Universidade Presbiteriana Mackenzie

Tecnologia em Ciência de Dados - Projeto Aplicado 3

Matheus Vinícius Gomes

Leandro Rodrigues Dos Santos

Luiz Eduardo de Mei Salvador Coelho

Matheus Neves de Castro

Projeto Aplicado III

Sistema de Recomendação de Filmes para Plataforma de Streaming

SÃO PAULO

2º Semestre de 2024

Autores:
Matheus Vinícius Gomes / 10408179 / mathvgomes@gmail.com
Leandro Rodrigues Dos Santos / 23019689 / lers.138@gmail.com
Luiz Eduardo de Mei Salvador Coelho / 23024585 / Luiz02coelho@gmail.com
Matheus Neves de Castro / 10415190 / 10415190@mackenzista.com.br

Projeto Aplicado III

Sistema de Recomendação de Filmes para Plataforma de Streaming

Professora: CAROLINA TOLEDO FERRAZ

SÃO PAULO

2º Semestre de 2024

Resumo (deverá ser elaborado apenas no final do projeto.)

Sumário

Introdução……..5

Referencial teórico……6

Metodologia……12

Resultados……13

Conclusão e trabalhos futuros……14

Introdução

Sistema de Recomendação de Filmes para Plataforma de Streaming

Com o crescimento acelerado das plataformas de streaming, os usuários enfrentam o desafio de escolher entre milhares de filmes disponíveis. A diversidade de gêneros, estilos e diretores faz com que essa tarefa se torne muitas vezes cansativa, levando à sobrecarga de opções e à dificuldade de encontrar conteúdos relevantes para cada perfil. Sistemas de recomendação de filmes surgem como uma solução fundamental para guiar os usuários em suas escolhas, proporcionando uma experiência mais personalizada e eficiente ao indicar títulos com base em suas preferências e comportamentos anteriores.

A motivação para este projeto decorre da necessidade de melhorar a experiência dos usuários nas plataformas de streaming, facilitando a descoberta de filmes que correspondam aos seus gostos e expectativas. Com uma recomendação automatizada e personalizada, espera-se aumentar a satisfação dos usuários e o tempo de engajamento na plataforma, além de incentivar a exploração de novos filmes que de outra forma poderiam passar despercebidos.

O desenvolvimento de um sistema de recomendação de filmes é justificado pela crescente demanda por soluções que ajudem a gerenciar a vasta quantidade de opções oferecidas nas plataformas de streaming. Um sistema de recomendação eficiente não apenas facilita a navegação, mas também contribui para uma experiência mais envolvente e personalizada, impulsionando tanto a retenção de usuários quanto o sucesso da plataforma em um mercado altamente competitivo.

O objetivo geral deste projeto é desenvolver um sistema de recomendação de filmes que ofereça sugestões personalizadas com base nas preferências e comportamentos dos usuários. Os objetivos específicos incluem:

1. **Analisar e explorar dados de consumo de filmes:** Realizar uma análise exploratória dos dados de usuários e suas interações com o catálogo de filmes.
2. **Preparar e limpar os dados:** Implementar técnicas de pré-processamento e limpeza dos dados para garantir que o modelo tenha uma base confiável e bem estruturada.
3. **Selecionar e aplicar técnicas de Machine Learning:** Escolher e implementar algoritmos de aprendizado de máquina apropriados para criar um modelo de recomendação eficaz.
4. **Avaliar o desempenho do modelo:** Definir e aplicar métricas para medir a precisão e a relevância das recomendações, ajustando o modelo conforme necessário.
5. **Documentar e apresentar os resultados:** Registrar as etapas e resultados do projeto, ressaltando os impactos na experiência do usuário e as possíveis melhorias futuras.

