Universidade Presbiteriana Mackenzie

Tecnólogo em Ciência de Dados

Projeto Aplicado III

SISTEMA DE RECOMENDAÇÃO DE LIVROS

e-LITER

CAIO HENRIQUE SESTARI

DIANNA MAYUMI SANTOS KATAYAMA RODRIGUES

EMERSON MOREIRA BALIZA

NICOLAS PINOTTI

PEDRO AUGUSTO ALVES MACENA LIMA

RICARDO PARDONO

São Paulo

2024

Universidade Presbiteriana Mackenzie

Tecnólogo em Ciência de Dados

Projeto Aplicado III

SISTEMA DE RECOMENDAÇÃO DE LIVROS

e-LITER

Projeto apresentado para a disciplina.
Projeto Aplicado III - Etapa 4.
Orientador: Prof. Thiago Donizetti dos Santos.

CAIO HENRIQUE SESTARI - 10407008

DIANNA MAYUMI SANTOS KATAYAMA RODRIGUES - 10407132

EMERSON MOREIRA BALIZA - 10369752

NICOLAS PINOTTI - 10408010

PEDRO AUGUSTO ALVES MACENA LIMA - 10407713

RICARDO PARDONO - 10408248

São Paulo

2024

Universidade Presbiteriana Mackenzie

RESUMO

O aumento exponencial da disponibilidade de informação online redefiniu a interação humana com o conteúdo digital, especialmente no domínio literário. A vasta oferta de livros acessíveis com um simples clique trouxe consigo o desafio da escolha, levando os leitores a enfrentarem dificuldades na descoberta de obras mais alinhadas aos seus interesses. Para lidar com esse dilema, os Sistemas de Recomendação emergem como ferramentas essenciais para filtrar, prever e sugerir conteúdo personalizado aos usuários por meio de critérios como preferências individuais, semelhança entre conteúdos e avaliações de outros leitores. Este trabalho apresenta o desenvolvimento de um sistema de recomendação chamado e-LITER, destinado a auxiliar usuários na escolha de livros, explorando técnicas de filtragem colaborativa com o uso do algoritmo Singular Value Decomposition (SVD). A eficácia do modelo final com o SVD é avaliada através de métricas como RMSE (Root Mean Square Error) e MAE (Mean Absolute Error), e comparada com o algoritmo KNN (k-Nearest Neighbors) e com uma versão inicial do SVD sem seleção e otimização de parâmetros

Palavras-chave: *sistemas de recomendação, filtragem colaborativa, algoritmo SVD.*

Universidade Presbiteriana Mackenzie

LISTA DE TABELAS

Universidade Presbiteriana Mackenzie

SUMÁRIO

1.0 INTRODUÇÃO

1.1 Motivação

Nos últimos anos, a quantidade de informação disponível online aumentou exponencialmente, transformando a maneira como interagimos com conteúdo digital. No domínio da leitura, isso não é diferente; milhões de livros estão ao alcance de um clique. Contudo, essa abundância traz consigo o desafio da escolha. Diante de tantas opções, os leitores apresentam dificuldades em encontrar livros que correspondam aos seus interesses e preferências (MANGUEL, 2013).

O aumento exponencial da disponibilidade de informação online redefiniu a interação humana com o conteúdo digital, especialmente no domínio literário. A vasta oferta de livros acessíveis com um simples clique trouxe consigo o desafio da escolha, levando os leitores a enfrentarem dificuldades na descoberta de obras mais alinhadas aos seus interesses. Para lidar com esse dilema, os Sistemas de Recomendação emergem como ferramentas cruciais, personalizando o conteúdo para os usuários por meio de critérios como preferências individuais, semelhança entre conteúdos e avaliações de outros leitores (RICCI, et al., 2011).

Educacionalmente a tecnologia desempenha um papel transformador e a integração de novas tecnologias se torna essencial para melhorar a experiência de aprendizado. Reconhecendo a literatura como uma poderosa ferramenta para o desenvolvimento cognitivo e social, a promoção de hábitos de leitura desde cedo pode impactar positivamente a vida dos estudantes. Diante do cenário desafiador evidenciado pelo desempenho baixo em leitura dos alunos brasileiros registrado no Programa Internacional de Avaliação de Estudantes 2022 e divulgado pelo INEP (2023), a necessidade de abordagens inovadoras, especialmente no contexto da pandemia, motivam a criação de um sistema de recomendação de livros personalizado.

1.2 Contexto

É neste contexto que os Sistemas de Recomendação ganham relevância, atuando como ferramentas essenciais para filtrar, prever e sugerir conteúdo personalizado aos usuários. Esse tipo de sistema visa não apenas facilitar a descoberta de novos gêneros e autores, mas também incentivar o hábito de leitura, criar comunidades de leitores e incorporar elementos de gamificação. Ao explorar essa tecnologia, as empresas podem almejar aumentar vendas, diversificar itens,

elevar a satisfação e fidelidade do usuário, e aprimorar a compreensão das necessidades dos leitores e ainda difundir o saudável hábito de uma cultura mais erudita através da leitura personalizada influenciando positivamente toda a sociedade.

Os Sistemas de Recomendação podem ser definidos como mecanismos de personalização de conteúdo, onde através de critérios como preferência do usuário, semelhança entre conteúdos acessados, avaliação de outros, etc., são feitas sugestões entre usuários (ADOMAVICIUS; TUZHILIN, 2005).

Sistemas de recomendação de livros já estão inseridos na interseção entre a tecnologia da informação, aprendizado de máquina e a promoção da leitura. Em um cenário onde a tecnologia desempenha um papel crucial na transformação de diversos setores, a educação também precisa buscar integrar inovações para melhorar a experiência de aprendizado.

1.3 Justificativa

A literatura é uma ferramenta poderosa para o desenvolvimento cognitivo e social, e a promoção de hábitos de leitura desde cedo pode impactar positivamente a vida dos estudantes. A busca por abordagens inovadoras, que consideram o contexto educacional dos países e as mudanças decorrentes da pandemia, motivam a criação de um sistema de recomendação de livros mais específico capaz de incentivar e transformar positivamente os usuários.

De acordo com o Inep (2023), o desempenho dos alunos brasileiros em leitura está muito abaixo do desejável segundo o Programa Internacional de Avaliação de Estudantes 2022, o que significa que a capacidade de interpretação e entendimento global de textos são limitados. Um sistema de recomendação de livros pode desempenhar um papel importante na facilitação e no incentivo ao hábito de leitura tanto em jovens quanto em adultos, através da descoberta de novos gêneros e autores, fornecendo sugestões relevantes e estimulantes, na criação de comunidades de leitura, na facilidade e acessibilidade e incorporando elementos de gamificação.

Segundo (RICCI, et al., 2011), existem várias razões pelas quais as empresas podem querer explorar esta tecnologia: aumentar o número de itens vendidos, vender mais itens diversificados, aumentar a satisfação e fidelidade do usuário, entender melhor o que o usuário deseja.

1.4 Objetivo

O objetivo geral desse projeto é desenvolver um Sistema de Recomendação de livros utilizando técnica de filtragem colaborativa e algoritmo Singular Value Decomposition (SVD). O sistema será chamado de “e-LITER” ”, sendo o “e-” a representação de um modelo digital e “Liter” uma abreviação de “Literatura”. Para atingir o objetivo geral temos como objetivos específicos:

- Coletar, preparar e analisar dados sobre livros, usuários e classificações;
- Dividir os dados em conjunto de treinamento e teste;
- Realizar a validação cruzada e otimização de parâmetros com GridSearchCV para avaliar o desempenho do modelo em diferentes divisões de dados usando as métricas de desempenho Root Mean Square Error (RMSE) e Mean Absolute Error (MAE);
- Treinar o modelo usando o conjunto de treinamento;
- Verificar a precisão das previsões usando o conjunto de teste e as métricas de desempenho RMSE e MAE;
- Implementação do Sistema de Recomendação.

A base de dados escolhida para o projeto foi extraída do Github em 04/03/24. Os dados são públicos, não sensíveis e foram coletados em sites de resenhas e vendas de livros. Podem ser acessados pelo link: https://github.com/zygmuntz/goodbooks-10k.

O conjunto de dados apresenta os seguintes arquivos :

- **books.csv :** informações sobre o ID do livro, ID na plataforma Goodreads, ID no site Best Reads, ID do livro em Banco de Dados central, número de versões, International Standard Book Number (ISBN), nome(s) do(s) autor(es), nome do livro original, nome do livro, código do idioma, média das avaliações, número total de avaliações, número de resenhas escritas do livro, número de avaliações de nota 1 a 5, url da imagem da capa, url da imagem da capa do livro em tamanho menor.
- **ratings.csv :** informações relacionadas ao ID do usuário, ID do livro e avaliação do livro pelo usuário.

A construção do sistema será compartilhada na plataforma colaborativa GitHub,amplamente difundida no ambiente da programação e construção de arquitetura dentro da área de informação e poderá ser acessada pelo link: https://github.com/RickPardono/Sistema-de-recomenda-o-de-livros .

2. REFERENCIAL TEÓRICO

Os Sistemas de Recomendação podem ser definidos como mecanismos de personalização de conteúdo, onde através de critérios como preferência do usuário, semelhança entre conteúdos acessados, avaliação de outros, etc., são feitas sugestões entre usuários (ADOMAVICIUS; TUZHILIN, 2005).

Considerando que as recomendações têm um papel fundamental na decisão de compra dos usuários ou até mesmo os influencia a consumir conteúdos que são semelhantes aos seus gostos, pode-se entender a dimensão dos Sistemas de Recomendação, seja no mundo digital (onde se tem aplicações que irão auxiliar os usuários a compreenderem melhor o que desejam, melhorar a experiência de navegação, e consequentemente obter resultados personalizados às suas preferências), seja em plataformas de relacionamentos entre os usuários, como por exemplo as redes sociais, que conectam pessoas com gostos similares (NOGUEIRA, 2019).

Uma forma bastante utilizada para classificação dos Sistemas de Recomendação é a divisão em 3 tipos: filtragem baseada em conteúdo, filtragem colaborativa e abordagens híbridas (SCARIOTT, 2019).

Algoritmos de recomendação baseados em conteúdo aprendem os perfis de interesse dos usuários com base nas características presentes em cada um dos itens que esses usuários avaliaram previamente. Sendo assim, eles constroem os perfis dos usuários a partir do perfil dos itens. O tipo de perfil derivado depende do método de aprendizado empregado. Estes perfis de usuários são modelos que demandam algum tempo de observação e são atualizados a medida em que novas evidências relacionadas às preferências dos usuários são observadas (SOUZA, 2011) .

