Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística e Departamento de Informática
Programa de Especialização em *Data Science* e *Big Data*

Eric Ramos Souza

Análise Comparativa dos Algoritmos SVD e KNN para Sistemas de Recomendação de Produtos em Marketplaces

Curitiba
2025

Eric Ramos Souza

Análise Comparativa dos Algoritmos SVD e KNN para Sistemas de Recomendação de Produtos em Marketplaces

Artigo apresentado ao Programa de Especialização em *Data Science* e *Big Data* da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Prof. Marcos Antonio Zanata Alves

Curitiba

2025

Análise Comparativa dos Algoritmos SVD e KNN para Sistemas de Recomendação de Produtos em Marketplaces

A Comparative Analysis of SVD and KNN Algorithms for Product Recommendation Systems in E-commerce Marketplaces

Eric Ramos Souza[1], Marcos Antonio Zanata Alves[2]

[1] Aluno do programa de Especialização em Data Science & Big Data

[2] Departamento de Informática (DInf), Setor de Ciências Exatas, Universidade Federal do Paraná (UFPR)

Resumo

Com o crescimento exponencial do volume de dados na internet, os sistemas de recomendação tornaram-se essenciais para marketplaces. Este estudo apresenta uma análise comparativa entre os algoritmos K-Nearest Neighbors (KNN) e Singular Value Decomposition (SVD) em sistemas de recomendação. Foi utilizado o dataset "eCommerce behavior data from multi category store" do Kaggle que possui interações de compras dos usuários em uma plataforma de marketplace. A avaliação foi realizada através das métricas de precisão Mean Absolute Error (MAE), Root Mean Square Error (RMSE) e Hit Rate, comparando diferentes recortes de catálogo, entre o top 1.000 ao top 4.000 produtos, e também foi mensurado a eficiência dos modelos em grupos de 500 a 2.500 usuários. Os resultados mostraram que o KNN se saiu melhor em todas as métricas de precisão, já o SVD se mostrou muito mais rápido e eficiente nas predições, sendo 100 vezes mais rápido na geração de recomendações. Contudo, o estudo identificou que a limitação do catálogo aos produtos mais populares criou um ambiente artificialmente denso, isto favoreceu o KNN, pois removeu sua principal desvantagem, a alta esparsidade, e uma das vantagens do SVD, sua capacidade de generalização. O resultado mostrou que para a escolha do algoritmo ideal deve-se considerar volume de dados, esparsidade da matriz e tempo de predição.
Palavras-chave: Sistemas de Recomendação, Filtragem Colaborativa, KNN, SVD, Marketplace/E-commerce

Abstract

With the exponential growth of data volume on the internet, recommendation systems have become essential for marketplaces. This study presents a comparative analysis between K-Nearest Neighbors (KNN) and Singular Value Decomposition (SVD) algorithms in recommendation systems. The "eCommerce behavior data from multi category store" dataset from Kaggle was used, which contains user purchase interactions on a marketplace platform. The evaluation was conducted through precision metrics Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Hit Rate, comparing different catalog subsets, from top 1,000 to top 4,000 products, and the efficiency of the models was also measured across groups of 500 to 2,500 users. The results showed that KNN performed better in all precision metrics, while SVD proved much faster and more efficient in predictions, being 100 times faster in generating recommendations. However, the study identified that limiting the catalog to the most popular products created an artificially dense environment, which favored KNN by removing its main disadvantage, high sparsity, and one of SVD's advantages, its generalization capability. The results showed that for choosing the ideal algorithm, one should consider data volume, matrix sparsity, and prediction time.
Keywords: Recommendation Systems. Collaborative Filtering. KNN. SVD. Marketplace/E-commerce.

1 Introdução

O volume de informações disponíveis na internet vem crescendo exponencialmente a cada ano. Em 2010, possuíamos em torno de 2 zettabytes de dados, já em 2024 chegamos à marca de 147 zettabytes, com a estimativa de crescimento de mais de 20% para o ano de 2025, podendo ultrapassar os 181 zettabytes. Se olharmos para os últimos 10 anos, este aumento representa um crescimento de mais de 10 vezes em relação ao ano de 2015 [14]. Com este crescimento gigantesco, por mais que as pessoas tenham muito mais acesso à informação, vídeos ou produtos, criou-se um grande problema para o usuário e para as plataformas: como encontrar itens verdadeiramente relevantes em meio a tanta informação?