Referencial teórico

Definição de bibliotecas

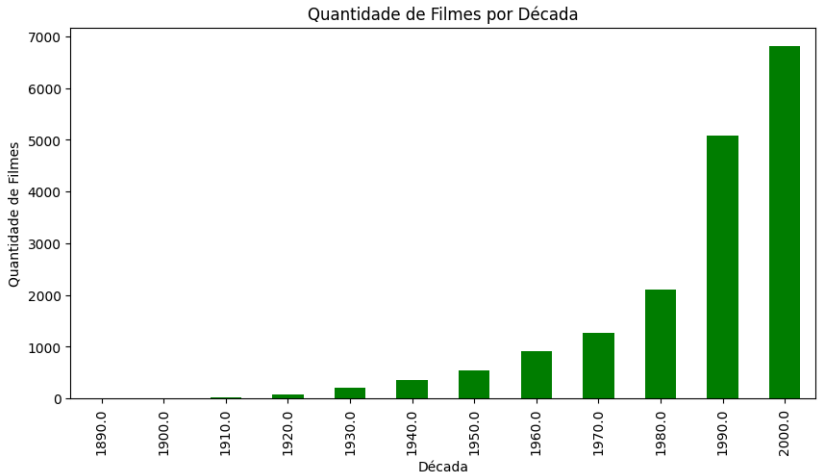
Para este trabalho, usaremos algumas bibliotecas python para adquirir, tratar, visualizar e modelar os nossos dados, sendo elas:

Fase	Bibliotecas
Adquirir Dados	kaggle
Tratar Dados	os, shutil, pandas, numpy
Visualizar os dados	matlotlib, seaborn
Modelar	Surprise (svd)

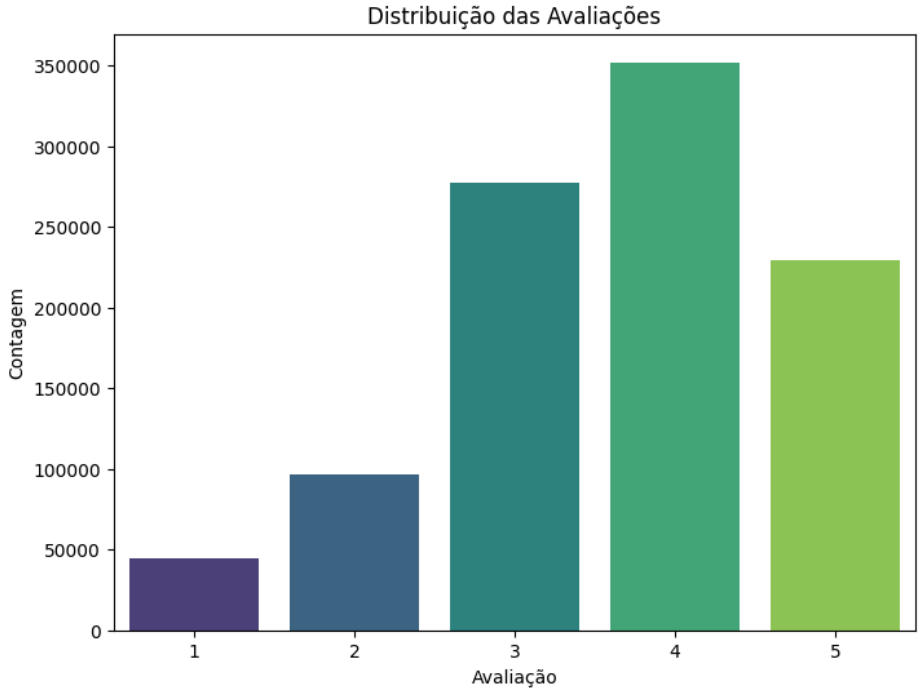
Explorando os dados

A Netflix revolucionou o mundo do entretenimento, inicialmente enviando DVDs às casas dos clientes e evoluindo para uma plataforma de streaming de conteúdo, e posteriormente produtora e desenvolvedora de filmes e roteiros originais. Em uma competição do kaggle, a Netflix decidiu abrir os seus dados com o desafio de quem conseguia fazer o melhor algoritmo de recomendação com um prêmio de $1MM de dólares. Esses dados estão divididos entre avaliações dos usuários aos filmes, uma tabela de informações sobre o filme e podem ser adquiridos publicamente no site do kaggle:
https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data/data