A vantagem de seu uso é que ele consegue começar a sugerir itens mesmo sem muitas informações iniciais do usuário, porém, é incapaz de sugerir novos assuntos, ficando sempre preso a temas que já são de conhecimento do utilizador. Isso é chamado de overspecialization, que pode ser traduzido como Superespecialização (SCARIOTT, 2019).

No algoritmo de filtragem colaborativa, o sistema busca itens de usuários parecidos para fazer as sugestões. A expressão “filtragem colaborativa” foi criada para designar um tipo de sistema onde exista colaboração entre um grupo de interessados.

A filtragem colaborativa automatiza essencialmente o processo de recomendações "boca a boca", isto é, as informações são recomendadas a um utilizador baseado nos valores atribuídos por outras pessoas com interesses semelhantes (FERREIRA; OLIVEIRA, 2012).

Sistemas de filtragem colaborativa identificam automaticamente relações entre os usuários com base em similaridades não somente de seus perfis, mas também de seus comportamentos.

Dentre as abordagens dos sistemas de recomendações, a filtragem colaborativa tem sido amplamente utilizada e estudada nos mais variados campos e domínios de conhecimento, como por exemplo em âmbito de pesquisas acadêmicas (TAVARES; ASSUMPÇÃO, 2019).

A filtragem colaborativa surgiu com a proposta de preencher lacunas abertas na filtragem baseada em conteúdo, essa técnica se diferencia da anterior pelo fato de não ser necessário compreender ou reconhecer o conteúdo dos itens. Essa técnica tem a possibilidade de apresentar aos utilizadores, recomendações inesperadas, ou seja, recomendações de itens que o utilizador não tinha conhecimento.

Apesar de sua robustez, a filtragem colaborativa apresenta alguns desafios em seu uso, como por exemplo (NOGUEIRA, 2023) :

- Esparsidade dos dados: Em base de dados com um volume muito grande de dados, teria que ser feita à similaridade de um usuário alvo com todos os outros.
- Memória: À medida que aumenta o número de itens avaliados pelo usuário ou itens de uma coleção pessoal, mais recurso computacional será exigido para gerar recomendações.

O principal problema da filtragem colaborativa é a partida a frio ou seja o fato de que um usuário deve votar em vários itens antes de receber recomendações (ERRITALI et al.,2021)

O Sistema de Recomendação com abordagem híbrida busca somar os pontos fortes da filtragem colaborativa e da baseada em conteúdo. Ele aproveita qualidades da filtragem baseada em conteúdo, como os bons resultados para usuários incomuns e a precisão, mesmo com poucos utilizadores, pois não depende de encontrar

relações com pessoas parecidas. Além disso, agrega coisas interessantes da filtragem colaborativa como a descoberta de novos relacionamentos entre usuários e a recomendação de itens diretamente relacionada com o histórico de pontuações que o usuário já atribuiu no passado (SCARIOTT, 2019).

Surprise é uma biblioteca open-source focada em sistemas de recomendação, em específico os que fazem uso do paradigma de filtragem colaborativa. Apresenta como uma excelente ferramenta para sistemas de recomendação, pois a documentação é muito rica, detalhando de forma clara como são implementados seus algoritmos e possui ferramentas para avaliar os modelos (TAVARES; ASSUMPÇÃO, 2019).

Dentre os algoritmos presentes na Surprise temos:

- Algoritmos básicos: são algoritmos bem simples, com pouca precisão, mas que podem ser úteis na comparação de precisão. Exemplos: NormalPredictor e Baseline;
- Algoritmos inspirados no K-NN (basic nearest neighbors): algoritmos baseados na análise de vizinhança. Podem ser usados tanto para a análise baseada em usuário quanto na baseada em item. Exemplos: KNNBasic, KNNWithMeans, KNNWithZScope, KNNBaseline.
- Algoritmos baseados em fatoração de matriz: algoritmos que usam técnicas matemáticas para decompor uma matriz complexa em produtos de matrizes mais simples. Exemplos: SVD, SVD++, NMF ;
- Algoritmo Slope One: este é um algoritmo bem simples, mas com bons resultados, que calcula a diferença média nas avaliações entre pares de itens para todos os usuários que avaliaram ambos os itens;
- Algoritmo baseado em Co-Clustering: esse algoritmo busca identificar busca identificar blocos homogêneos dentro da matriz, agrupando linhas e colunas ao mesmo tempo. Isso é particularmente útil para descobrir padrões localizados que podem não ser visíveis quando se olha para as linhas ou colunas isoladamente.

Tavares e Assumpção (2019), realizaram um estudo usando o conjunto de dados Movielens, que contém 1.000.209 avaliações feitas por 6.040 usuários em 3.706 filmes para avaliar a performance dos algoritmos de filtragem colaborativa SVD, SVD++, NMF, KNNWithMeans, KNNWithZScore, KNNBaseline, Slope One e Co-Clustering. Os resultados mostram que os algoritmos baseados em fatoração de matrizes, como o SVD e o SVD++, apresentaram melhor desempenho em termos de precisão (medidos pelo MAE e RMSE) e tempo de execução. O SVD++ teve o melhor desempenho em precisão, mas com um tempo de execução significativamente maior, enquanto o SVD ofereceu um equilíbrio entre precisão e eficiência temporal. Algoritmos como o KNNWithMeans e o Slope One também foram destacados por seu bom desempenho.

Jayalakshmi et al. (2021), realizaram um estudo sobre sistema de recomendação de livros baseado em filtragem colaborativa utilizando a técnica de similaridade de cosseno. O sistema implementado como uma aplicação web, recomenda livros com base nas avaliações dos usuários. O método proposto utiliza um algoritmo de filtragem colaborativa baseado em item, construindo uma matriz de interação usuário-item e aplicando a medida de similaridade de cosseno para identificar livros similares. Por padrão, a aplicação também recomenda livros com base na popularidade, utilizando uma média ponderada das avaliações. A funcionalidade adicional permitiu aos usuários buscar livros por autor ou editora, fornecendo o nome de um livro como entrada. O desempenho do sistema de recomendação foi avaliado utilizando o métrica RMSE (Root Mean Square Error), e os resultados indicaram que o desempenho da recomendação foi melhor quando a medida de similaridade de cosseno é utilizada em comparação com a distância euclidiana.

Erritali et al. (2021) investigaram a implementação de sistemas de recomendação usando técnicas de filtragem colaborativa, especificamente através dos algoritmos K Nearest Neighbors (KNN) e Singular Value Decomposition (SVD). O objetivo principal foi melhorar a qualidade das recomendações, abordando desafios como o problema de "cold start. A pesquisa utilizou o conjunto de dados Movielens 100k, com avaliações de filmes feitas por usuários, para testar e avaliar o modelo híbrido proposto. Os resultados foram avaliados com base em métricas como Erro Quadrático Médio (RMSE), Erro Absoluto Médio (MAE), precisão e recall. O modelo SVD superou o KNN em uma abordagem de predição aleatória em todas as métricas, indicando uma precisão superior na predição de avaliações de filmes. Curiosamente, a combinação dos resultados do KNN e SVD por meio de média das estimativas de

classificação não mostrou melhorias significativas em comparação ao uso isolado do SVD.

Rana e Deeba (2019) propuseram um sistema de recomendação online de livros que utiliza Filtragem Colaborativa com Similaridade de Jaccard (JS) para fornecer recomendações mais precisas. Segundo os autores, a Filtragem Colaborativa (CF) é desafiada pela escalabilidade, esparsidade e problemas de início frio, e a abordagem proposta visa superar essas dificuldades ao aplicar a Similaridade de Jaccard. Essa medida é baseada no índice calculado para um par de livros, levando em consideração o número de usuários comuns que avaliaram ambos os livros em relação à soma de usuários que avaliaram cada um dos livros individualmente. Livros com um alto índice de JS têm prioridade nas recomendações. A pesquisa revelou que o algoritmo proposto, utilizando um conjunto de dados compactado, é mais preciso do que algoritmos existentes com conjuntos de dados completos. Além disso, ao contrário da maioria dos algoritmos que usam avaliações absolutas para medir a semelhança, a Similaridade de Jaccard considera o número de usuários comuns, o que resulta em recomendações mais precisas.

Ahmed e Letta (2023) investigaram o uso de algoritmos de filtragem colaborativa, especificamente K Nearest Neighbors (KNN) e Singular Value Decomposition (SVD), para o desenvolvimento de sistemas de recomendação de livros. Utilizando um conjunto de dados composto por 5.189 registros e 76.888 avaliações de livros coletadas do sistema de informação de estudantes da Universidade de Gondar, os pesquisadores testaram abordagens baseadas em memória e baseadas em modelo. Para a abordagem baseada em memória, foi implementada a fatoração de matrizes com melhorias de desempenho. Na abordagem baseada em modelo, os algoritmos KNN e SVD foram avaliados experimentalmente. Os resultados mostraram que o modelo baseado em SVD, otimizado e com uma pontuação RMSE de 0.1623, superou o desempenho do modelo KNN, que teve um RMSE de 1.0535, indicando que a técnica de fatoração de matrizes tem um melhor desempenho do que os modelos baseados em KNN para sistemas de recomendação colaborativa. A precisão do modelo baseado em SVD foi de 85%, enquanto a do modelo KNN foi de 53%. Os autores concluíram que a técnica de fatoração de matrizes, especificamente o algoritmo SVD, supera os recomendadores baseados em vizinhança.

Nguyen (2021), realizou um estudo sobre sistemas de recomendação utilizando filtragem colaborativa com o método de Decomposição por Valores Singulares (SVD)

e uma abordagem incremental baseada em SVD. O método Incremental SVD teve como objetivo melhorar a precisão das previsões e reduzir o tempo de execução em comparação com o SVD tradicional. Esse método foi projetado para lidar com a adição dinâmica de novos usuários ou itens, atualizando a fatoração SVD existente sem a necessidade de recomputar todo o modelo do zero. Foi empregado uma base de dados de produtos de beleza da Amazon. Esta base inclui mais de 2 milhões de avaliações de clientes abrangendo produtos de beleza vendidos desde maio de 1996 até 2018. Para os experimentos, foram considerados apenas os usuários que classificaram 10 ou mais produtos, convertendo os dados em uma matriz usuário-item, onde as linhas representam usuários e as colunas representam itens. A pesquisa revelou melhorias tanto na acurácia das previsões quanto no tempo de resposta do sistema de recomendação com o método Incremental SVD, abrindo caminho para aprimoramentos significativos em sistemas de recomendação em plataformas de e-commerce, como a Amazon, ao lidar eficientemente com grandes conjuntos de dados e fornecer recomendações personalizadas mais rápidas e precisas para os usuários.