Em resposta a este grande desafio, a tecnologia surgiu com uma solução revolucionária: os sistemas de recomendação. Eles transformaram tão rapidamente a forma de se consumir conteúdo que viraram a base de

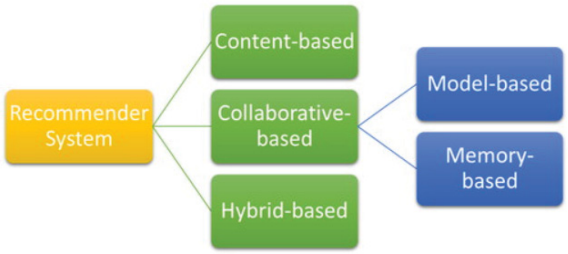
muitas empresas, como Netflix, Spotify, Instagram e TikTok, que construíram impérios baseando-se exclusivamente em algoritmos de recomendação. Até mesmo em marketplaces, a implementação de sistemas de recomendação tornou-se uma questão de sobrevivência, como a Amazon, que têm aproximadamente 35% de sua receita vinda de recomendações personalizadas de produtos [15].

A capacidade de apresentar produtos relevantes no momento certo se tornou crucial para se manter vivo no mercado, e para isso existem duas principais abordagens: a Filtragem Baseada em Conteúdo, que analisa as características e atributos dos produtos para identificar similaridades e sugerir itens parecidos, e a Filtragem Colaborativa, que se baseia no comportamento coletivo para identificar padrões e gerar recomendações.

Este trabalho visa comparar os algoritmos de recomendação KNN e SVD em um contexto de marketplace, avaliando métricas de precisão das recomendações geradas e também a escalabilidade dos modelos.

2 Discussão

Os sistemas de recomendação são algoritmos cujo objetivo é sugerir itens relevantes aos usuários, podendo ser filmes, vídeos, músicas, artigos ou produtos. A finalidade do algoritmo é prever o que um usuário gostaria de comprar/consumir com base no seu comportamento na plataforma. Para realizar essa predição, existem duas principais abordagens, a filtragem baseada em conteúdo e a filtragem colaborativa.



A Filtragem Baseada em Conteúdo (Content-Based Filtering, CBF) foi uma das primeiras abordagens a serem utilizadas. Seu princípio baseia-se na suposição de que, se o usuário teve interesse em um item, há uma grande probabilidade de que outros itens com características semelhantes também sejam de seu interesse. Esse método analisa as características ou metadados dos produtos com os quais o usuário já interagiu e constrói um perfil com base em suas preferências. A partir disso, o algoritmo recomenda outros itens com atributos similares. Sua principal vantagem é a autonomia, pois não depende dos dados de outros usuários para realizar recomendações, o que facilita a sugestão de itens novos (cold-start) e também proporciona maior privacidade, já que os dados do usuário não são utilizados para gerar recomendações para outros usuários. No entanto, este método apresenta algumas limitações como das funcionalidades, onde as características dos itens não capturam exatamente os gostos do usuário, e de superespecialização do modelo, resultando em uma bolha de recomendações muito similares e reduzindo a chance de descoberta de itens diferentes que também poderiam ser relevantes [1].

Porém, o que realmente transformou os sistemas de recomendação foi a chegada da Filtragem Colaborativa (Collaborative Filtering, CF). Em vez de ficar somente nas características dos produtos, o CF começou a utilizar o comportamento de um grupo de usuários para gerar as recomendações, partindo do princípio que os usuários com comportamentos similares tendem a preferir itens similares. Com isso, permitindo a descoberta de novos produtos fora da bolha, quebrando a limitação imposta pela filtragem baseada em conteúdo [2].

Ao invés de se concentrar apenas nas características dos produtos, essa abordagem passou a olhar para o comportamento coletivo dos usuários. A premissa é que as melhores recomendações vêm de outras pessoas com gostos semelhantes aos do usuário alvo. Essa mudança de perspectiva permitiu a descoberta de produtos de forma mais ampla e surpreendente, superando a bolha de recomendações que a filtragem por conteúdo pode criar [2].

Enquanto a Filtragem Baseada em Conteúdo foca nas características dos produtos, a Filtragem Colaborativa adota uma perspectiva diferente, centrada no comportamento das pessoas. Ela opera sob a premissa de que usuários com padrões de avaliação ou interação semelhantes no passado provavelmente compartilharão preferências por novos itens no futuro.

Essa técnica analisa o comportamento e as preferências passadas de um grupo de pessoas para sugerir um conteúdo para o usuário alvo. Considere, por exemplo, dois usuários fictícios, A e B, que frequentemente compram e avaliam positivamente produtos de beleza similares em um marketplace. Caso o usuário A adquira um novo perfume e o avalie com cinco estrelas, o sistema de recomendação identificará essa alta similaridade entre A e B e sugerirá o mesmo perfume para B, que ainda não o comprou. O produto se torna uma forte recomendação por ter sido avaliado positivamente por alguém com um histórico de compras e preferências similares.

Essas abordagens têm como base a matriz de interação usuário-item, também conhecida como matriz de utilidade. As linhas nesta matriz representam os usuários do sistema e as colunas representam os itens do catálogo. Cada célula (u, i) na matriz armazena o valor da interação entre cada usuário u e cada item i [3].