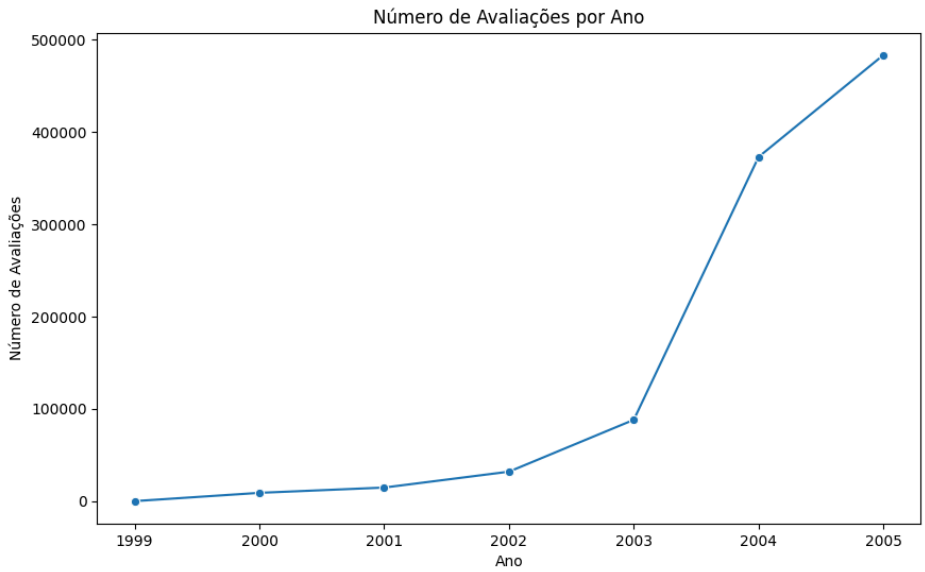
A empresa, lançou muitos filmes (17.026 até a época do lançamento do dataset) ao longo do tempo, com conteúdo de diversas eras tentando atingir todos os públicos, das crianças aos idosos:



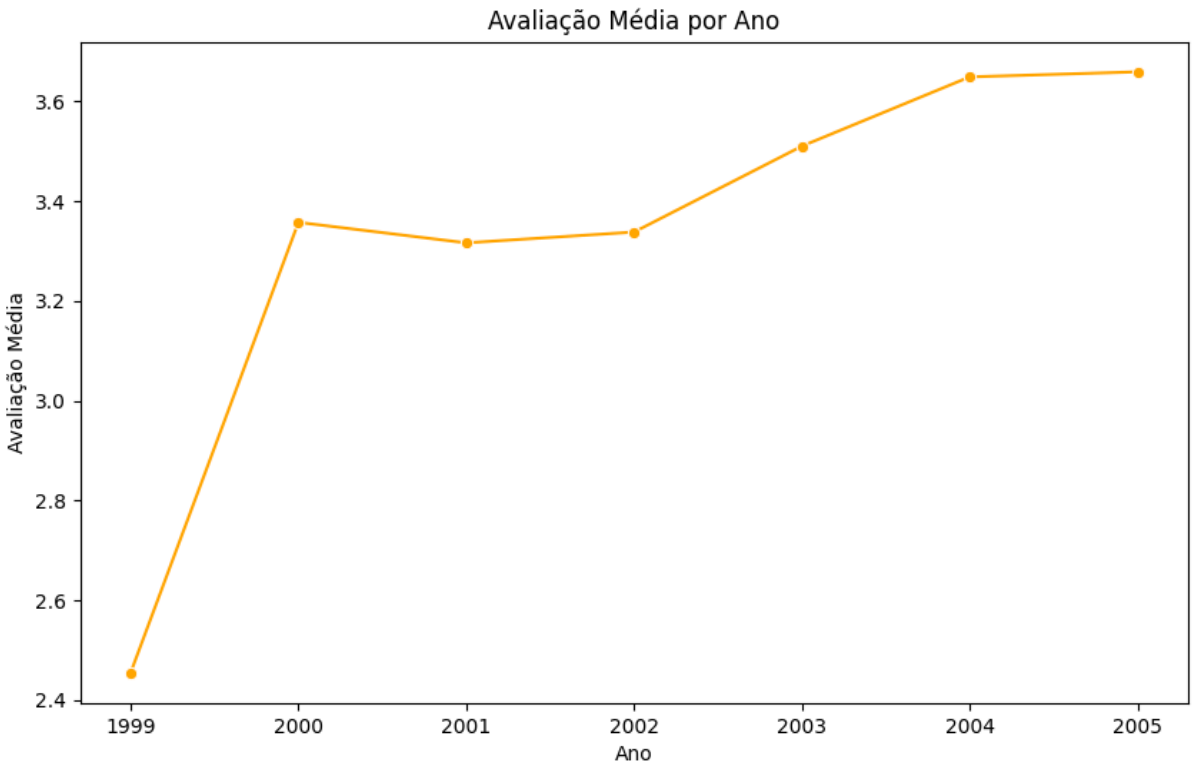
Os dados de avaliações dos filmes são gigantescos e contém, por ID de usuário, o filme a data da avaliação e a nota em si. Os usuários podem dar notas de 0 a 5 e essa é a distribuição:



No começo do serviço, as avaliações eram menos comuns, mas, isso mudou com o tempo:



Parece também que os usuários ficaram mais satisfeitos com o conteúdo ao longo do tempo, indicando uma possível causa como a qualidade das ofertas personalizadas que os algoritmos fazem em detrimento a escolhas aleatórias:



Sabendo da quantidade massiva de informações que temos a disposição, vamos seguir para o treinamento e teste de performance do modelo.

Tratar e preparar a base de dados para o treinamento.

Sabendo da quantidade massiva de informações que temos a disposição, vamos seguir para o tratamento da base para o treinamento do modelo.

Os dados do Netflix Prize contam com um conjunto chamado “probe” que consiste em clientes com filmes que estão no conjunto de treinamento, mas sem a coluna de avaliação visível, que servirá como amostra de testes.

Já para o treinamento, usaremos a combinação de dados dos clientes com mais de 10 avaliações, para evitar sparsity e melhorar a performance do modelo.

Tratando os dados txt:

```
# Carregando o arquivo de avaliações (combined_data_1.txt)
data_path = 'combined_data_1.txt'

# Criar um DataFrame vazio para armazenar os dados
data = {'MovieID': [], 'CustomerID': [], 'Rating': [], 'Date': []}

# Lendo o arquivo linha por linha
with open(data_path, 'r') as file:
    movie_id = None
    for line in file:
        line = line.strip()
        if line.endswith(':'):
            # Linha com o MovieID
            movie_id = int(line.replace(':', ''))
        else:
            # Linha com avaliação (CustomerID, Rating, Date)
            customer_id, rating, date = line.split(',')
            data['MovieID'].append(movie_id)
            data['CustomerID'].append(int(customer_id))
            data['Rating'].append(float(rating))
            data['Date'].append(date)

# Convertendo o dicionário para um DataFrame
df = pd.DataFrame(data)

# Exibir as primeiras linhas
df.head()
```