Koren et al. (2009), exploraram como a fatoração de matrizes mapeia itens e usuários em um espaço fatorial conjunto, permitindo que as interações entre usuários e itens sejam modeladas eficientemente. A abordagem destaca-se pela capacidade de contornar desafios associados à esparsidade de dados e ao problema do início frio, por meio da minimização do erro quadrado regularizado apenas nas avaliações conhecidas, evitando o overfitting através de um modelo regularizado. A eficácia das técnicas é exemplificada no contexto do Netflix Prize, demonstrando a superioridade desses modelos em prever avaliações de filmes com precisão. Os autores afirmaram que as técnicas de fatoração de matrizes emergem como metodologias dominantes em sistemas de recomendação, superando técnicas clássicas de vizinhança mais próxima por sua capacidade de incorporar informações adicionais como feedback implícito, efeitos temporais e níveis de confiança. A pesquisa conclui que a fatoração de matrizes não só oferece previsões precisas e um modelo compacto eficiente em termos de memória, mas também flexibilidade para modelar diversas situações reais, reforçando sua importância no avanço dos sistemas de recomendação.

Zhou et al. (2012) afirmam que a aplicação de métodos de redução da dimensionalidade de dados, especificamente o SVD, são uma das soluções mais populares para os problemas de esparsidade. SVD é um algoritmo de fatoração de matrizes que pode extrair características dos recursos do conjunto de dados, dividindo a matriz original de classificações de usuário-itens em três multiplicações de matrizes

menores. Dada uma matriz mx A (N é o número de itens, M é o número de usuários) com classificação(A) = r, o $SV\ D(A)$ é definido como:

$$SV\ D(A) = U \times S \times V^{T}$$

onde U, S, V são dimensões $m \times r$, $r \times r$, $r \times n$. Enquanto a matriz do meio S é uma matriz diagonal com r elementos diferentes de zero, que são os valores singulares de A, as matrizes U e V são ortogonais [21]. U e V são conhecidos como vetores singulares da esquerda e da direita, respectivamente com as primeiras r colunas de U correspondendo aos valores singular diferentes de zero que abrangem o espaço das colunas, e as primeiras r colunas de V abrangem o espaço das linhas da matriz A.

A acuracidade do sistema de recomendação SVD é avaliada através de duas medidas populares: Root Mean Square Error (RMSE) e Mean Absolute Error (MAE). Quanto mais baixo o valor de ambas as métricas, melhor o desempenho dos algoritmos de recomendação. O SVD diminui a dimensão da matriz de utilidade e é a melhor aproximação linear da matriz original A usando três matrizes de multiplicação (SARWAR et al., 2002).

Medidas populares de acurácia estatística chamadas Root Mean Square Error (RMSE) e Mean Absolute Error (MAE) mostram a proximidade da classificação prevista com as classificações verdadeiras. Os menores valores de ambas as métricas correspondem a maior precisão do sistema recomendador (NGUYEN, 2017). Se rui representa a verdadeira classificação do item i pelo usuário u e $\hat{r}ui$ mostra a classificação prevista no item i pelo usuário u, RSME e MAE de N pares de previsão de classificação correspondentes são definidos:

$$RSME = \sqrt{\frac{\Sigma_{i=1}^{N}(r_{ui} - \hat{r}_{ui})^2}{N}}$$

$$MAE = \frac{\Sigma_{i=1}^{N}|r_{ui} - \hat{r}_{ui}|}{N}$$

O objetivo desse projeto é desenvolver um Sistema de Recomendação de livros chamado “e-LITER”, utilizando técnica de Filtragem Colaborativa e algoritmo Singular Value Decomposition (SVD).

3.0 METODOLOGIA

Neste capítulo serão descritas todas as etapas para a elaboração deste projeto.

3.1 REVISÃO TEÓRICA SOBRE SISTEMAS DE RECOMENDAÇÃO

Inicialmente foi realizada uma revisão teórica sobre Sistemas de Recomendação, explorando conceitos fundamentais, tipos de sistemas, algoritmos populares e desafios enfrentados. A revisão analisou diferentes abordagens de Sistemas de Recomendação como Filtragem Baseada em Conteúdo, Filtragem Colaborativa, Abordagens Híbridas, e os principais algoritmos utilizados em Sistemas de Recomendação como SVD, KNN e outros.

A revisão norteou a escolha pelos integrantes do projeto em optar pelo Sistema de Recomendação com Filtragem Colaborativa e algoritmo SVD. O Singular Value Decomposition (SVD) apresenta características únicas e vantagens quando comparada com outras metodologias como filtragem baseada em conteúdo ou abordagens híbridas. Aqui estão algumas razões detalhadas para essa escolha:

- **Descoberta de Interesses Latentes:** o SVD é eficaz em identificar fatores latentes ou características ocultas nos dados que representam padrões de interesses entre os usuários, que podem não ser diretamente observáveis. Isso permite que o sistema capture nuances complexas nas preferências dos usuários, superando muitas vezes o que é possível através da filtragem baseada em conteúdo, que se limita a características explícitas dos itens;
- **Independência do Conteúdo dos Itens:** enquanto a filtragem baseada em conteúdo requer metadados detalhados do item (como gênero, autor, temas), a filtragem colaborativa pode funcionar efetivamente sem qualquer informação sobre o conteúdo do item. Isso é particularmente útil em cenários onde tais metadados não estão disponíveis ou são difíceis de quantificar.
- **Adaptabilidade a Diferentes Usuários e Itens:** a filtragem colaborativa, especialmente via SVD, adapta-se bem à diversidade de usuários e itens, pois aprende com as interações entre usuários e itens, ajustando continuamente as recomendações com base nos novos dados recebidos.
- **Superação de Problemas Comuns em Filtragem Colaborativa:** o SVD pode mitigar o problema de esparsidade de dados ao reduzir a dimensionalidade dos dados de entrada, projetando-os em um espaço de características latentes onde os padrões são mais densos e significativos.Técnicas baseadas em SVD podem ser escaladas eficientemente usando algoritmos otimizados para

decomposição de matrizes, permitindo que eles sejam aplicados em grandes conjuntos de dados.

Como limitações para o SVD podemos citar:

1. **Cold Start de Usuário**: o SVD depende de dados históricos de avaliações para determinar os fatores latentes que descrevem as preferências dos usuários. Sem essas avaliações, o sistema não tem base para inferir as preferências do novo usuário, tornando difícil oferecer recomendações personalizadas e precisas inicialmente.
2. **Cold Start de Item:** o SVD requer dados de interação para identificar as características latentes de novos itens. Sem avaliações, os novos itens não podem ser adequadamente representados no espaço latente, o que impede que o sistema recomende esses itens aos usuários de forma eficaz até que acumulem avaliações suficientes.

3.2 BIBLIOTECAS IMPORTADAS

Para o desenvolvimento do projeto em Python foram importadas as seguintes bibliotecas segundo código abaixo:

```
import pandas as pd
import numpy as np
from surprise import Dataset, Reader, SVD, accuracy
from surprise.model_selection import train_test_split,
cross_validate, GridSearchCV
from surprise.accuracy import rmse
from surprise.accuracy import mae
import difflib
import matplotlib.pyplot as plt
import seaborn as sns
```

A biblioteca **Pandas** foi usada para carregamento, manipulação e análise de dados. É fundamental também para realizar operações de pré-processamento como tratamento de valores nulos e duplicados.

NumPy foi usada para dar suporte a arrays e matrizes multidimensionais e uma gama de operações matemáticas. O NumPy geralmente suporta operações com dados dentro de Pandas e outras bibliotecas.

Surprise é uma biblioteca específica para construir e analisar sistemas de recomendação que oferece suporte a vários algoritmos de recomendação, incluindo SVD, e ferramentas para validação cruzada e otimização de hiperparâmetros.

Difflib foi utilizada para encontrar correspondências entre sequências como strings. No projeto especificamente, foi usada para encontrar o livro mais próximo do título fornecido, o que é útil em funções de recomendação para identificar e buscar informações de livros.

Matplotlib e **Seaborn** são bibliotecas de plotagem usadas para gerar diversos tipos de gráficos. Foram utilizadas para visualizar distribuições e outros dados estatísticos durante a análise exploratória.

3.3 COLETA DOS DADOS

Os datasets books.csv e ratings.csv (disponíveis em https://github.com/zygmuntz/goodbooks-10k), foram extraídos do Github do usuário Zygmunt Zając e importados para a estrutura de um DataFrame Pandas:

```
ratings_data = pd.read_csv('/content/ratings.csv')
books_metadata = pd.read_csv('/content/books.csv')
```

3.4 PREPARAÇÃO DOS DADOS E ANÁLISE EXPLORATÓRIA

A preparação e análise exploratória de dados são etapas cruciais em qualquer projeto de sistema de recomendação, incluindo este projeto de recomendação de livros, pois garantem que o modelo seja construído e treinado em um conjunto de dados limpo e bem compreendido. Essas etapas envolvem a verificação de completude e integridade dos dados, a verificação de dados irrelevantes, duplicados ou nulos que podem impactar no modelo. Essas etapas fornecem insights fundamentais sobre as características e padrões nos dados, o que é essencial para tomar decisões informadas durante o desenvolvimento do modelo. Ao visualizar distribuições de avaliações, verificar a popularidade dos livros, e entender o comportamento dos usuários, os desenvolvedores podem identificar tendências, anomalias, e a estrutura geral dos dados, o que ajuda a otimizar a precisão das recomendações.

Essas etapas são descritas a seguir:

Visualização das 5 primeiras linhas do DataFrame “ratings_data”:

```
ratings_data.head()
```

user_id	book_id	rating
1	258	5
2	4081	4
2	260	5
2	9296	5
2	2318	3

Verificação do número de linhas e colunas:

```
ratings_data.shape
```

(5976479, 3)

Observamos um DataFrame extenso, com quase 6 milhões de linhas e três colunas.

Verificação de valores nulos:

```
ratings_data.isnull().sum()
```

user_id 0
book_id 0
rating 0
dtype: int64

Não há valores nulos.

Verificação de valores duplicados:

```
ratings_data.duplicated().sum()
```

0

Não há valores duplicados.