O feedback contido nesta matriz apresenta-se de dois tipos:

- Feedback Explícito: ocorre quando o usuário fornece uma avaliação numérica sobre um item, como uma classificação por estrelas (de 1 a 5). Este tipo de feedback é considerado de alta qualidade, por expressar com mais detalhes o nível de preferência, mas tende a ser mais difícil de conseguir;
- Feedback Implícito: é pressuposto a partir do comportamento do usuário, sem ele fornecer uma opinião direta. Por exemplo, cliques, visualizações de página, compras, tempo gasto em uma página de produto, ou adições ao carrinho de compras. O feedback implícito é muito mais abundante em plataformas de e-commerce. Entretanto, é um dado que pode não representar exatamente a realidade. Um clique não assegura uma preferência positiva, ou a compra de um produto não garante uma experiência positiva com o produto.

3 Materiais e Métodos

No conjunto de técnicas de Filtragem Colaborativa, alguns algoritmos se destacam como os melhores representantes de cada tipo de abordagem. O K-Nearest Neighbors (KNN) é o exemplo clássico da abordagem baseada em memória, enquanto a Singular Value Decomposition (SVD) é uma técnica frequentemente mencionada no contexto das abordagens baseadas em modelo, pois muitos algoritmos derivam desta técnica [1, 2].

3.1 Abordagens Baseadas em Memória (Memory-Based)

Quando pensamos nas estratégias baseadas em memória, lidamos com a versão mais direta da filtragem colaborativa. O sistema somente armazena cada avaliação que já aconteceu entre usuários e itens e consulta esse histórico na hora de criar uma sugestão. Dentro dessa abordagem cabem duas frentes principais [2]:

- Filtragem Baseada no Usuário (User-Based CF): Ela tem como base buscar outras pessoas com gostos parecidos ao usuário-alvo. Essa semelhança é medida comparando as linhas da matriz, e pode ser calculada usando métricas como Similaridade de Cosseno, Correlação de Pearson, entre outras. Em seguida, a avaliação prevista para o usuário-alvo para cada item é calculada como uma média ponderada das avaliações que os vizinhos do usuário deram para o mesmo item [2, 3].
- Filtragem Baseada no Item (Item-Based CF): Nesta abordagem, ao invés de buscar usuários parecidos, buscamos itens que costumam receber avaliações semelhantes. O sistema compara colunas da matriz e descobre, por exemplo, que quem comprou o item A tende a comprar o item B. Para estimar a nota do usuário para um item específico, o sistema identifica os itens mais similares que o usuário já avaliou e calcula uma média ponderada das notas, onde o peso é determinado pela similaridade entre os itens. Em geral, as relações entre itens permanecem estáveis por mais tempo do que o gosto dos usuários, o que ajuda esse método a escalar melhor [2, 4].

3.1.1 K-Nearest Neighbors (KNN)

O algoritmo K-Nearest Neighbors (KNN) é a representação mais direta do conceito de“ mais próximos” em sistemas de recomendação, sendo considerada a implementação clássica da Filtragem Colaborativa baseada em memória. Seu objetivo é identificar um subconjunto de usuários ('vizinhos') com perfis de preferência mais semelhantes ao do usuário-alvo para, a partir de suas interações, inferir novas recomendações [4, 5].

O processo de implementação do KNN em um sistema de recomendação é muito simples, independente se ele baseado no usuário ou no item. Abaixo temos os passos para a implementação da Filtragem Colaborativa baseada no usuário (user-based CF) utilizando o algoritmo KNN [4, 5].

- Estruturação dos Dados: Primeiramente, os dados de ratings ou de interações são estruturados na matriz usuário-item. Nesta matriz, cada linha representa um vetor de avaliações de um usuário, e cada coluna representa um vetor de avaliações de um item.
- Cálculo de Similaridade: Existem diferentes métodos de se calcular a distância ou similaridade para quantificar a “proximidade” entre os vetores. Para dados de recomendação em marketplaces, os quais são vetores de alta dimensão e esparsialidade, a similaridade de cosseno é a métrica mais comum e aconselhada. Ela mede o cosseno do ângulo entre dois vetores, focando na orientação (padrão de gosto) em vez da magnitude (nível de avaliação) [2, 5].

$$\text{Cosine}(x, y) = \frac{\sum_{i=1}^{n} x_i \, y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \, \sqrt{\sum_{i=1}^{n} y_i^2}}$$

- Encontrando Vizinhos: Para um usuário-alvo, o algoritmo KNN identifica os Top-N (k) usuários mais semelhantes com base nas pontuações de similaridade calculadas.
- Geração de Predições: Uma vez que os k vizinhos são identificados, suas avaliações para um item específico (que o usuário-alvo não avaliou) são agregadas. A forma mais comum de agregação é uma média ponderada, onde o peso da avaliação de cada vizinho é proporcional à sua similaridade com o usuário-alvo [4, 5].