Filtrando usuários com, no mínimo, 10 avaliações:

```
# Definindo limites mínimos
min_ratings_per_user = 10
min_ratings_per_movie = 10

# Filtrando usuários que avaliaram menos de 10 filmes
filtered_users = df['CustomerID'].value_counts()[df['CustomerID'].value_counts() >= min_ratings_per_user].index
df = df[df['CustomerID'].isin(filtered_users)]

# Filtrando filmes que receberam menos de 10 avaliações
filtered_movies = df['MovieID'].value_counts()[df['MovieID'].value_counts() >= min_ratings_per_movie].index
df = df[df['MovieID'].isin(filtered_movies)]

# Exibindo as dimensões após o filtro
df.shape
```

Aplicando a separação dos conjuntos de teste e treinamento com o arquivo probe:

```python
# Carregar o arquivo probe.txt
probe_path = 'probe.txt'

# Lendo o arquivo probe
probe_data = {'MovieID': [], 'CustomerID': []}

with open(probe_path, 'r') as file:
    movie_id = None
    for line in file:
        line = line.strip()
        if line.endswith(':'):
            # Linha com o MovieID
            movie_id = int(line.replace(':', ''))
        else:
            # Linha com o CustomerID
            customer_id = int(line)
            probe_data['MovieID'].append(movie_id)
            probe_data['CustomerID'].append(customer_id)

# Convertendo para DataFrame
df_probe = pd.DataFrame(probe_data)

# Criar uma coluna que indique se os dados estão no probe
df['is_probe'] = df[['MovieID', 'CustomerID']].apply(
    lambda x: 1 if ((x['MovieID'], x['CustomerID']) in zip(df_probe['MovieID'], df_probe['CustomerID'])) else 0, axis=1)

# Dividir em treino e teste
df_train = df[df['is_probe'] == 0]
df_test = df[df['is_probe'] == 1]

# Remover a coluna auxiliar
df_train.drop(columns='is_probe', inplace=True)
df_test.drop(columns='is_probe', inplace=True)

# Verificar tamanhos dos conjuntos de treino e teste
df_train.shape, df_test.shape
```

Esse tratamento nos permitiu ter 968858, 4 em treino e 30919, 4 em teste.

Técnica de treinamento

Para o treinamento, vamos usar a técnica de filtragem colaborativa baseada em fatores latentes, com o algoritmo SVD. O conjunto de treinamento será usado para o modelo aprender as relações entre usuários e filmes. Essa abordagem é baseada em parelhar usuários com perfis semelhantes (em termos de padrões de avaliação).

Para isso, com a biblioteca surprise, implementaremos sobre os dados tratados e separados entre treino e teste, o seguinte algoritmo:

```python
from surprise import SVD, Dataset, Reader
from surprise.model_selection import train_test_split, accuracy

# Configurar os dados para a biblioteca Surprise
reader = Reader(rating_scale=(1, 5))

# Preparar o conjunto de treino para Surprise
train_data = Dataset.load_from_df(df_train[['CustomerID', 'MovieID', 'Rating']], reader)

# Preparar o conjunto de teste para Surprise
test_data = list(zip(df_test['CustomerID'], df_test['MovieID'], df_test['Rating']))

# Carregar o conjunto de treino
trainset = train_data.build_full_trainset()

# Inicializar o modelo SVD
model = SVD()

# Treinar o modelo
model.fit(trainset)

# Fazer previsões para o conjunto de teste
predictions = model.test(test_data)

# Avaliar o desempenho do modelo com RMSE
accuracy.rmse(predictions)
```

Avaliando o desempenho do modelo.

Escolhemos como métrica de avaliação o RMSE, que avalia a diferença entre as previsões e as avaliações reais feitas pelo modelo. Ele se baseia no cálculo da raiz quadrada da média dos erros ao quadrado, penalizando erros maiores em detrimento aos menores.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2}$$

Como resultados iniciais, nosso modelo alcançou uma acurácia de 1.0565, revelando uma margem significativa para otimização e obtenção de resultados mais precisos.

Metodologia

Resultados

Conclusão e trabalhos futuros