Verificação do tipo de dados e uso da memória:

```
ratings_data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5976479 entries, 0 to 5976478
Data columns (total 3 columns):
 #   Column   Dtype
---  ------   -----
 0   user_id  int64
 1   book_id  int64
 2   rating   int64
dtypes: int64(3)
memory usage: 136.8 MB
```

Visualização das 5 primeiras linhas do DataFrame “books_metadata”:

```
books_metadata.head()
```

```
[ ]  1 books_metadata.head()
```

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	...	ratings_count	work_ratings_count
0	1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008.0	The Hunger Games	...	4780653	4942365
1	2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	Harry Potter and the Philosopher's Stone	...	4602479	4800065
2	3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005.0	Twilight	...	3866839	3916824
3	4	2657	2657	3275794	487	61120081	9.780061e+12	Harper Lee	1960.0	To Kill a Mockingbird	...	3198671	3340896
4	5	4671	4671	245494	1356	743273567	9.780743e+12	F. Scott Fitzgerald	1925.0	The Great Gatsby	...	2683664	2773745

5 rows × 23 columns

Verificação do número de linhas e colunas do DataFrame “books_metadata”:

```
books_metadata.shape
```

(10000, 23)

Verificação de valores nulos:

```
books_metadata.isnull().sum()
```

book_id 0
goodreads_book_id 0
best_book_id 0
work_id 0
books_count 0

```
isbn                   700
isbn13                 585
authors                  0
original_publication_year    21
original_title         585
title                    0
language_code         1084
average_rating           0
ratings_count            0
work_ratings_count       0
work_text_reviews_count      0
ratings_1                0
ratings_2                0
ratings_3                0
ratings_4                0
ratings_5                0
image_url                0
small_image_url          0
dtype: int64
```

Os atributos com valores nulos de"books_metadata" não serão usados na construção do modelo e não necessitarão portanto de tratamento.

Verificação de valores duplicados:

```
books_metadata.duplicated().sum()
0
```

Não há valores duplicados.

Verificação do tipo de dados e uso da memória:

```
books_metadata.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   book_id                    10000 non-null  int64  
 1   goodreads_book_id          10000 non-null  int64  
 2   best_book_id               10000 non-null  int64  
 3   work_id                    10000 non-null  int64  
 4   books_count                10000 non-null  int64  
 5   isbn                       9300 non-null   object 
 6   isbn13                     9415 non-null   float64
 7   authors                    10000 non-null  object 
 8   original_publication_year  9979 non-null   float64
 9   original_title             9415 non-null   object 
```

```
 10  title                       10000 non-null  object
 11  language_code               8916 non-null   object
 12  average_rating              10000 non-null  float64
 13  ratings_count               10000 non-null  int64
 14  work_ratings_count          10000 non-null  int64
 15  work_text_reviews_count     10000 non-null  int64
 16  ratings_1                   10000 non-null  int64
 17  ratings_2                   10000 non-null  int64
 18  ratings_3                   10000 non-null  int64
 19  ratings_4                   10000 non-null  int64
 20  ratings_5                   10000 non-null  int64
 21  image_url                   10000 non-null  object
 22  small_image_url             10000 non-null  object
dtypes: float64(3), int64(13), object(7)
memory usage: 1.8+ MB
```

Verificar quantidade total de usuários:

```
ratings_data['user_id'].nunique()
```

```
53424
```

Verificar quantidade total de livros avaliados:

```
ratings_data['book_id'].nunique()
```

```
10000
```

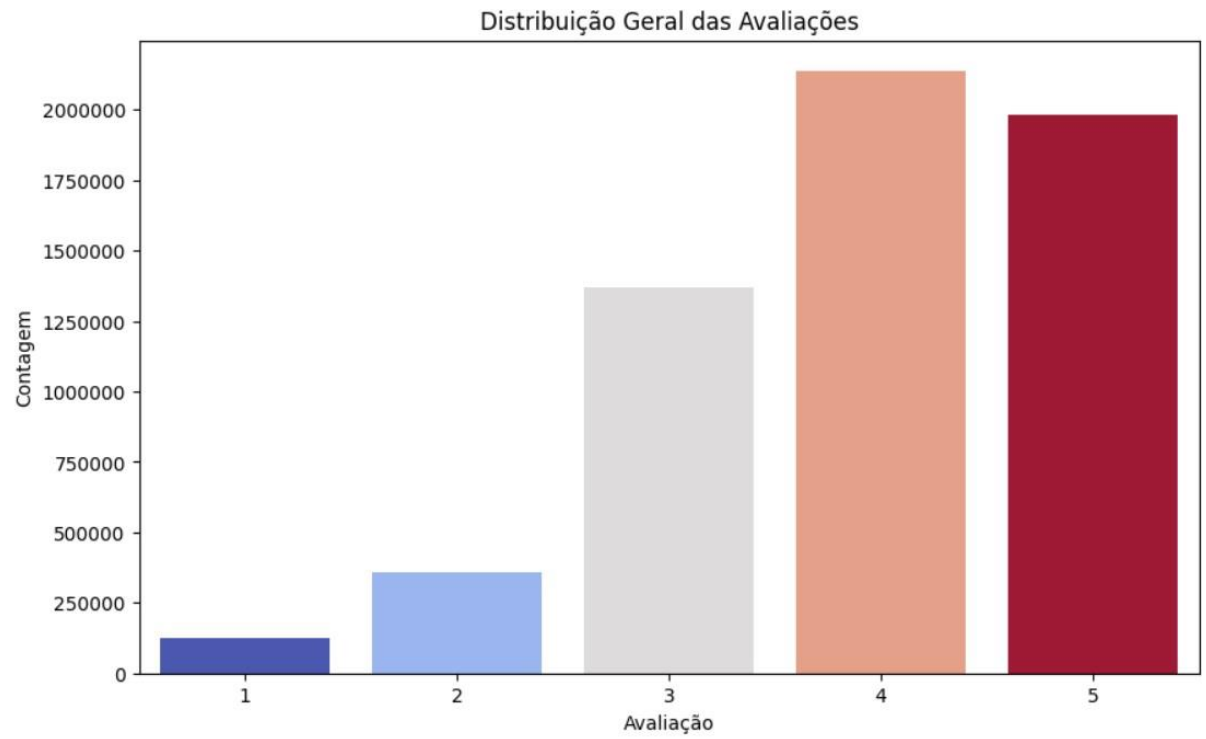
Verificar quantidade total de livros disponíveis:

```
books_metadata['book_id'].nunique()
```

```
10000
```

Criar gráfico de barras para avaliar distribuição das avaliações:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='rating', hue='rating', data=ratings_data,
palette='coolwarm', legend=False)
plt.title('Distribuição Geral das Avaliações')
plt.xlabel('Avaliação')
plt.ylabel('Contagem')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



Observa-se no gráfico que a quantidade de avaliações de livros com nota 1 e 2 são bem menores, que os livros avaliados com notas 3, 4 e 5. A barra com maior quantidade de avaliações representa as avalições com nota 4 (mais de 2 milhões), seguida pela barra da quantidade de avaliações com nota 5, e a barra com notas 3.

Contagem de avaliações por livro:

```
book_counts = ratings_data['book_id'].value_counts()
```

Imprimir estatísticas descritivas sobre as contagens de avaliações por livro:

```
print(book_counts.describe().round(2))
```

```
count    10000.00
mean       597.65
std       1267.29
min          8.00
25%        155.00
50%        248.00
75%        503.00
max      22806.00
Name: count, dtype: float64
```

O livro com a menor quantidade de avaliações tem 8 avaliações e o com a maior quantidade tem 22806 avaliações. A média de avaliações por livro é de aproximadamente 598 avaliações, o que é ótimo para o sistema de recomendação.

Contagem de avaliações por usuário:

```
user_counts = ratings_data['user_id'].value_counts()
```

Imprimir estatísticas descritivas sobre as contagens de avaliações por usuário:

```
print(user_counts.describe().round(2))
```

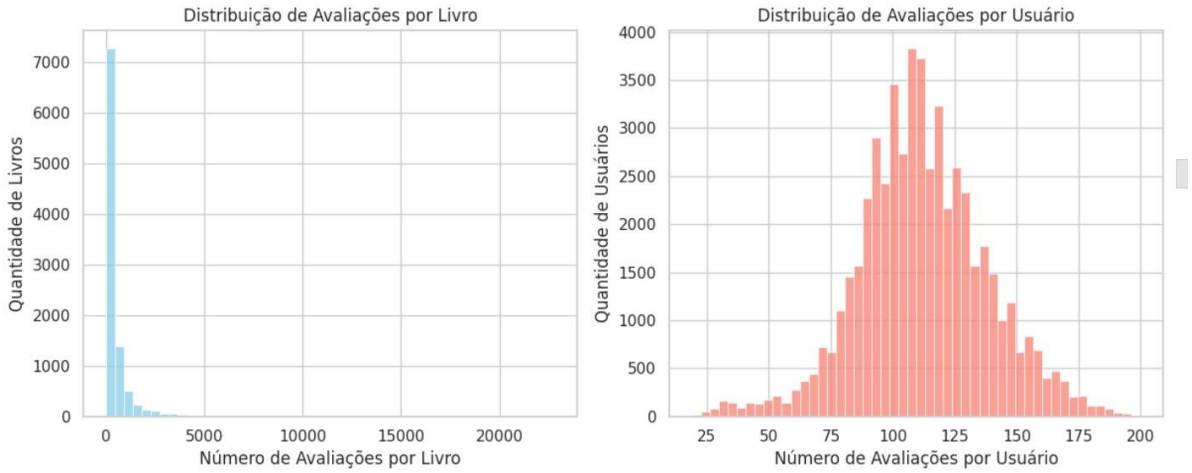
```
count    53424.00
mean       111.87
std         26.07
min         19.00
25%         96.00
50%        111.00
75%        128.00
max        200.00
Name: count, dtype: float64
```

O usuário com a menor quantidade de avaliações tem 19 avaliações e o com a maior quantidade tem 200 avaliações. A média de avaliações por usuário é de aproximadamente 112 avaliações, o que é ótimo para o sistema de recomendação. Um total de 53424 usuários fizeram avaliações de livros.