- Recomendação: O processo é repetido para todos os itens não avaliados pelo usuário-alvo. Os itens com as maiores pontuações de predição são então selecionados para compor a lista de Top-N recomendações.

A principal vantagem do KNN é sua simplicidade na implementação e a facilidade de interpretar as recomendações realizadas. No entanto, suas desvantagens são significativas quando se trata de escala ou matrizes com alta esparsidade. Está técnica possui um alto consumo computacional, ao calcular as similaridades com muitos outros vetores em tempo de predição. Outra dificuldade é a degradação da performance em decorrência da esparsidade da matriz, por haver pouca sobreposição nas interações entre usuários, sendo esta situação muito comum em cenários de marketplace [2, 4, 5].

A transição da Filtragem Colaborativa baseada em memória para a baseada em modelos foi uma resposta direta aos desafios de negócio e escalabilidade. Em grandes plataformas de marketplace, onde a matriz de interação pode ter uma esparsidade superior a 99%, a probabilidade de dois usuários terem avaliado um número significativo de itens em comum é extremamente baixa. Isso torna a busca por “vizinhos” (a base dos métodos de memória) ineficaz e computacionalmente custoso para milhões de usuários. Para superar essas barreiras, surgiram as abordagens baseadas em modelo [2, 4, 5].

3.2 Abordagens Baseadas em Modelo (Model-Based)

Em vez de consultar diretamente a matriz de interações, as abordagens baseadas em modelos utilizam esses dados para treinar um modelo de aprendizado de máquina. O objetivo do modelo é aprender os padrões e as preferências implícitas nos dados para, essencialmente, prever e preencher os valores ausentes da matriz esparsa [4, 6].

O ponto forte dessas abordagens é a capacidade de generalizar. Ao invés de depender de sobreposições diretas, técnicas como Fatoração de Matrizes e Redes Neurais aprendem uma representação compacta e de baixa dimensão de usuários e itens. Essas representações são conhecidas como fatores latentes [4, 6].

Esses fatores latentes podem ser interpretados como vetores que representam a essência dos interesses de um usuário e as características intrínsecas de um item em um espaço de características de baixa dimensão.[6, 8] O modelo aprende que as preferências podem ser explicadas por um pequeno número de conceitos abstratos. Como, por exemplo, o sistema pode notar que usuários que compram fraldas e lenços umedecidos também se interessam por outros produtos de bebê. Ele aprende o conceito latente de “novo pai/mãe” e pode sugerir itens relevantes a um novo usuário com o mesmo comportamento inicial.

A vantagem dessa abordagem resolve o principal gargalo técnico do método anterior. Primeiramente, o modelo demonstra uma notável tolerância à esparsidade, pois ao generalizar a partir de padrões latentes, consegue operar de forma eficaz mesmo com dados escassos. Em termos de arquitetura, também oferece alta escalabilidade. O treinamento, embora computacionalmente custoso, é um processo offline, tornando a geração de recomendações em tempo real extremamente rápida e eficiente para milhões de usuários, um benefício direto da redução de dimensionalidade e da simplicidade dos cálculos na fase de predição [4, 6, 8].

Do ponto de vista da qualidade, esses avanços técnicos se traduzem em uma experiência superior. A capacidade de capturar padrões complexos resulta em recomendações mais precisas e relevantes. Mais do que isso, ao identificar conexões não óbvias, o sistema promove a descoberta e a serendipidade, enriquecendo a jornada do usuário. Portanto, essa mudança representa uma transição de uma abordagem reativa (que depende de dados existentes) para uma preditiva, capaz de inferir preferências em um imenso volume de dados ausentes [4, 6].

Um dos grandes representantes desta abordagem é o Singular Value Decomposition (SVD) que serve como base para diversas técnicas baseadas em modelo.

3.2.1 Fatoração de Matrizes com Singular Value Decomposition (SVD)

A Singular Value Decomposition (SVD) é uma técnica de álgebra linear poderosa que se tornou a base para muitas abordagens de CF baseadas em modelo. Seu objetivo é ir além das interações superficiais e descobrir as características latentes (fatores ocultos) que governam as preferências dos usuários e as propriedades dos itens [6, 9].

O conceito matemático por trás da SVD é decompor a matriz de interação original A (de m usuários por n itens) no produto de três outras matrizes:

$$A = U\Sigma V^T$$

Onde:

- U é a matriz de fatores latentes dos usuários, com dimensões m × k. Cada uma das m linhas desta matriz é um vetor que representa um usuário no espaço latente de k dimensões. Este vetor pode ser interpretado como a afinidade do usuário por cada um dos k “conceitos” latentes. Por exemplo, em um contexto de filmes, os conceitos poderiam ser implicitamente aprendidos como “ação x nichado (0,8” “comédia x blockbuster (0,3)”, etc.
- V é a matriz de fatores latentes dos itens, com dimensões n × k. Cada uma das n linhas desta

matriz (ou colunas em sua transposta, V^T) é um vetor que representa um item no mesmo espaço latente de k dimensões. Este vetor descreve a composição do item em relação a esses conceitos. Por exemplo, o quanto um filme específico se alinha com os conceitos de “ação”, “comédia”, etc.

