindo a dimensionalidade para dois componentes.

```
chosen_movie = "Toy Story (1995)"
chosen_movie_genres = movies.loc[movies['title'] == chosen_movie, 'genres'].values[0]
chosen_genres = set(chosen_movie_genres.split('|'))
movies_with_similar_genres = movies[movies['genres'].apply(lambda x: len(set(x.split('|')).intersection(chosen_genres)) > 0)]
genres_matrix = movies_with_similar_genres['genres'].str.get_dummies(sep='|')
movies_genre_matrix = pd.concat([movies_with_similar_genres[['movieId']], genres_matrix], axis=1).set_index('movieId')
svd = TruncatedSVD(n_components=2, random_state=42)
matrix_svd = svd.fit_transform(movies_genre_matrix)
```

h. Visualização da Redução de Dimensionalidade

A visualização dos dados após a redução de dimensionalidade foi realizada com um gráfico de dispersão. O filme escolhido foi destacado em vermelho, enquanto os filmes próximos foram destacados em azul.

```
plt.figure(figsize=(10, 6))
plt.scatter(svd_df['Component 1'], svd_df['Component 2'], alpha=0.5)
plt.scatter(svd_df.loc[chosen_movie_id, 'Component 1'], svd_df.loc[chosen_movie_id, 'Component 2'], color='red', label=chosen_movie, s=100)
for movie_id in closest_movie_ids:
    if movie_id != chosen_movie_id:
        plt.scatter(svd_df.loc[movie_id, 'Component 1'], svd_df.loc[movie_id, 'Component 2'], color='blue', label=movies.loc[movies['movieId'] == movie_i
plt.show()
```

i. Recomendação de Filmes com Base em Gêneros

Uma função foi implementada para recomendar filmes com base na similaridade de gêneros. O filme escolhido é comparado com outros filmes, e aqueles com mais gêneros em comum são sugeridos.

```
def recommend_movies_by_genre(movie_title, n_recommendations=5):
    chosen_movie_genres = movies[movies['title'].str.contains(movie_title, case=False, na=False)]['genres'].values
    if len(chosen_movie_genres) == 0:
        print(f"Erro: O filme '{movie_title}' não foi encontrado.")
        return pd.Series([])
    chosen_movie_genres = chosen_movie_genres[0].split('|')
    def genre_similarity(genres):
        movie_genres = genres.split('|')
        return len(set(movie_genres).intersection(set(chosen_movie_genres)))
    movies['similarity_score'] = movies['genres'].apply(genre_similarity)
    recommended_movies = movies[movies['title'] != movie_title].sort_values(by='similarity_score', ascending=False)
    return recommended_movies[['title', 'genres']].head(n_recommendations)
```

j. Exibição dos Filmes Mais Próximos

Por fim, a função de recomendação é chamada com o filme "Pulp Fiction" como exemplo, exibindo os filmes recomendados com base nos gêneros.

```
chosen_movie2 = "Pulp Fiction"
recommended = recommend_movies_by_genre(chosen_movie2, n_recommendations=100)
for idx, row in recommended.iterrows():
    print(f"- {row['title']} (Gêneros: {row['genres']})")
```

k. Conclusão

O sistema de recomendação de filmes foi implementado com sucesso, utilizando SVD para redução de dimensionalidade e filtragem de filmes por gênero. As análises realizadas permitiram identificar filmes populares e bem avaliados, além de oferecer recomendações personalizadas aos usuários com base em suas preferências. Esse projeto pode ser expandido com a inclusão de mais dados, técnicas de aprendizado de máquina mais sofisticadas e melhorias na interface do usuário.