Criar dois histogramas para visualizar a distribuição de avaliações por livro e a distribuição de avaliações por usuário :

```
sns.set(style="whitegrid")
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
sns.histplot(book_counts, bins=50, color='skyblue', ax=ax[0],
kde=False)
ax[0].set_title('Distribuição de Avaliações por Livro')
ax[0].set_xlabel('Número de Avaliações por Livro')
ax[0].set_ylabel('Quantidade de Livros')
sns.histplot(user_counts, bins=50, color='salmon', ax=ax[1], kde=False)
ax[1].set_title('Distribuição de Avaliações por Usuário')
ax[1].set_xlabel('Número de Avaliações por Usuário')
ax[1].set_ylabel('Quantidade de Usuários')
plt.tight_layout()
```

```
plt.show()
```

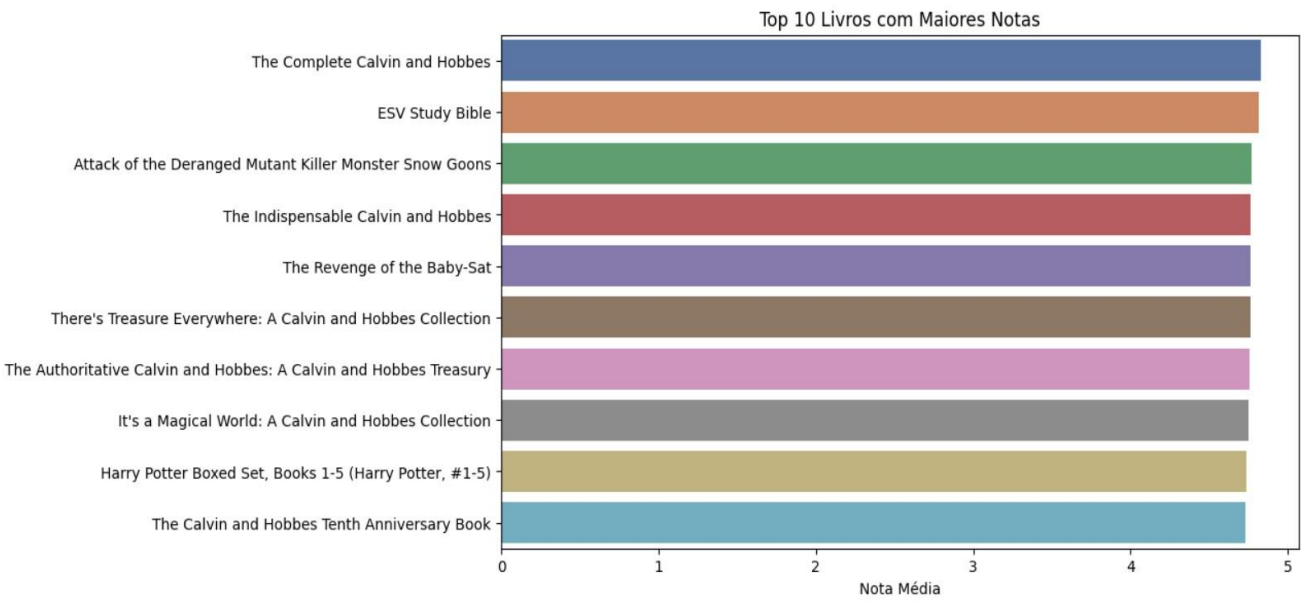


No histograma da esquerda vemos uma distribuição fortemente assimétrica à direita, com uma quantidade significativa de livros (mais de 75%) recebendo 503 ou menos avaliações. Há um pequeno número de livros (Outliers) com muitas avaliações, estendendo a cauda do histograma para a direita. Portanto há uma quantidade suficiente de interações do usuário para a maioria dos itens no sistema, o que é fundamental para qualquer sistema de recomendação baseado em filtragem colaborativa.
No histograma da direita vemos uma distribuição das avaliações por usuário simétrica, indicando que a maioria dos usuários avaliou um número de livros em torno da média (112 avaliações). Não parece haver outliers significativos. Todos os usuários fizeram pelo menos 19 avaliações, e nenhum fez mais do que 200. A distribuição indica que há uma quantidade suficiente de dados de avaliação por usuário, o que é bom para técnicas de filtragem colaborativa, já que o modelo pode aprender melhor com uma maior consistência nas avaliações dos usuários.

Criar gráfico de barras para visualizar os 10 livros com maiores notas:

```
book_avg_ratings = ratings_data.groupby('book_id')['rating'].mean()
# Ordenar os livros pela nota média e selecionar os top 10
top_10_books_ids = book_avg_ratings.nlargest(10).index
# Juntar com os metadados dos livros para obter os títulos dos top 10 livros.
top_10_books_info= books_metadata[books_metadata['book_id'].isin(top_10_books_ids)]
```

```
top_10_books_info = top_10_books_info.set_index('book_id').join(book_avg_ratings,
on='book_id')
top_10_books_info = top_10_books_info.nlargest(10, 'rating')
plt.figure(figsize=(10, 6))
sns.barplot(x='rating', y='title',
data=top_10_books_info,hue='title', palette='deep')
plt.title('Top 10 Livros com Maiores Notas')
plt.xlabel('Nota Média')
plt.ylabel('')
plt.show()
```

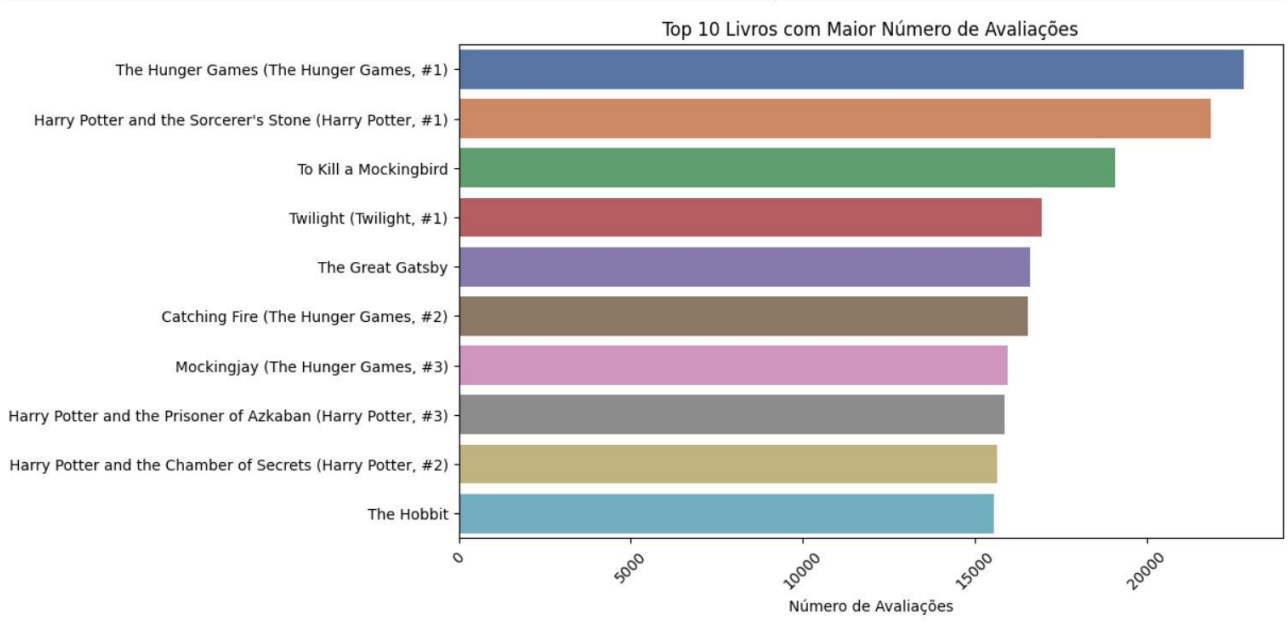


Criar gráfico de barras para visualizar os 10 livros com maior número de avaliações:

```
# Contar o número de avaliações por livro
book_ratings_counts = ratings_data.groupby('book_id')['rating'].count()
# Ordenar os livros pelo número de avaliações e selecionar os
top 10
top_10_books_ids = book_ratings_counts.nlargest(10).index
# Obter os títulos dos top 10 livros
top_10_books_titles =  books_metadata[books_metadata['book_id'].isin
(top_10_books_ids)]
# Juntar os títulos com os contagens de avaliações
```

```
top_10_books = top_10_books_titles.set_index('book_id').join
(book_ratings_counts, on='book_id')
top_10_books = top_10_books.nlargest(10, 'rating')
```

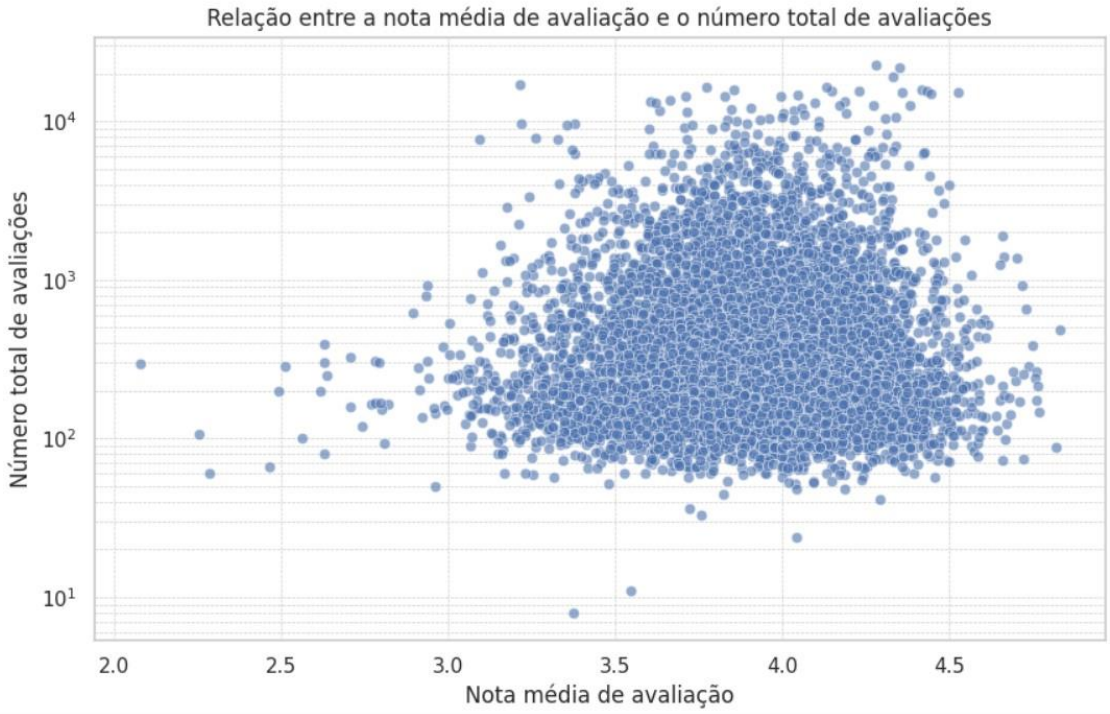
```
# Plotar os resultados
plt.figure(figsize=(10, 6))
sns.barplot(x='rating', y='title', data=top_10_books,
hue='title', palette='deep')
plt.title('Top 10 Livros com Maior Número de Avaliações')
plt.xlabel('Número de Avaliações')
plt.ylabel('')
plt.xticks(rotation=45)
plt.show()
```