- Σ é uma matriz diagonal de k × k cujos elementos na diagonal são os valores singulares. Esses valores, ordenados por magnitude, representam a importância de cada fator latente. Fatores latentes com valores singulares maiores capturam mais variância nos dados de interação originais.

A dimensão k, o número de fatores latentes, é um hiperparâmetro fundamental. Ela é escolhida para ser muito menor que m e n, resultando em uma redução de dimensionalidade significativa. É essa redução que força o modelo a aprender uma representação compacta e generalizada dos dados.

A SVD lida com a esparsidade de forma inerente. O modelo não precisa da matriz A completa para aprender os fatores latentes em U e V. Ele pode ser treinado usando apenas as avaliações conhecidas. Uma vez que os vetores latentes são aprendidos, o modelo pode prever qualquer avaliação ausente na matriz original. A avaliação prevista para um par (u, i) é simplesmente o produto escalar do vetor do usuário u (da matriz U) e o vetor do item i (da matriz V). Em termos matriciais, a matriz completa de predições é obtida pela reconstrução: $A = U\Sigma V^T$. O modelo generaliza a partir dos padrões de gosto, em vez de depender de interações diretas. Com base nos conceitos apresentados, a próxima seção abordará a avaliação e mensuração do desempenho desses modelos [4, 6-9].

3.3 Mensuração dos Modelos

A forma mais confiável de avaliar um sistema de recomendação é através de testes online, como os Testes A/B, pois consegue capturar o comportamento real dos usuários em um ambiente de produção, refletindo com mais precisão a complexa realidade do comportamento humano. Em vez de simular cenários, um teste online captura diretamente o impacto do sistema em métricas de negócio vitais, como a taxa de cliques, o tempo de engajamento, as conversões de vendas e a receita incremental [4, 11].

Contudo, os testes online não são viáveis neste projeto acadêmico devido a limitações práticas como, a necessidade de acesso a usuários reais em ambiente de produção e as restrições relacionadas à privacidade de dados. Como alternativa, este estudo utilizará avaliação offline, utilizando um dataset público e histórico para simular o desempenho dos algoritmos e validar as hipóteses de pesquisa. A performance será mensurada através de métricas específicas, começando pelas que quantificam o erro de predição dos ratings [4, 10, 11].

3.3.1 Mean Absolute Error (MAE)

Para realizar a avaliação do desempenho dos modelos será mensurado por meio de técnicas de validação cruzada(cross-validation). Primeiramente, para medir a precisão da predição de notas, serão calculadas as métricas MAE e RMSE através do procedimento padrão de cross-validation [10].

O Mean Absolute Error (MAE) calcula o desvio médio absoluto entre a nota prevista $\hat{y}_i$ e a nota real y_i, atribuindo o mesmo peso a todos os erros e mantendo a interpretação na mesma unidade da escala de avaliação, por exemplo, estrelas de 1 a 5 [10, 11].

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Por não elevar os erros ao quadrado, o MAE é pouco sensível a outliers e facilita a explicação do “o quanto, em média, erramos por filme” [10].

3.3.2 Root Mean Square Error (RMSE)

O RMSE segue a mesma ideia do MAE, mas eleva o erro ao quadrado antes de calcular a média e devolve o resultado à mesma escala tomando a raiz [10]:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Esse procedimento penaliza desajustes grandes de forma mais severa, tornando o RMSE sensível a outliers. Na prática, continua sendo a métrica offline mais adotada porque resume a variância dos erros com facilidade de interpretação [10].

Tanto MAE quanto RMSE avaliam a exatidão de notas individuais, mas não respondem à pergunta central para o usuário: “Recebi recomendações interessantes?”. Por isso complementamos a análise com a métrica Top-N Hit Rate [11-13].

3.3.3 Top-N Hit Rate (HR)

Para avaliar a qualidade de uma lista de recomendações, é fundamental verificar se ao menos um dos itens sugeridos é de fato relevante para o usuário. A métrica ideal para essa tarefa é a Taxa de Acerto (Hit Rate), que mede exatamente essa capacidade do sistema de “acertar” uma recomendação pertinente dentro de um ranking limitado de sugestões [11-13].