Criar gráfico de dispersão para visualizar relação entre nota de avaliação e número total de avaliações:

```
# Calculando a média de avaliação para cada livro
average_ratings = ratings_data.groupby('book_id')['rating'].mean()
# Calculando o número total de avaliações para cada livro
ratings_count = ratings_data.groupby('book_id')['rating'].count()
```

```
# Criando um DataFrame com as informações calculadas
books_info = pd.DataFrame({'Average Rating': average_ratings,
'Number of Ratings': ratings_count})
# Criando o scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=books_info, x='Average Rating', y='Number
of Ratings', alpha=0.6)
plt.title('Relação entre a nota média de avaliação e o número
total de avaliações')
plt.xlabel('Nota média de avaliação')
plt.ylabel('Número total de avaliações')
plt.xscale('linear')
plt.yscale('log')
plt.grid(True, which="both", ls="--", linewidth=0.5)
plt.show()
```



Verifica-se no gráfico de dispersão que há uma concentração densa de pontos em torno do meio da escala de notas, particularmente entre 3,5 e 4,5. Isto indica que a maioria dos livros recebe notas dentro dessa faixa e, também possuem uma

quantidade moderada de avaliações. A tendência de pontos indica que livros com um número maior de avaliações tendem a ter notas médias melhores.

3.5 CRIAÇÃO DO OBJETO READER E CARREGAMENTO DOS DADOS DE AVALIAÇÕES

```
reader = Reader(rating_scale=(1,5))
data = Dataset.load_from_df(ratings_data[['book_id','user_id',
'rating']],reader)
```

O objeto Reader é uma parte essencial da biblioteca Surprise, que serve para interpretar os dados de entrada de avaliações. O parâmetro rating_scale=(1,5) informa ao Reader que as avaliações no conjunto de dados variam de 1 a 5. Esse parâmetro ajuda a biblioteca a entender corretamente a escala dos dados de avaliação, garantindo que os algoritmos de recomendação processarão os dados dentro do intervalo esperado. “Dataset.load_from_df()” é um método da Surprise que carrega os dados de um DataFrame do Pandas para uma estrutura que pode ser usada diretamente pelos algoritmos de recomendação da biblioteca.

3.6 DIVISÃO DOS DADOS EM CONJUNTO DE TREINAMENTO E TESTE

```
trainset, testset = train_test_split(data, test_size=0.25)
```

Foi utilizado a função train_test_split para dividir o conjunto de dados em um conjunto de treinamento (75%) e um conjunto de teste (25%). Essa divisão é essencial para poder treinar o modelo em um subconjunto de dados e validar seu desempenho em um conjunto separado, não visto durante o treinamento.

3.7 CONFIGURAÇÃO, VALIDAÇÃO CRUZADA E TREINAMENTO DO MODELO

```
svd = SVD(verbose=True, n_epochs=20)
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=3,
verbose=True)
```

```
Processing epoch 19
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).
Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)   0.8410  0.8400  0.8408  0.8406  0.0004
```

```
MAE (testset)    0.6507  0.6499  0.6507  0.6504  0.0004
Fit time         145.73  152.83  145.59  148.05  3.38
Test time        45.35   46.75   36.95   43.02   4.33
{'test_rmse': array([0.84095074, 0.84000788, 0.84083453]),
 'test_mae': array([0.65066692, 0.64991797, 0.65070177]),
 'fit_time': (145.73221015930176, 152.82777094841003, 145.5862340927124),
 'test_time': (45.34975457191467, 46.748693227767944, 36.9525671005249)}
```

```
svd.fit(trainset)
```

O modelo SVD é configurado com os parâmetros “Default”. Em seguida foi utilizado cross_validate para avaliar o desempenho do modelo usando validação cruzada em três folds, medindo as métricas RMSE e MAE. O cross_validate ajuda a garantir que o desempenho do modelo é estável e consistente em diferentes subconjuntos dos dados, reduzindo o risco de sobreajuste (overfitting) e fornecendo uma medida mais confiável de como o modelo irá se comportar em dados não vistos. Isso forneceu uma estimativa robusta do desempenho do modelo. Logo após, o modelo é treinado no conjunto de treinamento completo.

3.8 AVALIAÇÃO DO DESEMPENHO DO MODELO

```
predictions_svd = svd.test(testset)
rmse = accuracy.rmse(predictions_svd)
mae = accuracy.mae(predictions_svd)
```

```
RMSE: 0.8344
MAE:  0.6443
```

Após o treinamento, o modelo é testado no conjunto de teste para calcular o RMSE e MAE final, que são medidas de quão bem o modelo prevê as avaliações reais.

O RMSE mede a magnitude média dos erros entre os valores previstos pelo modelo e os valores reais. É calculado como a raiz quadrada da média dos quadrados dos erros. Quanto menor o valor do RMSE, melhor, pois indica que os erros entre as previsões e os valores reais são menores. O RMSE é sensível a outliers, pois os erros são elevados ao quadrado antes de serem médios, o que dá um peso maior aos erros grandes.

MAE mede a média dos erros absolutos entre as previsões e os valores reais, fornecendo uma visão clara da magnitude média dos erros sem considerar sua direção. Assim como o RMSE, menores valores de MAE indicam um melhor

desempenho do modelo, mas o MAE é menos sensível a outliers, pois não eleva os erros ao quadrado.

Na literatura sobre sistemas de recomendação, especialmente aqueles aplicados ao domínio de livros e entretenimento, um RMSE na faixa de 0.8 a 1.2 é frequentemente relatado como dentro do intervalo aceitável. O valor de RMSE de 0.8344 está na extremidade inferior dessa faixa, indicando um bom desempenho.

Da mesma forma, um MAE de 0.6443 é considerado bom, especialmente quando comparado com benchmarks de sistemas semelhantes.

3.9 IMPLEMENTAÇÃO DO MODELO E GERAÇÃO DE RECOMENDAÇÕES

O sistema é implementado través da construção de funções auxiliares:

- get_book_id: identifica o ID de um livro com base no título aproximado, usando correspondência de strings.
- get_book_info: retorna informações detalhadas sobre um livro com base em seu ID.
- predict_review: estima a avaliação que um usuário daria a um livro.
- generate_recommendation: gera recomendações para um usuário específico. Ela faz previsões de avaliação para cada livro no catálogo e recomenda aqueles com uma avaliação prevista acima de um limiar especificado (neste caso, 4). As recomendações são limitadas a um número específico de livros (três neste exemplo de recomendação para o usuário de ID de número 1000).

```
def get_book_id(book_title, metadata):
    existing_titles = list(metadata['title'].values)
    closest_titles = difflib.get_close_matches(book_title,
    existing_titles)
    book_id = metadata[metadata['title'] ==
    closest_titles[0]]['book_id'].values[0]
    return book_id
def get_book_info(book_id, metadata):
     book_info = metadata[metadata['book_id'] ==
     book_id][['book_id', 'isbn','authors', 'title',
     'original_title']]
     return book_info.to_dict(orient='records')
def predict_review(user_id, book_title, model, metadata):
```

```
    book_id = get_book_id(book_title, metadata)
    review_prediction = model.predict(uid=user_id,
    iid=book_id)
    return review_prediction.est
def generate_recommendation(user_id, model, metadata,
thresh=4, num_recommendations=3):
    book_titles = list(metadata['title'].values)
    recommendations = []
    for book_title in book_titles:
        if len(recommendations) >= num_recommendations:
            break
        rating = predict_review(user_id, book_title, model,
        metadata)
        if rating >= thresh:
           book_id = get_book_id(book_title, metadata)
          book_info = get_book_info(book_id, metadata)
           recommendations.append(book_info[0])
        return recommendations
```

Geração de recomendações para o usuário de ID 1000:

```
generate_recommendation(1000, svd, books_metadata)
```

```
[{'book_id': 2,
  'isbn': '439554934',
  'authors': 'J.K. Rowling, Mary GrandPré',
  'title': "Harry Potter and the Sorcerer's Stone (Harry Potter, #1)",
  'original_title': "Harry Potter and the Philosopher's Stone"},
 {'book_id': 5,
  'isbn': '743273567',
  'authors': 'F. Scott Fitzgerald',
  'title': 'The Great Gatsby',
  'original_title': 'The Great Gatsby'},
 {'book_id': 6,
  'isbn': '525478817',
  'authors': 'John Green',
  'title': 'The Fault in Our Stars',
  'original_title': 'The Fault in Our Stars'}]
```

3.10 COMPARAÇÃO COM O KNN

Um segundo modelo preditivo foi construído com o algoritmo KNN (k-Nearest Neighbors) para comparação do seu desempenho com o desempenho do Sistema de Recomendação que usou o algoritmo SVD. O KNN é usado para prever as preferências ou interesses de um usuário com base nas preferências de outros usuários que são semelhantes a ele (vizinhos mais próximos). Ao analisar um grupo de usuários que deram avaliações semelhantes para itens semelhantes, o KNN faz previsões sobre como um usuário poderia avaliar ou preferir novos itens. As métricas de avaliação foram as mesmas utilizadas no SVD (RMSE e MAE), e os passos para divisão dos dados em conjuntos de treinamento e teste e a validação cruzada com 3 Folds foram igualmente implementados.