A metodologia para calcular essa métrica será a leave-one-out cross-validation. O processo é intuitivo: para cada usuário em nosso conjunto de dados, ocultamos temporariamente uma de suas interações positivas. Com o restante de seu histórico, o modelo gera uma lista com as 10 melhores recomendações (Top-10). Se o item ocultado estiver presente nessa lista, contabilizamos um “acerto”. Ao final, a Hit Rate é

calculada como a razão entre o número total de acertos e o número total de usuários no conjunto de teste, fornecendo uma medida clara de sua eficácia em cenários realistas [12]. A fórmula é definida como:

$$HR = \frac{\text{Hits}}{\text{Total de Usuários}}$$

Apesar de sua eficácia, é importante reconhecer que a Hit Rate possui uma limitação: ela trata todos os "hits" da mesma forma, sem diferenciar um item encontrado na primeira posição de um na décima. Para isto, existem métricas derivadas que abordam essa e outras questões, entretanto não serão abordadas neste artigo. Entre elas, destacam-se o ARHR (Average Reciprocal Hit-Rank), que atribui um peso maior aos acertos no topo da lista; o cHR (Conditional Hit Rate), que pode filtrar as avaliações para considerar apenas itens acima de um limiar de relevância; e o rHR (Rating-based Hit Rate), que analisa o desempenho em diferentes faixas de notas. Contudo, para os objetivos deste projeto, a análise se concentrará exclusivamente na Hit Rate como métrica fundamental da utilidade do ranking [11, 13].

3.4 Tratamento de Dados

O ponto de partida deste estudo foi o conjunto de dados "eCommerce behavior data from multi category store" do Kaggle, que detalha o comportamento de usuários em um grande marketplace. Para alinhar a base ao objetivo de prever compras, filtramos o volume original de 0,5 bilhão de eventos, retendo somente os 6 milhões de interações de compras (purchase). Embora representem apenas um pouco mais de 1% do total, as compras são um sinal implícito de preferência muito mais fidedigno do que uma simples visualização de página (page_view) ou uma adição ao carrinho (add_to_cart), que pode ser abandonada. Essa decisão reduziu drasticamente o tamanho dos dados sem comprometer a relevância do nosso alvo preditivo.

Em seguida, foi aplicada uma etapa de limpeza e pré-processamento para garantir a qualidade dos dados. Este processo incluiu a remoção de registros duplicados, a correção de sessões de usuário inconsistentes (como aquelas com mais de um user_id ou com duração superior a um dia), a exclusão de produtos sem categoria associada e a filtragem de usuários com baixo engajamento, que compraram menos de 10 produtos distintos. Além disso, novas colunas foram criadas colunas auxiliar na análise exploratória dos dados.

Outro tratamento importante foi a identificação e exclusão de usuários com comportamento de compra B2B. Utilizando padrões como um alto volume de compras ou a aquisição repetida do mesmo produto em um curto período, esses usuários foram removidos. Este filtro é importante, pois o comportamento de empresas não representam os gostos pessoais daquele usuário e acabam distorcendo o modelo cujo objetivo é identificar padrões de usuários do varejo.

Mesmo após todos os tratamentos, a matriz de interação final ainda era massiva, com aproximadamente 25.000 usuários e 22.000 produtos. O tamanho da matriz inviabilizou o uso do algoritmo KNN, que exige um consumo de memória extremamente elevado para armazenar a matriz de similaridade. Uma tentativa inicial de reduzir percentualmente o dataset não foi bem sucedida, pois a dimensionalidade do catálogo de produtos permanecia muito alta.

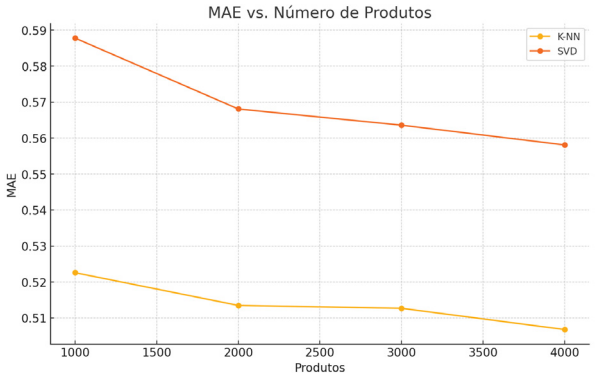
Como solução, foi necessário aplicar um corte drástico no catálogo, limitando o escopo do modelo para incluir no máximo os 4.000 produtos mais populares. Embora essa decisão tenha viabilizado o processamento computacional, ela introduziu um forte viés de popularidade no sistema. Com isso, a análise a seguir compara o desempenho dos 2 modelos, para vermos os resultados práticos e os possíveis efeitos desta limitação.