Segue o código em Python do modelo com KNN a seguir:

```
import pandas as pd
import numpy as np
from surprise import Dataset, Reader, KNNBasic, accuracy
from surprise.model_selection import train_test_split,
cross_validate
from surprise.accuracy import rmse
from surprise.accuracy import mae
import difflib
import matplotlib.pyplot as plt
import seaborn as sns

reader = Reader(rating_scale=(1,5))
data = Dataset.load_from_df(ratings_data[['book_id','user_id'
,'rating']], reader)

trainset, testset = train_test_split(data, test_size=0.25)

knn = KNNBasic(sim_options={'name': 'cosine', 'user_based':
True})

cross_validate(knn, data, measures=['RMSE', 'MAE'], cv=3,
verbose=True)

knn.fit(trainset)
```

```
predictions_knn = knn.test(testset)
rmse_knn = accuracy.rmse(predictions_knn)
mae_knn = accuracy.mae(predictions_knn)
```

```
RMSE: 0.9619
MAE:  0.7584
```

```
generate_recommendation(1000, knn, books_metadata)
```

```
[{'book_id': 1,
  'isbn': '439023483',
  'authors': 'Suzanne Collins',
  'title': 'The Hunger Games (The Hunger Games, #1)',
  'original_title': 'The Hunger Games'},
 {'book_id': 2,
  'isbn': '439554934',
  'authors': 'J.K. Rowling, Mary GrandPré',
  'title': "Harry Potter and the Sorcerer's Stone (Harry Potter, #1)",
  'original_title': "Harry Potter and the Philosopher's Stone"},
 {'book_id': 3,
  'isbn': '316015849',
  'authors': 'Stephenie Meyer',
  'title': 'Twilight (Twilight, #1)',
  'original_title': 'Twilight'}]
```

Ao comparar os resultados do algoritmo SVD com os do algoritmo KNN, observamos uma diferença significativa no desempenho em relação ao RMSE e MAE. O SVD apresentou RMSE e MAE com os respectivos valores em 0.8344 e 0.6443, ou seja, valores menores em comparação com o KNN (RMSE: 0.9619 e MAE: 0.7584), o que indica uma melhor capacidade de predição das avaliações de filmes. Isso significa que as predições feitas pelo SVD estão mais próximas das avaliações reais dos usuários do que as predições feitas pelo KNN.

Uma possível explicação para essa diferença de desempenho é a natureza dos dois algoritmos. O SVD é uma técnica baseada em fatorização de matriz que modela as interações usuário-item através de uma representação de baixa dimensionalidade, buscando capturar padrões latentes nos dados. Por outro lado, o KNN é um algoritmo de filtragem colaborativa baseado em vizinhos mais próximos, que utiliza a similaridade entre os usuários ou itens para fazer recomendações.

Assim como a pesquisa de Erritali et al. (2021) sobre a implementação de sistemas de recomendação utilizando tanto KNN quanto SVD, os resultados encontrados corroboram a superioridade do SVD em relação ao KNN em termos de RMSE e MAE.

Isso está alinhado com a descoberta de que o SVD superou o KNN em métricas de precisão na predição de avaliações de filmes. Essa diferença de desempenho pode ser atribuída à capacidade do SVD de capturar padrões latentes nos dados de forma mais eficaz do que o KNN, especialmente em conjuntos de dados com alta dimensionalidade e esparsidade.

3.11 AJUSTES DO PIPELINE DE TREINAMENTO

Uma significativa melhoria no treinamento do modelo SVD foi realizada com a implementação do GridSearchCV, para otimizar os parâmetros do modelo SVD. Isso incluiu ajustar o número de fatores, o número de épocas, a taxa de aprendizagem e o termo de regularização.

Uma instância do GridSearchCV foi criada usando o modelo SVD, o dicionário de parâmetros (param_grid), medidas de desempenho (RMSE e MAE), validação cruzada de 3 folds (cv=3), e execução paralela (n_jobs=-1 para usar todos os processadores disponíveis).

O método fit foi chamado para executar o GridSearchCV no conjunto de dados fornecido. Isso envolve treinar o modelo SVD com todas as combinações possíveis de parâmetros especificados e validar cada combinação usando a validação cruzada para identificar os parâmetros que produzem o menor erro de previsão.

Após a conclusão do GridSearchCV, os melhores parâmetros para cada métrica de desempenho foram impressos:

- Para RMSE, os melhores parâmetros foram {'n_factors': 100, 'n_epochs': 30, 'lr_all': 0.01, 'reg_all': 0.1}.
- Para MAE, os melhores parâmetros foram {'n_factors': 100, 'n_epochs': 30, 'lr_all': 0.005, 'reg_all': 0.02}.

Um novo modelo SVD foi configurado usando uma combinação equilibrada das melhores configurações de ‘lr_all’ e ‘reg_all’ derivadas das otimizações para RMSE e MAE, resultando em lr_all=0.0075 e reg_all=0.06.

O modelo foi treinado usando o conjunto de treinamento completo.

O modelo otimizado foi avaliado no conjunto de teste, e os resultados finais para RMSE e MAE foram calculados:

- RMSE: 0.8159, uma melhoria em relação aos resultados antes da otimização, indicando que as previsões do modelo estão mais precisas.

- MAE: 0.6323, também uma melhoria, mostrando que, em média, o erro absoluto das previsões é bastante baixo.

4. RESULTADOS

Inicialmente, foi desenvolvido uma versão preliminar do Sistema de Recomendação e-LITER, utilizando a técnica de filtragem colaborativa, implementando o modelo SVD configurado com os parâmetros “Default” e utilizando a validação cruzada (cross_validate) para avaliar o desempenho em três folds, medindo as métricas RMSE e MAE. Em seguida, o modelo foi treinado no conjunto de treinamento completo e testado no conjunto de teste para calcular o RMSE e MAE final.

O valor calculado de RMSE para o modelo SVD com os parâmetros básicos foi de 0.8344 e de MAE 0.6443. Na literatura sobre sistemas de recomendação, especialmente aqueles aplicados ao domínio de livros e entretenimento, um RMSE na faixa de 0.8 a 1.2 é frequentemente relatado como dentro do intervalo aceitável. O valor de RMSE de 0.8344 está na extremidade inferior dessa faixa, indicando um bom desempenho. Da mesma forma, um MAE de 0.6443 é considerado bom, especialmente quando comparado com benchmarks de sistemas semelhantes.

Foram construídas 4 funções auxiliares para implementação do sistema de recomendação (get_book_id, get_book_info, predict_review, e generate_recommendation), para identificar o ID de um livro com base no título aproximado, retornar informações detalhadas sobre um livro, estimar a avaliação que o usuário daria ao livro e gerar recomendações para este usuário. Com a versão preliminar do sistema finalizada, são geradas três recomendações para o usuário de ID de número 1000 como podemos verificar a seguir:

```
generate_recommendation(1000, svd, books_metadata)
```

```
[{'book_id': 2,
  'isbn': '439554934',
  'authors': 'J.K. Rowling, Mary GrandPré',
  'title': "Harry Potter and the Sorcerer's Stone (Harry Potter, #1)",
  'original_title': "Harry Potter and the Philosopher's Stone"},
 {'book_id': 5,
  'isbn': '743273567',
  'authors': 'F. Scott Fitzgerald',
  'title': 'The Great Gatsby',
  'original_title': 'The Great Gatsby'},
 {'book_id': 6,
  'isbn': '525478817',
  'authors': 'John Green',
  'title': 'The Fault in Our Stars',
  'original_title': 'The Fault in Our Stars'}]
```

Um segundo modelo preditivo foi construído com o algoritmo KNN (k-Nearest Neighbors) para comparação do seu desempenho com o desempenho da versão preliminar do SVD. O KNN é usado para prever as preferências ou interesses de um usuário com base nas preferências de outros usuários que são semelhantes a ele (vizinhos mais próximos).

O modelo KNN foi configurado com a métrica cosseno e demais parâmetros básicos. De forma semelhate ao SVD foi utilizado a validação cruzada (cross_validate) para avaliar o desempenho em três folds, medindo as métricas RMSE e MAE. Em seguida, o modelo foi treinado no conjunto de treinamento completo e testado no conjunto de teste para calcular o RMSE e MAE final.

Ao comparar os resultados do algoritmo SVD com os do algoritmo KNN, observamos uma diferença significativa no desempenho em relação ao RMSE e MAE. O SVD apresentou RMSE e MAE com os respectivos valores em 0.8344 e 0.6443, ou seja, valores menores em comparação com o KNN (RMSE: 0.9619 e MAE: 0.7584), o que indica uma melhor capacidade de predição das avaliações de filmes. Isso significa que as predições feitas pelo SVD estão mais próximas das avaliações reais dos usuários do que as predições feitas pelo KNN.

Podemos observar como consequência do menor desempenho do KNN, a geração de duas recomendações diferentes de livros do KNN para o mesmo usuário de ID de número 1000 verificado no SVD:

```
generate_recommendation(1000, knn, books_metadata)
```

```
[{'book_id': 1,
  'isbn': '439023483',
  'authors': 'Suzanne Collins',
  'title': 'The Hunger Games (The Hunger Games, #1)',
  'original_title': 'The Hunger Games'},
 {'book_id': 2,
  'isbn': '439554934',
  'authors': 'J.K. Rowling, Mary GrandPré',
  'title': "Harry Potter and the Sorcerer's Stone (Harry Potter, #1)",
  'original_title': "Harry Potter and the Philosopher's Stone"},
 {'book_id': 3,
  'isbn': '316015849',
  'authors': 'Stephenie Meyer',
  'title': 'Twilight (Twilight, #1)',
  'original_title': 'Twilight'}]
```

O SVD é uma técnica baseada em fatorização de matriz que modela as interações usuário-item através de uma representação de baixa dimensionalidade, buscando capturar padrões latentes nos dados. Por outro lado, o KNN é um algoritmo de filtragem colaborativa baseado em vizinhos mais próximos, que utiliza a similaridade entre os usuários ou itens para fazer recomendações.

De acordo com a pesquisa de Erritali et al. (2021) sobre a implementação de sistemas de recomendação utilizando tanto KNN quanto SVD, os resultados encontrados neste trabalho corroboram a superioridade do SVD em relação ao KNN em termos de RMSE e MAE.

Essa diferença de desempenho pode ser atribuída à capacidade do SVD de capturar padrões latentes nos dados de forma mais eficaz do que o KNN, especialmente em conjuntos de dados com alta dimensionalidade e esparsidade.

Uma versão final do Sistema de Recomendação e-LITER foi elaborada, realizando um refinamento do modelo SVD através da implementação do GridSearchCV, para otimizar os parâmetros do modelo. Isso incluiu ajustar o número de fatores, o número de épocas, a taxa de aprendizagem e o termo de regularização.

Uma instância do GridSearchCV foi criada usando o modelo SVD, o dicionário de parâmetros (param_grid), medidas de desempenho (RMSE e MAE), validação cruzada de 3 folds (cv=3), e execução paralela (n_jobs=-1 para usar todos os processadores disponíveis).

O grid de parâmetros a serem pesquisados é mostrado a seguir:

```
param_grid = {
    'n_factors': [100],
    'n_epochs': [20, 30],
    'lr_all': [0.005, 0.01],
    'reg_all': [0.02, 0.1]
              }
```

Os melhores parâmetros encontrados para RMSE foram: 'n_factors': 100, 'n_epochs': 30, 'lr_all': 0.01, 'reg_all': 0.1.

Para MAE, os melhores parâmetros encontrados foram: 'n_factors': 100, 'n_epochs': 30, 'lr_all': 0.005, 'reg_all': 0.02.

Um novo modelo SVD foi configurado usando uma combinação equilibrada das melhores configurações de ‘lr_all’ e ‘reg_all’ derivadas das otimizações para RMSE e MAE, resultando em lr_all=0.0075 e reg_all=0.06. O modelo foi treinado usando o conjunto de treinamento completo. O modelo otimizado foi avaliado no conjunto de teste, e os resultados finais para RMSE e MAE foram calculados:

- RMSE: 0.8159, uma melhoria em relação aos resultados antes da otimização, indicando que as previsões do modelo estão mais precisas.

- MAE: 0.6323, também uma melhoria, mostrando que, em média, o erro absoluto das previsões é bastante baixo.

Comparando esses resultados com os resultados dos estudos de Koren et al. (2009) e Rana e Deeba (2019), o Sistema e-LITER alcançou uma melhoria notável em termos de RMSE e MAE, mostrando uma eficácia comparável ou superior às abordagens híbridas ou baseadas em conteúdo.

A tabela a seguir mostra os valores obtidos de RMSE e MAE para os diferentes modelos deste projeto:

Tabela 1 – Valores obtidos para RMSE e MAE nos diferentes modelos

	RMSE	MAE
SVD (versão inicial)	0.8344	0.6443
KNN	0.9619	0.7584
SVD (versão final)	0.8159	0.6323

O SVD, especialmente após a otimização, demonstrou grande adaptabilidade e precisão na previsão de avaliações, superando o modelo SVD inicial e métodos baseados em vizinhança como o KNN. O SVD efetivamente capturou relações complexas e latentes no conjunto de dados

Como pontos negativos para o modelo final do SVD para o Sistema de Recomendação e-LITER podemos citar:

- Complexidade e Custo Computacional: A otimização de parâmetros com GridSearchCV, embora benéfica, aumentou a complexidade e o custo computacional do modelo.
- Limitações em Cold Start: o modelos SVD enfrenta desafios com novos usuários ou itens devido ao problema de cold start.

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou o desenvolvimento de um sistema de recomendação chamado e-LITER, destinado a auxiliar usuários na escolha de livros, explorando técnicas de filtragem colaborativa com o uso do algoritmo Singular Value Decomposition (SVD). O projeto abordou desde a coleta e preparação dos dados até a implementação e avaliação de modelos de recomendação.

O Sistema e-LITER foi avaliado usando métricas padrão de RMSE e MAE, onde foi demonstrado a eficácia na recomendação de livros, com o modelo SVD refinado através da otimização de parâmetros alcançando resultados significativamente superiores aos modelos comparativos, como o KNN e o próprio SVD configurado com os parâmetros básicos. Os ajustes finos realizados através do GridSearchCV permitiram melhorias nos parâmetros do modelo, otimizando o desempenho e oferecendo recomendações precisas e personalizadas para os usuários**.**

As melhorias no sistema de recomendação têm um impacto direto na experiência do usuário, proporcionando recomendações mais precisas e personalizadas. Esta contribuição não só é relevante academicamente, mas também tem implicações práticas significativas para empresas e plataformas que dependem de sistemas de recomendação para engajamento do usuário e satisfação.

O Singular Value Decomposition apresenta características únicas e vantagens quando comparada com outras metodologias como filtragem baseada em conteúdo ou abordagens híbridas:

- Descoberta de Interesses Latentes: o SVD é eficaz em identificar fatores latentes ou características ocultas nos dados que representam padrões de interesses entre os usuários, que podem não ser diretamente observáveis.
- Independência do conteúdo dos Itens: enquanto a filtragem baseada em conteúdo requer metadados detalhados do item (como gênero, autor, temas), a filtragem colaborativa pode funcionar efetivamente sem qualquer informação sobre o conteúdo do item.
- Adaptabilidade a Diferentes Usuários e Itens: a filtragem colaborativa, especialmente via SVD, adapta-se bem à diversidade de usuários e itens, pois aprende com as interações entre usuários e itens, ajustando continuamente as recomendações com base nos novos dados recebidos.
- Superação de Problemas Comuns em Filtragem Colaborativa: o SVD pode mitigar o problema de esparsidade de dados ao reduzir a dimensionalidade

dos dados de entrada, projetando-os em um espaço de características latentes onde os padrões são mais densos e significativos.

Como pontos negativos para o modelo final do SVD para o Sistema de Recomendação e-LITER podemos citar o aumento da complexidade e custo computacional do modelo após a otimização de parâmetros com GridSearchCV, e as limitações em Cold Start para novos usuários e itens.

Visando a realização de trabalhos futuros, podemos combinar a filtragem colaborativa com técnicas de filtragem baseada em conteúdo para abordar as limitações de Cold Start e aumentar a precisão das recomendações. Podemos também explorar técnicas como PCA ou autoencoders para capturar melhor a variação nos dados e possivelmente melhorar a escalabilidade e a eficiência do sistema.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ADOMAVICIUS, G.; TUZHILIN, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. In IEEE Transactions On Knowledge and Data Engineering, vol. 17, no. 6, 2005, p. 734-749.10.

AGRAWAL, R. Book Recommender System [dataset]. Disponível em: https://www.kaggle.com/datasets/rxsraghavagrawal/book-recommender-system/data. Acesso em: 1 mar. 2024.

AHMED, E.; LETTA, A. Book Recommendation Using Collaborative Filtering Algorithm. Applied Computational Intelligence and Soft Computing, Volume 2023, Article ID 1514801, 12 pages, 2023. https://doi.org/10.1155/2023/1514801.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep). Divulgados os resultados do Pisa 2022. Assessoria de Comunicação Social do Inep, 5 dez. 2023. Disponível em: https://www.gov.br/inep/pt-br/assuntos/noticias/acoes-internacionais/divulgados-os-resultados-do-pisa-2022. Acesso em: 1º mar. 2024.

ERRITALI, M., HSSINA, B., GROTA, A. (2021). Building Recommendation Systems Using the Algorithms KNN and SVD. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 9(1), 71–80. https://doi.org/10.3991 /ijes. v9i1.20569.

FERREIRA, F. C.; OLIVEIRA, A. A. Os sistemas de recomendação na web comodeterminantes prescritivos na tomada de decisão. Journal of Information Systems And Technology Management, [s.l.], v. 9, n. 2, p.353-3668, 29 ago. 2012. TECSI.
http://dx.doi.org/10.4301/s1807-17752012000200008.

KOREN, Y., BELL, R., VOLINSKY, C. Matrix Factorization Techniques for Recommender Systems, in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

MANGUEL, A. A Transformação da Leitura na Era Digital. Tradução de Vera Whately. São Paulo: Companhia das Letras, 2013.

NOGUEIRA, D. S. Bookify: Explorando o Uso de Sistemas de Recomendação. 2023. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2023.

NGUYEN, W. Collaborative Filtering Recommendation Method
and SVD-based Incremental Approach. In Proceedings of ACM Conference (Conference'17). ACM, New York, NY, USA, 5 pages, 2017. https://doi.org/10.1145.

RANA, A.; DEEBA, K. Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity). Journal of Physics: Conference Series, vol. 1362, art. 012130, 2019. doi:10.1088/1742-6596/1362/1/012130.

RICCI, F.; ROKACH, L.; SHAPIRA, B.; KANTOR, P. B. Recommender Systems Handbook. 1. ed. Nova York: Springer, 2011.

SARWAR, B., KARYPIS, G., KONSTAN, J., RIEDL. J. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In Fifth international conference on computer and information science,Vol. 1. Citeseer, 27–8.

SCARIOTT, S. Desenvolvimento de um Recomendador de Livros com Base em Técnicas de Mineração de Dados. 2019. Trabalho de Conclusão de Curso (Curso de Sistemas de Informação) - Universidade de Caxias do Sul, Área do Conhecimento de Ciências Exatas e Engenharias, Caxias do Sul, 2019.

SOUZA, B. F. M. Modelos de fatoração matricial para recomendação de vídeos. 2011. 66 f. Dissertação (Mestrado) - Curso de Programa de Pós-graduação em Informática, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, Rio de Janeiro, 2011.

TAVARES, J. F.; ASSUMPÇÃO, M. Filtragem colaborativa com Scikit-Surprise. 2019. Departamento de Ciência da Computação, Universidade Estadual de Maringá. Disponível em: https://maringa.academia.edu/Departments/Ciência_da_Computação /Documents. Acesso em: 1º mar. 2024.

Zhou, X., He, J., Huang, G., Zhang, Y. A personalized recommendation algorithm based on approximating the singular value decomposition (ApproSVD). In 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Vol. 2. IEEE, 458–464.

7. APÊNDICE A - CRONOGRAMA

Etapa 1 (08/02/2024 a 07/03/2024):
– 21/02/2024: Organização do grupo;
– 28/02/2024: Definição do tema e dos objetivos do projeto;
– 01/03/2024: Pesquisa de referencial teórico;
– 04/03/2024: Coleta de dados;
– 04/03/2024: Criação do Repositório GitHub com todos os arquivos do projeto;
– 07/03/2024: Entrega da atividade da Etapa 1.

Etapa 2 (08/03/2024 a 31/03/2023):
– 08/03/2024: Definição das bibliotecas Python;
– 09/03/2024: Análise Exploratória e tratamento dos dados;
– 14/03/2024: Definição das técnicas de treinamento e preparação da base de dados;
– 19/03/2024: Treinamento do modelo inicial / definição da avaliação de desempenho;
– 26/03/2024: Descrição do referencial teórico da técnica proposta;
– 31/03/2024: Entrega da atividade da Etapa 2.

Etapa 3 (31/03/2024 a 19/04/2024):
– 31/03/2024: Análise dos resultados preliminares;
– 02/04/2024: Ajuste do pipeline de treinamento para melhoria do desempenho do modelo;
– 12/04/2024: Reavaliação do desempenho do modelo;
– 16/04/2024: Documentar os passos implementados;
– 19/04/2024: Entrega da atividade da Etapa 3.

Etapa 4 (20/04/2024 a 16/05/2024):
– 20/04/2024 a 08/05/2024: Organização e documentação dos resultados e conclusões;
– 09/05/2024: Preparação dos recursos para apresentação do Storytelling;
– 13/05/2024: Apresentação dos resultados do projeto em vídeo;
– 16/05/2024: Entrega dos documentos finais (documentação, apresentação e software).