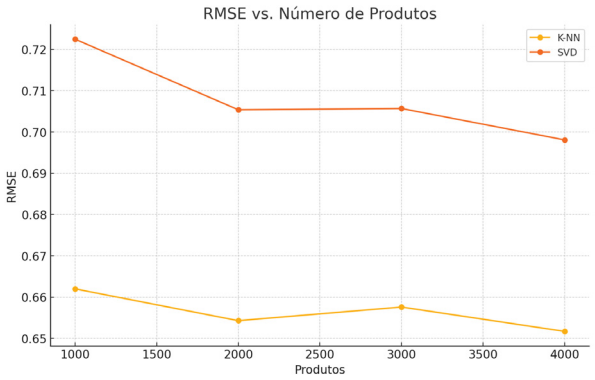
4 Resultados

A análise comparativa teve como objetivo avaliar tanto a precisão das recomendações quanto a agilidade das recomendações. A precisão foi medida através das métricas MAE, RMSE e Hit Rate, com testes realizados em diferentes recortes do catálogo, dos top 1.000 ao top 4.000 produtos mais populares. Já a eficiência foi avaliada medindo o tempo de predição para diferentes grupos de usuários, sendo de 500 a 2.500 usuários, permitindo assim analisar não somente a qualidade, mas também a escalabilidade e viabilidade de cada modelo.

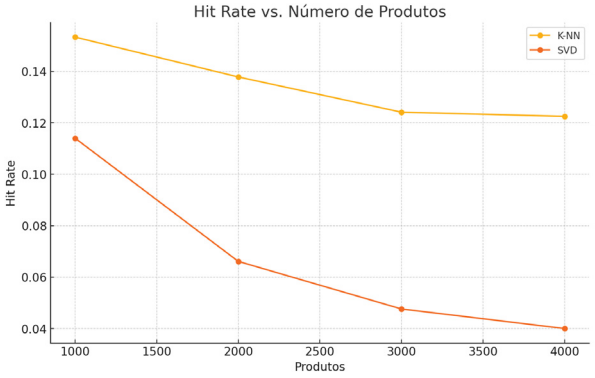
4.1 Precisão

Com relação à precisão do modelo foi obtido um MAE de 0,5226 e 0,5068 para o KNN para os top 1000 e 4000 produtos respectivamente, já para o SVD obtivemos 0,5878 e 0,5581. Já no indicador RSME foi alcançado um valor de 0,662 e 0,6517 no modelo KNN e no modelo SVD 0,7225 e 0,6981. Com isso podemos identificar que o algoritmo ganhador nas duas métricas foi o KNN. Entretanto, quando analisamos mais detalhadamente o resultado, conseguimos ver o SVD teve uma piora menor comparando os top 1.000 vs os top 4.000 produtos, indicando se projetarmos para o total de 22.000 produtos, o modelo SVD tenderá a se sair melhor que o KNN nas métricas de precisão.



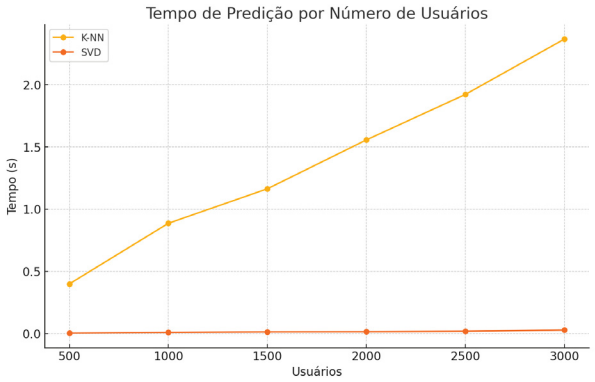


Ao olharmos para a métrica Hit Rate, já possuímos um cenário diferente, onde foi alcançado um valor de 0,1533 e 0,1225 para o algoritmo KNN e 0,114 e 0,04 para o modelo utilizando SVD. Nesta métrica ainda vemos uma vitória do KNN, entretanto a diferença entre os top 1000 vs os top 4000 produtos se inverteu, onde o KNN teve uma piora inferior ao SVD.



4.2 Eficiência

Ao olharmos para eficiência o jogo vira, para uma amostra de 500 usuários o KNN demorou 0,3993 s para gerar uma recomendação para todos os usuários, já o SVD demorou apenas 0,0038 s, se mostrando mais de 100x mais rápido que o KNN. E mesmo com o aumento do número de predições de 500 para 2500 usuários mantivemos uma média superior a 100x mais rápido, onde o KNN realizou no tempo de 1,9211 s e o SVD em 0,0188 s.



Observamos que o algoritmo KNN se saiu superior em todas as análises comparado com o SVD mesmo não sendo o modelo ideal para um cenário de marketplace. Entretanto, isto se deu, pois o maior desafio de KNN, foi retirado ao limitar a linha de produtos somente para os top 4000 produtos, que seria a esparsidade. Como o KNN parte do princípio do vizinho mais próximo, ao limitar o catálogo em 4000 produtos mais populares, foi criado um ambiente de dados artificialmente denso, contendo uma alta sobreposição de usuários, facilitando encontrar vizinhos, e tornando a recomendação direta mais eficiente.

Já quando olhamos para o SVD seu ponto forte não é uma recomendação mais direta, e sim encontrar fatores latentes (características ocultas) nos dados, com isso generalizando mais a recomendação. O principal cenário no qual o SVD se destaca é na alta esparsidade, a qual foi reduzida drasticamente ao limitar o catálogo a somente 4000 produtos, tornando o SVD menos apropriado para o cenário.

5 Conclusão

Nesta análise comparamos os algoritmos KNN e SVD em um cenário de marketplace. No comparativo entre os algoritmos foi observado que o KNN se saiu melhor em todas as métricas relacionadas a precisão do modelo, alcançando 0,5068 contra 0,5581 pontos no MAE e 0,6517 contra 0,6981 pontos no RSME, e no Hit Rate conseguindo alcançar uma diferença ainda maior de 0,1225 contra apenas 0,04 no SVD, mostrando ser mais preciso em todas as métricas análisadas. Por outro lado, o SVD se mostrou muito mais eficiente computacionalmente, sendo 100 vezes mais rápido na geração das recomendações, sendo capaz de gerar 2500 predições em apenas 0,0188 segundos, enquanto o KNN demorou 1,9211 segundos para gerar a mesma quantidade de predições.

Entretanto, é importante ressaltar que esses resultados foram obtidos em condições que favoreceram o algoritmo KNN. A limitação do catálogo aos 4.000 produtos mais populares criou um ambiente de dados artificialmente denso na matriz de interação, reduzindo a esparsidade típica de marketplaces reais, essa limitação removeu uma das principais dificuldades do

KNN, que é a dificuldade de encontrar vizinhos similares em matrizes com alta esparsidade. Este ambiente também removeu um dos pontos fortes do SVD, que é a sua capacidade de generalizar as recomendações através de fatores latentes, conseguindo gerar boas recomendações mesmo em ambientes onde é dificil encontrar pessoas com comportamentos similares ao do usuário alvo.

Os resultados obtidos reforçaram que a escolha de um algoritmo deve considerar as características do ambiente real de aplicação. Embora o KNN tenha se saído melhor na precisão neste estudo controlado, o SVD apresenta características mais adequadas para os desafios de marketplaces de grande escala, onde a alta esparsidade e a necessidade de escalabilidade são fatores críticos. Para aplicações práticas, é necessário avaliar os trade-offs entre precisão e eficiência, considerando o volume de dados, a esparsidade da matriz e os requisitos de tempo real do sistema.

6 Referências

[1] J. Murel and E. Kavlakoglu. O que é filtragem baseada em conteúdo? IBM Corporation, Mar. 2024. Available: https://www.ibm.com/br-pt/think/topics/content-based-filtering

[2] J. Murel and E. Kavlakoglu. O que é filtragem colaborativa? IBM Corporation, Mar. 2024. Available: https://www.ibm.com/br-pt/think/topics/collaborative-filtering

[3] V. Kurama. "What Is Collaborative Filtering: A Simple Introduction," Built In, Jan. 2025. Available: https://builtin.com/data-science/collaborative-filtering-recommender-system

[4] F. Ricci, L. Rokach, and B. Shapira, Eds., Recommender Systems Handbook, 3rd ed. New York: Springer, Feb. 2022.

[5] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," ResearchGate, Jan. 2011. Available: https://www.researchgate.net/publication/225265448_A_Comprehensive_Survey_of_Neighborhood-Based_Recommendation_Methods

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, Aug. 2009.

[7] J. B. A. Pereira Filho, "Um algoritmo de filtragem colaborativa baseado em SVD," Nov. 2010.

[8] A. Tam, "Using Singular Value Decomposition to Build a Recommender System," Machine Learning Mastery, Oct. 2021. Available: https://machinelearningmastery.com/using-singular-value-decomposition-to-build-a-recommender-system/

[9] R. Hinno, "Singular Value Decomposition (SVD) Algorithm Explained," Built In, Jul. 2024. Available: https://builtin.com/articles/svd-algorithm

[10] F. Berisha, "Quality of the predictions: mean absolute error, accuracy and coverage," ResearchGate, Sep. 2017. Available: https://www.researchgate.net/publication/327646833_Quality_of_the_predictions_mean_absolute_error_accuracy_and_coverage

[11] A. Elahi and A. Zirak, "A Comprehensive Survey of Evaluation Techniques for Recommendation Systems," arXiv, Nov. 2024. Available: https://arxiv.org/html/2411.01354v1

[12] Evidently AI Team, "10 metrics to evaluate recommender and ranking systems," Evidently AI, Feb. 2025. Available: https://www.evidentlyai.com/ranking-metrics/evaluating-recommender-systems

[13] D. Benveniste, "Deep Dive: All the Ranking Metrics for Recommender Systems Explained!," The AI Edge Newsletter, May 2023. Available: https://newsletter.theaiedge.io/p/deep-dive-all-the-ranking-metrics

[14] F. Duarte, "Data generated per day," Exploding Topics, 2025, updated Apr. 2025. Available: https://explodingtopics.com/blog/data-generated-per-day

[15] I. MacKenzie, C. Meyer, and S. Noble, "How retailers can keep up with consumers," McKinsey & Company, 2013. Available: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers