UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Leandro Andrade Damasceno

Aplicação de um Sistema de Recomendação para Ambientes Virtuais de Aprendizagem

RECIFE

2024

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Leandro Andrade Damasceno

Aplicação de um Sistema de Recomendação para Ambientes Virtuais de Aprendizagem

Monografia apresentada ao Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE), como requisito parcial para conclusão do Curso de Ciência da Computação, orientada pelo(a) professor(a) Patrícia Cabral de Azevedo Restelli Tedesco.

RECIFE
2024

UNIVERSIDADE FEDERAL DE

PERNAMBUCOCENTRO DE

INFORMÁTICA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Daasceno, Leandro Andrade.
Aplicação de um sistema de recomendação para ambientes virtuais de aprendizagem / Leandro Andrade Daasceno. - Recife, 2024.
67 p. : il., tab.

Orientador(a): Patrícia Cabral de Azevedo Restelli Tedesco
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2024.
Inclui referências, apêndices.

1. Sistemas de Recomendação. 2. Filtragem Colaborativa. 3. Filtragem Baseada em Conteúdo. 4. Filtragem Híbrida. 5. Machine Learning. I. Azevedo Restelli Tedesco, Patrícia Cabral de. (Orientação). II. Título.

000 CDD (22.ed.)

Leandro Andrade Damasceno

Aplicação de um sistema de recomendação para ambientes virtuais de aprendizagem

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título debacharel em ciência da computação.

Aprovado em:18/10/2024

BANCA EXAMINADORA

_____

Profa. Dra. Patrícia Cabral de Azevedo Restelli Tedesco (Orientadora)

Universidade Federal de Pernambuco

_____

Profa. Dra. Anjolina Grise de Oliveira (Examinadora Interna)

Universidade Federal de Pernambuco

_____

Dedico este trabalho ao meu pai e minha mãe,
meus maiores incentivadores, pela educação
que me deram e pelo amor que me dedicaram.

AGRADECIMENTOS

Agradeço a Deus pelos dons que Ele me concedeu nesta existência, os quais foram fundamentais para a realização deste projeto, proporcionando-me coragem, paciência e sabedoria. Sou grato aos meus pais, Mércia Juvino de Paula e Severino Andrade Damasceno, por sempre me motivarem e confiarem em minha capacidade de superar os obstáculos que a vida me apresentou. Também expresso minha gratidão à minha orientadora, Patrícia, por estar sempre disponível para me guiar na direção correta para o desenvolvimento do trabalho. Além disso, quero agradecer aos meus amigos ao longo do curso, os quais sempre me apoiaram e contribuíram para que eu chegasse até este ponto, próximo da conclusão desta jornada acadêmica.

“Tecnologia pode se tornar as 'asas' que permitirão ao mundo educacional voar mais longe e mais rápido do que nunca - se permitirmos.”
Jenny Arledge

RESUMO

Com o avanço da tecnologia nos dias atuais, hábitos do cotidiano têm passado por uma série de mudanças em cada área da vida, uma dessas áreas é a área educativa. Alunos e professores se viram numa situação de necessidade de se ambientar em plataformas virtuais de ensino. Além disso, existe a necessidade de que alunos recebam feedbacks personalizados das aulas, uma vez que, devido à correria do dia a dia dos professores não seria possível obtê-los. Nesse contexto, a ideia de um sistema de recomendação se destaca, pois é fundamental em uma plataforma de aprendizagem, ajudando desempenho de alunos e professores, filtrando conteúdos de interesse ao usuário, pois há um amplo volume de recursos. Nesse trabalho será visto os conceitos e as principais técnicas e algoritmos de filtragem de informação com aplicações na recomendação de recursos digitais educacionais, com o propósito de aplicação em um usuário usando uma base de dados de exemplo com tecnologia Python e alguns de machine learning.

Palavras-chave: Sistemas de Recomendação, Filtragem Colaborativa, Filtragem Baseada em Conteúdo, Filtragem Híbrida, Machine Learning

ABSTRACT

With the advancement of technology in today's world, everyday habits have been undergoing a series of changes in every area of life, and one of these areas is education. Students and teachers have found themselves in a situation where they need to become familiar with virtual teaching platforms. In this context, the idea of a recommendation system stands out, as it is fundamental in a learning platform, aiding both students' and teachers' performance by filtering content of interest to the user, given the vast volume of resources available. This work will explore the concepts and main techniques and algorithms of information filtering with applications in recommending educational digital resources, with the purpose of application to a user using a sample database with Python technology and some machine learning.

Keywords: Recommender Systems, Collaborative Filtering, Content-Based Filtering, Hybrid Filtering, Machine Learning

Lista de Figuras

Lista de Tabelas

Tabela de Siglas

Sigla	Significado
IA	Inteligência Artificial
SR	Sistema de Recomendação
SVD	Decomposição em valores singulares
AVA	Ambientes virtuais de aprendizagem
OHE	One-hot-encoding

Sumário

1. Introdução

O uso cada vez maior das tecnologias de informação e comunicação está gerando constantes transformações no modo como ensinamos e aprendemos. A internet está se tornando cada vez mais essencial para apoiar esse processo, especialmente ao oferecer suporte aos Ambientes Virtuais de Aprendizagem (AVA). Isso resulta em uma grande variedade derecursos educacionais disponíveis para alunos e professores.

Dentro desse contexto, os professores enfrentam um desafio significativo ao escolher e organizar os numerosos recursos educacionais disponíveis na Web, com o objetivo de melhorar a aprendizagem e a motivação dos alunos [1]. Isso se torna especialmente importante em ambientes de Educação a Distância, onde a motivação é fundamental.

Os Sistemas de Recomendação têm a capacidade de selecionar recursos educacionais com base nas características específicas dos alunos, ou mesmo com base nas características de um grupo de alunos em uma disciplina. Isso significa que esses sistemas podem ser valiosos para os professores ao ajudá-los a escolher os materiais digitais mais adequados para sua disciplina, bem como ao acompanhar o progresso dos alunos. Ao permitir essa personalização, a preocupação em identificar e atender a diferentes perfis de alunos é parcial ou completamente resolvida pelos sistemas de recomendação. Portanto, os Sistemas de Recomendação têm a capacidade de beneficiar todos os envolvidos no processo de ensino e aprendizagem.

O campo de pesquisa sobre sistemas de recomendações tem sido um tema bastante importante nesses últimos anos e chamado atenção tanto da área acadêmica quanto comercial. A cada instante, aproximadamente seis novos sites são lançados, mais de dois mil posts são compartilhados no Instagram, mais de quatro mil imagens são postadas no Facebook e mais de cinco mil tweets são compartilhados no Twitter. Devido a esses números em constanteascensão, os sistemas de recomendação se tornaram instrumentos essenciais e ubíquos, sendo agora empregados para personalizar as seleções e categorizar os conteúdos em plataformas e aplicativos online [2].

Na área da educação com as mudanças das metodologias de ensino nos últimos anos devido a pandemia, alunos e educadores sentiram urgência de uma educação de qualidade, a IA é capaz de oferecer essa educação de alto nível, se baseando nos gostos do ser humano, de experiências já vividas [1].

1.1 Objetivo

O propósito deste estudo é examinar três sistemas de recomendações, cada uma com sua técnica e algoritmo apropriado em ambientes educacionais, ressaltando a importância e a eficiência do mesmo aplicando para um usuário. Analisando as técnicas de filtragem baseada em conteúdo, filtragem colaborativa e filtragem híbrida [3].

1.2. Estrutura do documento

Este documento está organizado em seis capítulos, conforme descrito a seguir:

- **Capítulo 1 - Introdução**: Apresenta o contexto geral da pesquisa, a definição do problema, os objetivos do trabalho (geral e específicos) e a justificativa. Além disso, discute a relevância do tema abordado.

- **Capítulo 2 - Definição de Conceitos**: Este capítulo apresenta os conceitos fundamentais que embasam este trabalho. São discutidos os principais tipos de sistemas de recomendação, como a filtragem colaborativa, filtragem baseada em conteúdo e sistemas híbridos. Abordando tipos e limitações.

- **Capitulo 3 - Revisão de estudos anteriores**: Este capítulo apresenta uma revisão dos trabalhos mais relevantes na área de sistemas de recomendação. São abordados estudos sobre as diferentes abordagens, como a filtragem colaborativa, a filtragem baseada em conteúdo e os sistemas híbridos.

- **Capítulo 4 – Desenvolvimento e implementação**: Este capítulo descreve detalhadamente o processo de desenvolvimento e implementação do sistema de recomendação educacional. São abordadas as etapas de pré-processamento dos dados, a construção das recomendações utilizando diferentes abordagens, como filtragem colaborativa, filtragem baseada em conteúdo e sistemas híbridos. Além disso, discute-se as métricas utilizadas para avaliar o desempenho do sistema, como Root Mean Square Error (RMSE) e outras técnicas de avaliação para medir a qualidade e a precisão das recomendações geradas.

- **Capitulo 5 – Experimentos e análises**: Neste capítulo, são apresentados os experimentos realizados para avaliar o sistema de recomendação educacional. Os experimentos são conduzidos utilizando diferentes técnicas de recomendação, como filtragem colaborativa, baseada em conteúdo e sistemas híbridos. A Métrica RMSE é aplicada para medir o desempenho e a eficácia das recomendações.

- **Capítulo 6 - Conclusão e trabalhos futuros**: O capítulo final sintetiza os principais resultados do trabalho, destacando as contribuições do sistema de recomendação desenvolvido. São discutidas as limitações do sistema e possíveis melhorias a serem implementadas. Também são propostas direções para trabalhos futuros, como a integração de uma interface amigável para o usuário, melhorias no banco de dados.

- **Capítulo 7 - Referências bibliográficas**: Esta seção reúne todas as fontes de pesquisa e referências bibliográficas utilizadas ao longo do desenvolvimento da monografia. Inclui livros, artigos acadêmicos, documentos online e sites que fundamentaram a construção do sistema de recomendação e as discussões teóricas apresentadas nos capítulos anteriores, conforme as normas da ABNT.

2. Definição de conceitos

Esse capítulo mostra os principais conceitos utilizados nessa pesquisa. Essas definições serão baseadas na literatura existente, proporcionando uma visão abrangente e detalhada de cada termo.

2.1. Sistemas de recomendação

Os sistemas de recomendação (SRs) são ferramentas de software que sugerem itens úteis para os usuários, auxiliando em várias tomadas de decisões, como quais produtos comprar, quais músicas ouvir ou notícias a ler.

O termo "item" é usado genericamente para descrever qualquer coisa que o sistema possa recomendar aos usuários. Um SR geralmente foca em um tipo específico de item (uma bebida, um livro, ou notícia) e consequentemente, seu design, interface gráfica com usuário e a principal técnica de recomendação usada para gerar as sugestões são adaptados para fornecer recomendações úteis e eficazes para esse tipo específico de item.

Os SRs são úteis principalmente para pessoas que não sabem lidar com números gigantescos de itens alternativos que um site da web, por exemplo, pode disponibilizar [4]. Um exemplo que será aplicado neste trabalho, seria uma aula para um aluno selecionar diante de tantas e se matricular. As sugestões geralmente são personalizadas, e usuários individuais ou grupos distintos podem receber orientações diversas. Por outro lado, há também recomendações genéricas. Essas são bastante simplificadas em sua geração e frequentemente encontradas em publicações impressas, como revistas ou jornais. Exemplos típicos são ranqueamento de itens (Os 20 melhores discos selecionados, livros, etc.). apesar desses tipos de recomendações possam ser úteis e eficazes em algumas ocasiões, elas não são abordadas pela pesquisa de SRs.

Já no caso de recomendações personalizadas, mesmo em sua forma mais simples, elas são disponibilizadas como listas classificadas de itens. Quando essa classificação é executada os SRs tentam presumir quais os itens mais adequados, com base nas preferências e restrições dos usuários. Para executar essa atividade computacional, os sistemas de recomendação reúnem as preferências dos usuários, que podem ser expressas de forma explícita, como classificações de produtos, ou inferidas através da interpretação das ações do usuário. Por exemplo, um sistema de recomendação pode interpretar a visita a uma determinada página de produto como um indício implícito de preferência pelos itens exibidos naquela página [4].

2.2. Técnicas de recomendação

Entre as técnicas mais comuns para a elaboração de um sistema de recomendação, existem dois tipos muito notáveis: A filtragem baseada em conteúdo e a filtragem colaborativa. Podemos usar uma delas ou a combinação das duas, o que chamamos de filtragem híbrida.

2.2.1 Filtragem baseada em conteúdo

A recomendação baseada em conteúdo usa informações prévias do usuário em relação a um item para recomendar itens afins. Por exemplo, ao explorar vídeos no YouTube, o comportamento de navegação do usuário é monitorado para identificar os tipos de vídeos que ele ou ela prefere e, com base nisso, são recomendados conteúdos similares na funcionalidade de vídeos sugeridos [3].

Na figura 1, é mostrado um exemplo de como funciona a filtragem baseada emconteúdo. O usuário assistiu as disciplinas 1, 2 e 3 e disciplina "c" não contém informações similar a esses, sendo assim, "c" não vai ser recomendado ao usuário, entretanto a disciplina "a" possui características que imitam à três disciplinas assistidas, portanto será recomendado, as disciplinas "b" e "d", obedecendo o nível de similaridade levado em conta no sistema, da mesma forma podem ser recomendados, sendo "b" mais similar ao gosto do usuário em relaçãoa "d".

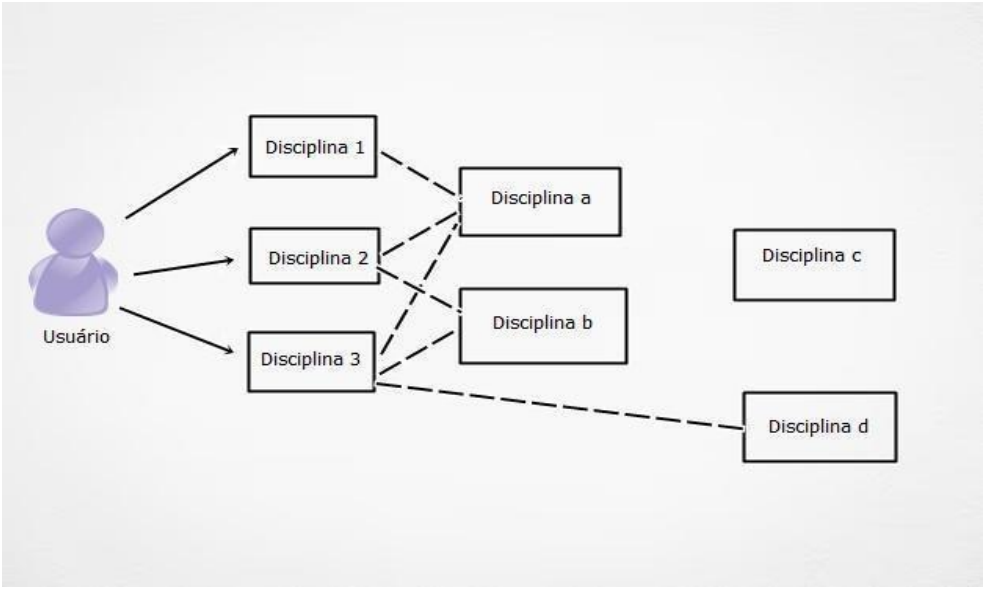


Figura 1. Ilustração da técnica por filtragem de conteúdo

2.2.1.1 Limitações

Nessa seção, serão apresentadas algumas limitações da técnica de filtragem baseada em conteúdo, juntamente com suas descrições [6].

Superespecialização: Apenas serão recomendados itens que o usuário avaliou antes, senso assim, há uma falta de variedades nas recomendações, resultando no problema de superespecialização, onde os usuários não são visíveis a uma pluralidade suficiente de itens novos ou diferentes. Um exemplo seria um usuário de streaming de música que avalia e escuta várias músicas no estilo sertanejo, um sistema de recomendação baseado em conteúdo que explora apenas esses gostos de início pode acabar recomendando apenas músicas desse gênero.

Análise de conteúdo limitado: As técnicas de recomendação baseadas em conteúdo enfrentam várias limitações relacionadas à análise de conteúdo. Elas dependem das características explicitamente associadas aos objetos recomendados, necessitando que o conteúdo esteja em uma forma que possa ser automaticamente analisada por um computador, como texto, ou que as características sejam atribuídas manualmente aos itens. Embora técnicas de recuperação de informação funcionem bem para extrair características de documentos de texto, a extração automática de características em domínios como imagens, áudio e vídeos é muito mais complexa. A atribuição manual de atributos também é impraticável devido à limitação de recursos.

Novo usuário: Ocorre quando um novo usuário se junta ao sistema e não há informações suficientes sobre suas preferências para gerar recomendações personalizadas. A recomendação baseada em conteúdo depende das interações anteriores do usuário com itens (como avaliações, cliques ou histórico de navegação) para identificar características que ele prefere.

2.2.2 Filtragem colaborativa

A recomendação baseada em filtragem colaborativa confia nas avaliações de usuários que compartilham interesses similares. Para isso acontecer os usuários devem classificar os itens do sistema, e essas classificações possibilita achar médias para os itens. Sendo assim o sistema de recomendação pode achar padrões de comportamento e automaticamente sugerir os itens julgados como mais interessantes pelos usuários com preferenciais iguais. A técnica para achar as relações entre o usuário alvo e seus vizinhos mais próximos consiste em três etapas 1) Calcular a similaridade do usuário alvo com os outros usuários; 2) Escolher um grupo de usuários com maiores similaridades para levar em conta na predição; e para fim, 3) Fazer uma normalização das avaliações e avaliar as predições, ponderando as avaliações dos usuários mais similares.

A figura 2 mostra um exemplo de como seria uma técnica de filtragem colaborativa. Levando em conta que o usuário alvo gostou da disciplina 1, 2 e 3, os vizinhos mais próximos são os usuários 1 e 3, pois igualmente curtiram esses três livros, dado que ambos gostaram da mesma forma das disciplinas 4 e 5, o sistema recomenda essas duas disciplinas ao usuário alvo.

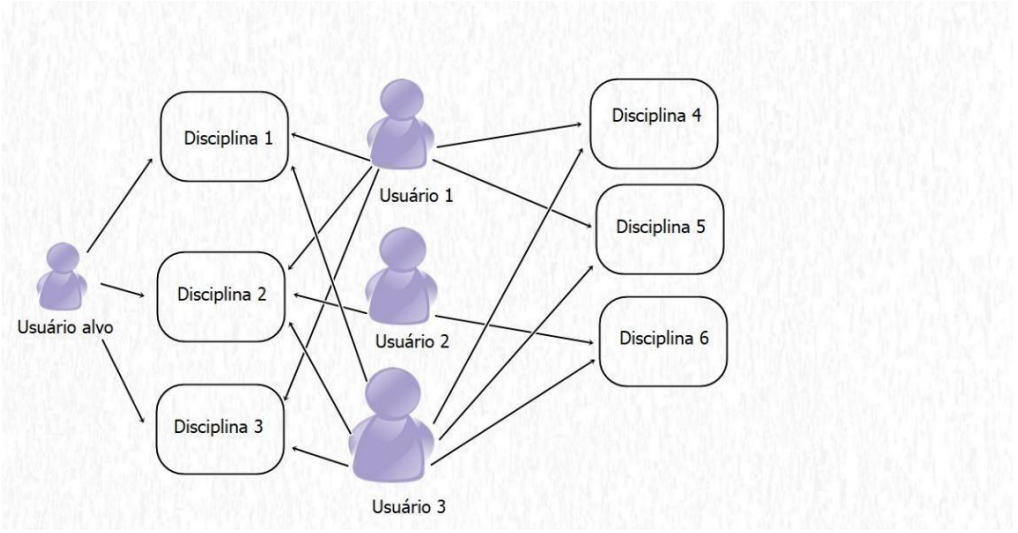


Figura 2. Ilustração da técnica de filtragem colaborativa

Sistemas de filtragem colaborativa podem ser categorizados como sistema baseado em vizinhança e sistemas baseados em modelos.

Conhecidos também como sistemas baseados em memória, os sistemas baseados em vizinhança implementam uma matriz de similaridade que pode ser entre itens ou entre usuários e assim a representação da matriz é mantida na memória. Quando reconhece essa similaridade, o sistema é capaz de recomendar itens novos.

As recomendações que se baseiam em usuários utilizam a similaridade entre os usuários e seus vizinhos mais próximos para estimar a nota que colocaria em itens que não foram avaliados ainda. Por outro lado, as recomendações que se baseiam em itens fazem previsões com base na similaridade entre os itens.

No caso de sistemas baseados em modelos, as operações são feitas com técnica de aprendizagem de máquina. Os algoritmos desse sistema usam o conjunto de avaliações dos usuários para elaborar um modelo com habilidade de fazer predições para depois recomendar novos itens.

Uma distinção muito significativa entre os algoritmos baseados em vizinhanças e os algoritmos baseados em modelos é a de que, no baseado em vizinhança, os cálculos são feitos para um caso por vez, já em modelos, um modelo geral que consegue ser aplicado ao conjunto todo de dados é gerado desde o começo.

2.2.2.1 Limitações

Nessa seção, serão apresentadas algumas limitações da técnica de filtragem baseada em conteúdo, juntamente com suas descrições [6].

Problema do novo item: Novos itens são constantemente adicionados ao sistema de recomendação. Como os sistemas colaborativos dependem exclusivamente das preferências dos usuários, até que o novo item seja avaliado por um número substancial de pessoas, o sistema não será capaz de recomendá-lo.

Exemplo: Usuário usa um serviço de streaming de filmes, exemplo o Amazon Prime, que recomenda filmes com base nas suas preferências e nas preferências de outros usuários. Um novo filme chega ao catalogo da Amazon, daí como esse filme é novo, ele ainda não foi avaliado por muitos usuários. O sistema de recomendação colaborativa da Amazon, que depende das avaliações e preferências de outros usuários para fazer recomendações, não tem informações suficientes sobre o filme.

Problema do novo usuário: Para que um sistema de recomendação baseado em conteúdo compreenda as preferências de um usuário e ofereça recomendações precisas, é necessário que o usuário avalie uma quantidade suficiente de itens. Assim, um usuário recém-chegado, com poucas avaliações, não receberá recomendações exatas.

Exemplo: Suponha que você acabou de se cadastrar em um serviço de streaming de filmes. O sistema de recomendação baseado em conteúdo precisa entender seus gostos para sugerir filmes que você provavelmente vai gostar. No entanto, como você é um novo usuário e ainda não assistiu ou avaliou muitos filmes, o sistema não tem informações suficientes sobre suas preferências. Como resultado, as recomendações iniciais podem não ser tão precisas, pois o sistema ainda está tentando entender quais gêneros e tipos de filmes você prefere. À medida que você assiste e avalia mais filmes, o sistema começará a oferecer sugestões mais alinhadas com seus gostos.

Esparsidade de dados: Refere-se ao problema onde a maioria dos itens disponíveis foi avaliada por apenas alguns usuários, criando uma matriz de dados muito esparsa (com muitos valores em branco). Isso dificulta a capacidade do sistema de encontrar usuários semelhantes e fazer recomendações precisas. Em casos onde os gostos de um usuário são incomuns ou onde um item foi avaliado por poucas pessoas, o sistema pode não conseguir gerar boas recomendações.

Exemplo: Suponha que um usuário usa uma plataforma de streaming de filmes que recomenda novos filmes com base nas avaliações de outros usuários. Na plataforma, existem milhares de filmes disponíveis, mas a maioria dos usuários só avalia algumas dezenas de filmes. Como resultado, muitos filmes têm poucas ou nenhuma avaliação. Se você tiver um gosto de gênero de filme bastante específico, é provável que poucas pessoas tenham avaliado os mesmos

filmes que você gosta. Isso significa que o sistema terá dificuldade em encontrar outros usuários com gostos semelhantes ao seu para oferecer recomendações precisas.

2.2.3 Sistema de recomendação híbrida

Como foi abordado nas sessões 2.2.1.1 e 2.2.2.1, as técnicas de filtragem baseada em conteúdo e colaborativa, respectivamente, possuem suas limitações significativas que podem impactar a eficácia das recomendações. Surge, então, a ideia dos sistemas de recomendação híbrida, que consiste na combinação de duas ou mais técnicas de recomendação. Esses sistemas são projetados para fornecer recomendações mais otimizadas, abordando diretamente as limitações das técnicas individuais.

Considere que um usuário da Netflix goste do filme Pânico 4, olhos famintos 2 e hereditário, ao longo do tempo usando Netflix, o usuário receberá recomendação de filmes de gênero suspense/terror. Do mesmo jeito, se dois usuários estiverem gostando ou visto conteúdo similar na Netflix, cada um vai adquirir sugestões baseadas no que o outro assistiu em seguida.

2.2.3.1 Classificações de sistemas híbridos

Existem 3 jeitos de elaborar um sistema de recomendação híbrida [7]:

1. **Monolíticos**: São aqueles em que um algoritmo de recomendação é composto e gerado de vários tipos de dados, podendo não ter uma distinção clara entre as partes do algoritmo, pois eles são combinados em uma estrutura unificada e não são separáveis ou aplicadas sequencialmente, mas funcionam em conjuntodentro do mesmo modelo. Há casos em que é preciso modificar os algoritmos de recomendação baseado em conteúdo e colaborativo para serem usados nessa abordagem. Se dividem em 3 categorias:
    - **Feature combination**: Recursos de diferentes dados são integrados e usados no contexto de um único sistema de recomendação. Por exemplo, podemos injetar recursos de um modelo de recomendação colaborativa

em um modelo de recomendação baseado em conteúdo. O modelo híbrido é capaz de considerar os dados colaborativos do subsistema contando exclusivamente com um modelo.

- **Meta-Level**: O resultado de uma técnica de recomendação é outro modelo de recomendação. Diferente do feature augmentation, o meta-nível substitui o conjunto de dados original por um modelo aprendido do modelo contribuinte como entrada para o modelo de recomendação principal.

2. **Conjuntos**: Um sistema conjunto utiliza várias técnicas de recomendação separadamente e combina suas saídas (recomendações) através de algum mecanismo, como votação ou ponderação. Esses sistemas podem ser do tipo sequencial ou paralelo: na abordagem sequencial, os algoritmos de recomendação são aplicados um após o outro em uma sequência definida, enquanto na abordagem paralela, múltiplos algoritmos de recomendação são executados ao mesmo tempo, independentemente uns dos outros, e suas saídas são combinadas posteriormente.

    - **Combinação sequencial (ou *cascade*)**: Definem um sistema de recomendação com uma estrutura hierárquica rigorosa, onde o sistema de recomendação principal produz o resultado primário e um modelo secundário é usado para resolver alguns problemas menores do resultado primário, como desempatar pontuações. Esta categoria é do tipo conjunto sequencial
    - **Feature *augmentation***: Uma recomendação é usada para gerar uma avaliação ou classificação que será usada como entrada posteriormente na próxima técnica de recomendação. Esta categoria é do tipo conjunto sequencial, porém com características do monolítico;
    - **Ponderada**: É aquele em que a pontuação de um item recomendado é computada a partir dos resultados de todas as técnicas de recomendação disponíveis presentes disponíveis no sistema. Esta categoria é do tipo conjunto paralelo

- **Comutação**: O sistema se alterna entre diferentes algoritmos de recomendação com base em certos critérios, como o tipo de usuário, o contexto ou a qualidade da recomendação esperada. Em vez de combinar os resultados de vários sistemas de recomendação, como em abordagens paralelas ou sequenciais, a comutação escolhe o algoritmo mais adequado para cada situação.

3. **Mistos**: Quando for prático fazer um grande número de recomendações simultaneamente, pode ser possível usar um híbrido "misto", em que as recomendações de mais de uma técnica são apresentadas juntas. Os sistemas híbridos mistos não se encaixam em nenhuma das duas categorias acima porque eles combinam várias técnicas de recomendação em um modelo que permite uma integração mais flexível e adaptável, sem seguir uma estrutura estritamente unificada como nos sistemas monolíticos, nem uma combinação de saídas separadas como nos sistemas em conjunto.

A estrutura dos sistemas híbridos é ilustrada pela Figura 2.3, que descreve diferentes abordagens como Monolítico, Conjunto e Misto.

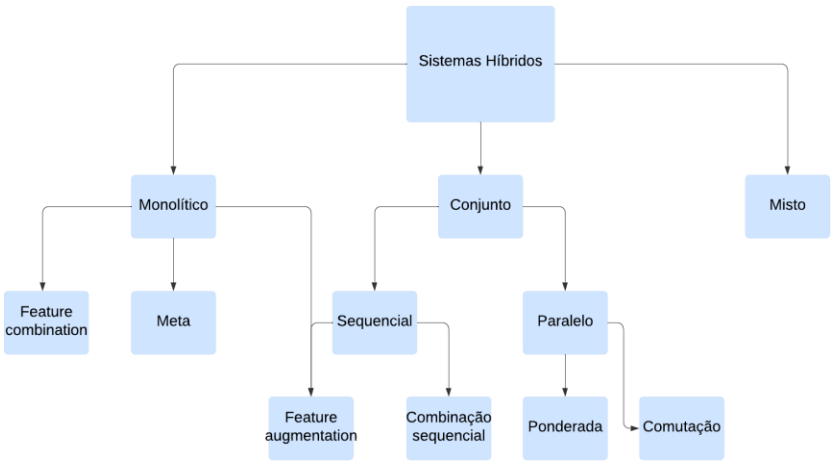


Figura 3. Taxonomia dos sistemas híbridos

3. Revisão de estudos anteriores

Nessa seção, é apresentada uma revisão dos estudos e pesquisas que foram importantes para o tema deste trabalho. A análise dos trabalhos relacionados permite identificar as abordagens, metodologias e resultados obtidos por outros pesquisadores, proporcionando uma base comparativa para o desenvolvimento deste trabalho.

Para a seleção dos trabalhos relacionados, foram estabelecidos os seguintes critérios:

- **Relevância temática**: Apenas artigos que abordassem sistemas de recomendação foram considerados, assegurando a pertinência para o estudo.
- **Metodologia**: Foram adicionados estudos que descrevessem claramente a metodologia empregada, incluindo algoritmos e técnicas de avaliação.
- **Ano de Publicação**: Foram priorizados trabalhos publicados no mínimo da década de 2010 até o ano atual (2024), garantindo que a pesquisa inclua as tendências e avanços mais recentes na área.

3.1. Movie Recommender System

O trabalho "*Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor*" desenvolve um sistema de recomendação de filmes usando os algoritmos *K-Means Clustering* e *K-Nearest Neighbor*. Utilizando o *dataset MovieLens*, os autores implementaram o sistema em Python, detalhando conceitos de aprendizado de máquina e técnicas de recomendação, como filtragem colaborativa e baseada em conteúdo. A arquitetura do sistema, fluxos de processo, pseudo-código e implementação são descritos. O sistema reduz o erro quadrático médio (RMSE) à medida que o número de clusters diminui, obtendo um RMSE mínimo de 1.081648, superando técnicas existentes em termos de precisão [8].

3.2. Sistema de recomendação de produtos utilizando mineração de dados

Este artigo investiga o uso de técnicas de mineração de dados em sistemas de recomendação, com foco na recomendação de produtos em e-commerce. Utilizando técnicas como Clusterização e Regras de Associação, provenientes da Inteligência Artificial, foi desenvolvido um sistema híbrido que integra essas técnicas para melhorar a personalização das recomendações. A pesquisa destaca a importância da personalização para atrair clientes e reduzir o tempo de busca por produtos relevantes. O sistema proposto analisa o histórico de compras dos usuários e compara com outros clientes para prever interesses futuros. A abordagem mostra contribuições significativas para a área de Inteligência Artificial ao integrar diferentes técnicas de mineração de dados e recomendação [9].

3.3. Elaboração de um serviço de recomendação híbrido ponderado e misto implementado em *webservice Restful*

Esse trabalho busca disponibilizar um serviço de recomendação híbrido implantado sobre a estrutura de um WebService RESTful, focado na estruturação dos resultados obtidos em algoritmos de recomendação baseados em filtragem colaborativa, conteúdo e híbrida (utilizando as abordagens ponderada e mista). Foi elaborado um estudo de caso baseado em avaliações de músicas dos mais variados gêneros. O algoritmo de filtragem híbrida ponderada obteve os melhores resultados com 81,4% de acerto nas recomendações, bem como, média das recomendações estatisticamente iguais as médias de avaliações, com base no teste de T [10].

3.4. Sistemas de recomendação para comércio eletrônico na Indonésia

Este artigo se concentra na construção de um sistema de recomendação com um método híbrido paralelo ponderado para comércio eletrônico na Indonésia. O conjunto de dados foi derivado de uma das maiores empresas de comércio eletrônico na Indonésia. Os experimentos usaram três técnicas de amostragem, a saber: validação de *bootstrapping*, séries temporais e amostragem sistemática. O melhor resultado desses experimentos produz uma medida F1 de

9,99%, alcançada pela combinação da abordagem de filtragem colaborativa baseada no usuário e a abordagem de filtragem baseada em conteúdo. Além disso, o valor das métricas de avaliação nesta pesquisa não é muito diferente das pesquisas anteriores sobre sistemas de recomendação. Isso indica que os sistemas de recomendação podem ser aplicados a empresas de comércio eletrônico na Indonésia [11].

3.5. Elaboração de um sistema recomendação híbrido baseado em combinação sequencial

Este trabalho pretende detalhar a filtragem híbrida com o método de combinação sequencial. Também possui por objetivo desenvolver um sistema de recomendação híbrido por combinação sequencial, disponibilizado via uma web service. Após o desenvolvimento, este algoritmo foi testado utilizando um estudo de caso cultural com músicas, onde usuários as avaliavam dando notas de 1 a 5. Depois de uma análise estatística, o sistema de recomendação híbrido por combinação sequencial obteve bons resultados pelo teste do erro quadrático médio, comparada as outras formas de filtragem, tendo um erro de apenas 1,4 ponto em uma amostragem, e 1,27 em outra. [12].

3.6 Conclusão

Os artigos analisam diferentes abordagens para a criação de sistemas de recomendação, com ênfase no uso de algoritmos híbridos que combinam técnicas como filtragem colaborativa e baseada em conteúdo. Esses sistemas, aplicados em domínios como filmes, músicas e e-commerce, destacam-se por melhorar a personalização e a relevância das recomendações. Técnicas de mineração de dados, como K-Means, Regras de Associação e clusterização, são utilizadas para identificar padrões de comportamento dos usuários e prever suas preferências futuras. Além disso, a personalização é vista como crucial para otimizar a experiência do usuário e aumentar a precisão das recomendações.

Os trabalhos demonstram que os sistemas híbridos tendem a ter melhor desempenho, com métricas como o erro quadrático médio (RMSE) e a taxa de acerto mostrando resultados superiores aos métodos tradicionais. A integração de diferentes técnicas e o uso de algoritmos avançados tornam esses sistemas eficazes em vários contextos, incluindo o comércio eletrônico e plataformas de streaming. Em geral, as pesquisas indicam que os sistemas de recomendação baseados em mineração de dados e abordagens híbridas são viáveis, escaláveis e capazes de oferecer uma experiência mais personalizada e precisa.

4 Desenvolvimento e implementação

Nesta seção, serão detalhados os passos que segui para implementar e desenvolver os sistemas de recomendações. O objetivo é fornecer uma visão clara e detalhada de como as soluções propostas foram desenvolvidas e aplicadas na prática. A primeira subseção descreve como foi gerada a criação do conjunto de dados que será trabalhada em cada tipo de recomendação proposta, em seguida, nas seções posteriores, um tipo específico de sistema de recomendação, incluindo os métodos utilizados, os desafios enfrentados e os resultados obtidos. A implementação foi realizada utilizando a linguagem Python no Visual Studio Code.

4.1 Simulador de um Conjunto de Dados Educacionais

. Nesta seção, é descrito o processo de geração de dados simulados para um sistema educacional. Os dados gerados são utilizados para testar e validar diferentes algoritmos de recomendação e análise de desempenho educacional. O código desenvolvido foi implementado em Python, utilizando a biblioteca pandas [13] para estruturar os dados e a função randint para gerar valores aleatórios. O conjunto de dados simulado inclui informações essenciais de um sistema educacional, como identificadores de alunos, cursos, professores, escolas, além de atributos como a duração do curso, modo de estudo e avaliação dos cursos pelos alunos.

O script foi configurado para gerar 1.000 registros, mas sua parametrização permite aumentar ou reduzir a quantidade de dados conforme necessário. As principais variáveis controladas no processo são:

- **n_courses**: número de cursos disponíveis;
- **n_subjects**: número de disciplinas disponíveis;
- **n_teachers**: número de professores gerados;
- **n_schools**: número de escolas representadas no dataset
- **n_students**: número de alunos gerados
- **dataset_size**: total de registros gerados no dataset final

O conjunto de dados gerado possui a seguinte estrutura:

- **course_id**: Identificador único para o curso.
- **student_id**: Identificador único do estudante.
- **subject_id**: Identificador único para a disciplina.
- **teacher_id**: Identificador único do professor.
- **school_id**: Identificador único da escola.
- **course_duration**: Duração do curso em horas, valor aleatório entre 10 e 200
- **study_mode**: Modo de estudo - retorna um inteiro que está mapeado para um modo de estudo específico (‘P’ para presencial, ‘O’ para online, etc.)
- **course_rating** : Avaliação do curso, valor entre 1 e 10

Este dataset pode ser exportado no formato CSV, permitindo facilmente a manipulação em análises posteriores. O script que o script pode ser ajustado para gerar diferentes volumes de dados, o que é útil em experimentos.

4.2 Recomendação baseado em item

Nessa seção será descrito os algoritmos usados para criação do código de geração de recomendações baseadas em item.

4.2.1 Normalização dos dados

Após a geração de dados é preciso normaliza-los com o intuito de que todas as variáveis estejam na mesma escala. As colunas normalizadas foram ‘course_duration_norm’ e a ‘course_rating_norm’, já que ambas podiam variar entre 20 a 200 horas e de 1 a 10, respectivamente.

O objetivo da normalização é alterar os valores das colunas numéricas no conjunto de dados para uma escala comum, sem distorcer as diferenças nos intervalos de valores. Para o aprendizado de máquina, nem todo conjunto de dados requer normalização. Ela é necessária somente quando os recursos têm intervalos diferentes [14].

O código segue a equação de normalização padrão, que é uma forma simples de escalonamento linear dos dados. A equação geral para normalização de um valor x é:

$$x_{normalized} = \frac{x - min}{max - min}$$

4.2.2 One-Hot Encoding (OHE)

Em relação a colunas categóricas, o algoritmo usado foi o OHE, o OHE é uma técnica usada para transformar variáveis categóricas em uma representação numérica. Cada categoria é convertida em uma coluna binária, onde apenas uma coluna (a correspondente à categoria) tem valor 1, e as outras têm valor 0. Isso é útil porque muitos algoritmos de aprendizado de máquina não conseguem lidar diretamente com dados categóricos [15].

O código cria um novo dataframe com as colunas que passaram pelo One-Hot Encoding (OHE). Em seguida, redefine os índices do dataframe gerado e, por fim, concatena o dataframe original com o dataframe gerado pelo OHE.

4.2.3 Cálculo da similaridade de cossenos

A similaridade de cossenos é uma métrica utilizada para medir a similaridade entre dois vetores em um espaço multidimensional. Essa medida é particularmente útil em contextos onde queremos comparar a direção dos vetores, independentemente de suas magnitudes. A fórmula da similaridade de cossenos é dada por:

$$\text{similaridade} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

Onde A e B são os vetores que estamos comparando, o produto de A e B é o produto escalar entre eles e o produto entre ||A|| e ||B|| são as normas desses vetores.

Em sistemas de recomendação baseados em conteúdo, a similaridade de cossenos é amplamente utilizada para recomendar itens semelhantes aos preferidos por um usuário [16].

O trecho de código implementado nesse trabalho primeiro é selecionado o vetor de características do curso de interesse (inputVec) utilizando course_id. Em seguida, calcula a similaridade de cosseno entre inputVec e os vetores de características de todos os cursos no DataFrame, armazenando esses valores na coluna sim. Finalmente, a função retorna os cursos com os maiores valores de similaridade, fornecendo recomendações personalizadas. Esse processo garante que os cursos recomendados sejam os mais semelhantes ao curso de interesse, ajudando na identificação de cursos que atendam às preferências e necessidades do usuário.

4.3 Filtragem colaborativa

Nessa seção será descrita os algoritmos usados para a criação do código de geração de recomendações por filtragem colaborativa.

4.3.1 Tabelas dinâmicas

Técnica amplamente usada para reorganizar dados, especialmente em análise de dados, como é o caso dos sistemas de recomendações. No contexto deste código, a pivot table tem um papel essencial para estruturar os dados de maneira que facilite o cálculo de similaridades entre cursos com base nas classificações atribuídas pelos alunos. No caso de sistemas de recomendação, uma pivot table é utilizada para transformar um conjunto de dados de formato relacional em um formato matricial, onde as colunas são os itens (no caso, cursos), e as linhas são os usuários (neste caso, estudantes).

No código desse trabalho o método pivot_table está sendo usado para organizar os dados de classificação de cursos (com as colunas representando os cursos e as linhas representando os estudantes). A coluna course_rating é usada para preencher as células com as avaliações dos alunos. Quando um aluno não avaliou um curso, a célula correspondente é preenchida com 0 usando o método .fillna(0).

4.3.2 Matrizes esparsas

Nem todos os usuários conseguem adquirir ou avaliar todos os itens, as matrizes utilizadas em algoritmos de filtragem colaborativa tendem a ser esparsas, ou seja, muitos dos elementos da matriz não possuem valores definidos. Os elementos definidos são aqueles em que o usuário forneceu algum tipo de informação sobre o item, como uma nota de avaliação, por exemplo.

No caso desse trabalho, a tabela dinâmica gerada, como dita na seção anterior, poderia gerar uma matriz esparsa pois haveria cursos não ranqueados pelos usuários, como no exemplo abaixo na tabela 1.

Aluno/Curso	Curso 1	Curso 2	Curso 3	.....	Curso N
Aluno 1	8	3	0	...	10
Aluno 2	0	10	6	...	0
Aluno 3	7	0	3	...	10
Aluno 4	2	9	0	....	0
. . .	. . .	6	7	. . .	. . .
Aluno M	1	9	6	...	0

Tabela 1 - Tabela com avaliações dadas por M alunos a N cursos

As avaliações com nota zero foram consideradas ausência de nota.

Ao lidar com matrizes esparsas, que contém um grande número de entradas zero, armazená-las como uma matriz densa é ineficiente. Essa ineficiência ocorre porque uma matriz densa armazena todos os elementos da matriz, inclusive os zeros. Se uma matriz tem dimensões n x m, a matriz densa usará n x m blocos de memória. Como a maioria das entradas em uma matriz esparsa são zeros, é desperdício alocar memória para esses valores zero.

Em Python, as estruturas de dados esparsas são implementadas de maneira eficiente pelo módulo scipy.sparse, que utiliza principalmente arrays Numpy. A ideia é simples: ao invés de armazenar todos os valores de uma matriz densa, vamos apenas guardar os valores diferentes de zero, junto com seus respectivos índices de linha e coluna.

O tipo de matriz esparsa no spicy usada nesse trabalho foi o CSR (Compressed Sparse Rows porque é o formato mais usado e amplamente conhecido. O formato CSR (assim como o CSC, que é a versão comprimida para colunas esparsas) é empregado em tarefas onde se escreve uma vez e se lê várias vezes [18].

No algoritmo desse trabalho foi aplicado o CSR_MATRIX na tabela dinâmica.

4.3.3 Decomposição em valores singulares (SVD)

Entre as técnicas usadas na filtragem colaborativa, uma das mais conhecidas é a Decomposição em Valores Singulares (SVD, do inglês Singular Value Decomposition). Essa técnica envolve a fatoração de matrizes e é aplicada na filtragem colaborativa para identificar características latentes (i.e. escondidas), tanto dos usuários quanto dos itens avaliados. Isso permite inferir as preferências individuais e, assim, realizar recomendações personalizadas [19].

A técnica analisa a matriz em busca de correlações e agrupa dados relacionados, reduzindo a dimensionalidade da matriz original. O SVD transforma uma matriz T (matriz usuário-item) em três matrizes:

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

A primeira matriz, U, representa os usuários, onde cada linha representa um usuário e cada coluna representa uma característica ou dimensão latente do usuário. As características ou dimensões latentes são determinadas pelos valores singulares na matriz $\sum$. A segunda matriz, V, representa os itens, onde cada linha representa um item e cada coluna representa uma característica ou dimensão latente do item.

- U é uma matriz com n linhas e r colunas;
- ∑ é uma matriz diagonal r linhas e r colunas;
- V é uma matriz com d linhas e r colunas

A decomposição ajuda a prever as notas que um usuário pode dar a um item, comparando a relação do usuário e do item com essas características latentes.

Por exemplo, ao aplicar SVD em uma matriz de avaliações de cursos por alunos, podem ser descobertas características como "cursos de ciências exatas" ou "cursos de ciências humanas". A predição das notas que um aluno daria a um curso pode ser calculada pelo produto escalar entre os vetores do aluno e do curso, indicando o nível de afinidade entre ambos. Isso ajuda a recomendar cursos que estejam mais alinhados com os interesses e o perfil de aprendizado de cada aluno.

O objetivo de usar o SVD para fatorar uma matriz de usuário-item é encontrar duas novas matrizes, uma representando usuários e outra representando itens, que, quando multiplicadas, se aproximam da matriz original o mais próximo possível. Essas novas matrizes são chamadas de representações latentes e contêm informações importantes sobre os usuários e itens que podem ser usados para fazer recomendações, como dito anteriormente.

Vamos tomar como exemplo uma matriz T, de dimensões 7 X 5, de usuário-item com duas características latentes:

Aluno/curso	Seminários em inteligência artificial	Sistemas inteligentes	Agentes autônomos	Paradigmas de linguagens computacionais	Algoritmos e estruturas de dados
Aluno 1	1	1	1	0	0
Aluno 2	3	3	3	0	0
Aluno 3	4	4	4	0	0
Aluno 4	5	5	5	0	0

Aluno 5	0	0	0	4	4
Aluno 6	0	0	0	5	5
Aluno 7	0	0	0	2	2

Tabela 2 – Avaliações dos cursos pelo os alunos (Aluno-Item)

$$T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}$$

Temos:

$$U = \begin{bmatrix} 0.14 & 0 \\ 0.42 & 0 \\ 0.56 & 0 \\ 0.70 & 0 \\ 0 & 0.60 \\ 0 & 0.75 \\ 0 & 0.30 \end{bmatrix}$$

A matriz U (7 X 2) representa os alunos, onde cada linha representa um aluno e cada coluna uma característica latente, nesse caso foram criadas duas características latentes: Desenvolvimento e Inteligência artificial, colunas 1 e 2, respectivamente. Nesse exemplo acima vemos que o aluno 1 prefere mais disciplinas de Inteligência artificial (0.14) do que desenvolvimento (0).

Já ∑, é a matriz diagonal de valores singulares representando a intensidade de cada característica. 12.4 é o peso da característica latente desenvolvimento e 9.5 de inteligência

artificial. Como decidimos r = 2 (número de características latentes) a matriz será 2x2, pois pegamos os dois valores mais significativos.

$$\Sigma = \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}$$

E para terminar, a matriz V, cuja as dimensões são 5x2, mas como na fórmula, mostrada anteriormente, ela se aplica usando sua transposta, virando assim 2x5. A matriz V representa a relação item-característica latente. Por exemplo:

$$V^T = \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

A matriz V^T representa a força do item com uma determinada característica latente. No exemplo acima o dado 0.58 (linha 1, coluna 3) significa que o item “Agentes autônomos” tem mais relação com inteligência artificial do que desenvolvimento, que é 0 (linha 2, coluna 5).

4.4 Filtragem híbrida

Nesta seção serão explicados os algoritmos utilizados na criação do código para gerar recomendações por meio da filtragem híbrida.

Como dito na seção 2.2.3, os sistemas de recomendações híbridas são compostos pela combinação da colaborativa junto com a baseado em conteúdo, sendo assim o algoritmo contém o cálculo da similaridade dos cossenos que é usado para calcular a semelhança entre cursos com base nas avaliações de outros usuários. Este é um método de filtragem baseada em conteúdo, que recomenda itens semelhantes ao que o aluno já visualizou ou avaliou. No mesmo código usa-se também o SVD (Singular Value Decomposition), um algoritmo de filtragem colaborativa para prever as classificações que o aluno pode dar a cursos com base em padrões aprendidos a partir das avaliações de outros usuários. Este é um dos principais algoritmos de recomendação colaborativa.

4.5 Métricas de avaliação de performance

É evidente que os sistemas de recomendação têm se tornado indispensáveis atualmente. No entanto, ao criar um sistema desse tipo, é crucial garantir que ele seja capaz de sugerir itens realmente relevantes para os usuários. Para medir a eficácia das recomendações, diversas métricas são amplamente utilizadas. Essas métricas são geralmente divididas em três categorias principais: medidas de exatidão preditiva, exatidão de classificação e exatidão de rankings [20].

A exatidão preditiva se refere ao quão próximos os valores previstos estão em relação aos valores reais de interesse dos usuários. Já a exatidão de classificação foca na frequência com que o sistema acerta ou erra ao recomendar itens. Por fim, a exatidão de rankings avalia seos itens estão sendo recomendados na ordem correta de importância para o usuário [21]. Neste estudo, será considerado a métrica pertencente ao primeiro grupo: exatidão preditiva, precisamente o RMSE.

No grupo conjunto de medidas de exatidão preditiva temos:

4.5.1 MAE

O Erro Médio Absoluto (MAE) quantifica a diferença média entre as avaliações previstas por um sistema de recomendação e os valores reais fornecidos pelos usuários. Ele é calculado pela seguinte fórmula:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

- n: número total de previsões realizadas.
- f_i: valor previsto pelo modelo para a i-ésima observação.
- y_i: valor real da i-ésima observação.
- e_i: erro de previsão, que é diferença entre o valor previsto (f_i e o valor real (y_i)

Essa fórmula mostra que o MAE é obtido ao calcular a média das diferenças absolutas entre as previsões do modelo (f_i) e os valores reais (y_i). O MAE é bastante útil porque interpreta o erro em termos da mesma unidade dos dados originais, o que o torna fácil de entender. Quanto menor o valor do MAE, mais preciso é o modelo.

4.5.2 MSE

O erro médio quadrático (MSE) se diferencia do erro absoluto médio (MAE) por dar mais peso a erros maiores, já que utiliza o quadrado da diferença entre a previsão e o valor real. Isso faz com que desvios maiores tenham um impacto mais significativo no cálculo final do erro.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2$$

4.5.3 NMAE

Essa métrica é uma variação do erro absoluto médio, ajustando o valor final do erro com base nos limites mínimo e máximo das avaliações do sistema.

$$\mathrm{NMAE} = \frac{\mathrm{MAE}}{x_{\max} - x_{\min}} = \frac{\sum_{i=1}^{n}|f_i - y_i|}{n\,(x_{\max} - x_{\min})} = \frac{\sum_{i=1}^{n}|e_i|}{n\,(x_{\max} - x_{\min})}$$

Nessa equação, x_{min} presenta o menor valor possível, enquanto x_{max} refere ao maior valor possível de uma avaliação no sistema.

4.5.4 RMSE

Essa métrica é uma variação do erro quadrático médio, sendo obtida através da sua raiz quadrada.

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{\sum_{i=1}^{n}(f_i - y_i)^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n}(e_i)^2}{n}}$$

Entre as vantagens das medidas de precisão preditiva estão a simplicidade no cálculo, o uso de propriedades estatísticas amplamente conhecidas e a capacidade de avaliar a exatidão das previsões realizadas. No entanto, suas desvantagens incluem o fato de serem altamente específicas e sensíveis, especialmente em sistemas com escalas de avaliação mais limitadas [20].

4.6 Treinamento e Teste com Divisão de Dados

Um dos aspectos fundamentais do aprendizado de máquina supervisionado é a validação e a análise do modelo. Ao medir o desempenho preditivo do seu modelo, é crucial garantir que o procedimento seja isento de parcialidade [22]. Neste trabalho foi usado 'train_test_split ()', função da biblioteca 'surprise'. 'Surprise' é uma biblioteca Python de código aberto projetada especificamente para tarefas de recomendação, facilitando o desenvolvimento e a avaliação de sistemas de recomendação [23].

Primeiro, dividimos os dados em conjunto de treinamento e conjunto de teste. Geralmente, o conjunto de treinamento é muito maior que o conjunto de teste. Agora, treinamos nosso sistema de recomendação usando apenas o conjunto de treinamento. É nessa etapa que ele obtém a relação entre itens e usuários. Uma vez treinado, podemos solicitar que ele faça previsões sobre como um novo usuário poderia avaliar alguns itens que ele nunca viu antes.

```
trainset, testset = train_test_split(data, test_size=0.3, random_state=10)
```

A função train_test_split() divide o conjunto de dados em dois subconjuntos: um para o treinamento (trainset), que corresponde a 70% dos dados, e outro para o teste (testset), que representa os 30% restantes. Essa divisão é fundamental para validar a capacidade de

generalização do modelo, prevenindo que ele se ajuste excessivamente aos dados de treinamento (overfitting).

Os parâmetros test_size e random_state indicam que 30% dos dados foram reservados para o conjunto de teste e assegura que a divisão seja reproduzível, ou seja, o mesmo conjunto de treino e teste será gerado em execuções subsequentes do código, garantindo consistência nos resultados, respectivamente.

A função train_test_split da biblioteca Surprise foi utilizada especificamente porque os dados manipulados neste trabalho tratam de interações entre alunos e cursos, o que configura um cenário clássico de recomendação. O uso dessa função permite a avaliação do modelo de recomendação de forma controlada, simulando como ele se comportaria ao fazer previsões para dados não vistos. Além disso, sua integração direta com o formato de dados da Surprise simplifica o processo de manipulação e avaliação do sistema, evitando transformações manuais adicionais.

5 Experimentos e Análise

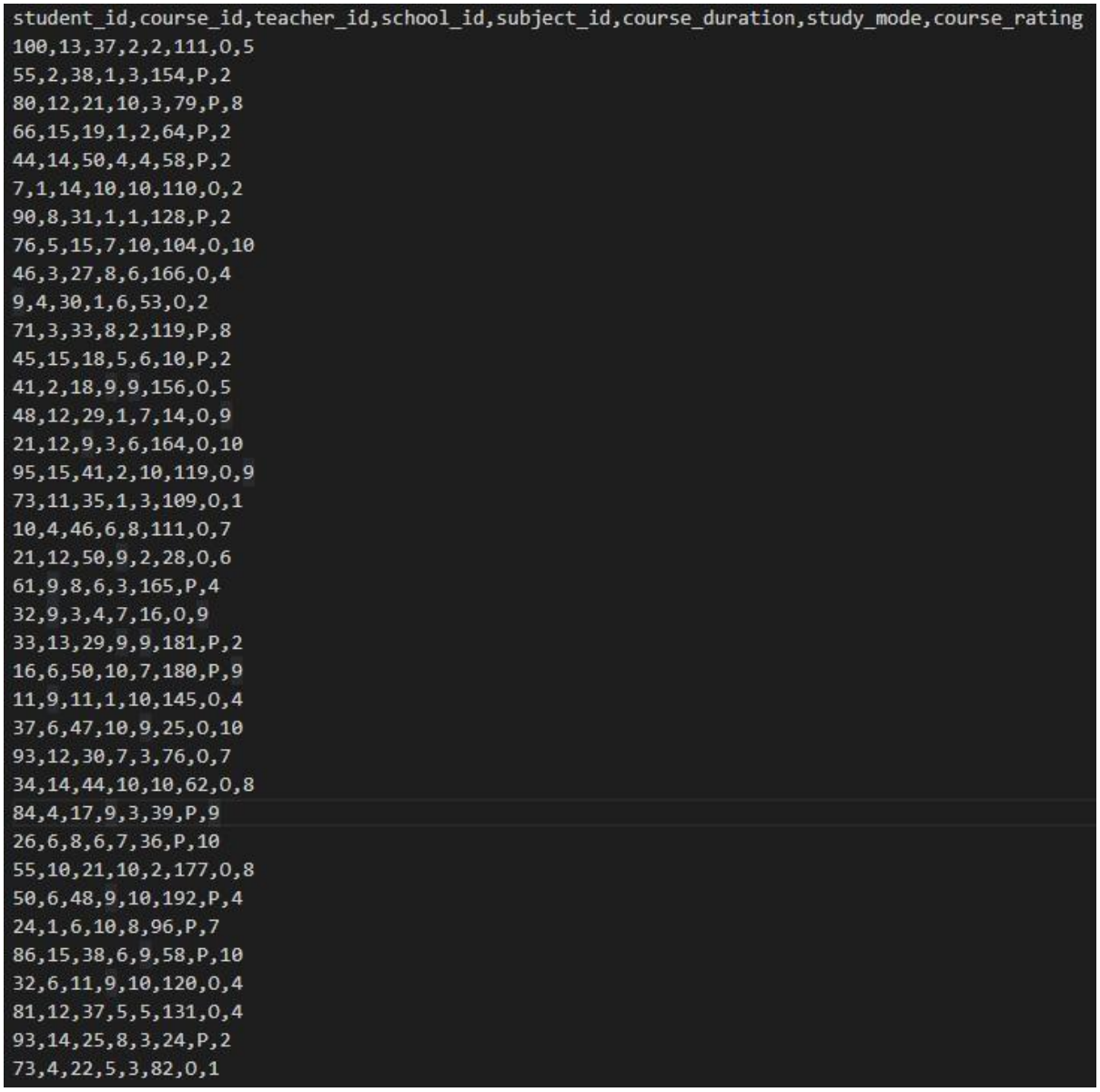
A análise foi conduzida utilizando três métodos principais de recomendação: filtragem baseada em conteúdo, filtragem colaborativa e um modelo híbrido que combina os dois. O objetivo dos experimentos foi comparar a precisão e a efetividade dos diferentes métodos na recomendação de cursos para alunos em uma plataforma educacional.

Os experimentos foram realizados utilizando um conjunto de dados descritos na seção 4.1, de avaliações de cursos por alunos, contendo as seguintes variáveis principais:

- student_id: Identificador único dos alunos.
- course_id: Identificador único dos cursos.
- course_rating: Avaliações que os alunos atribuíram aos cursos.

Os dados foram pré-processados utilizando normalização de atributos e codificação de variáveis categóricas, conforme necessário para os diferentes métodos. Foram gerados 50 registros para os experimentos em arquivo txt.

Entrada:



```
student_id,course_id,teacher_id,school_id,subject_id,course_duration,study_mode,course_rating
100,13,37,2,2,111,O,5
55,2,38,1,3,154,P,2
80,12,21,10,3,79,P,8
66,15,19,1,2,64,P,2
44,14,50,4,4,58,P,2
7,1,14,10,10,110,O,2
90,8,31,1,1,128,P,2
76,5,15,7,10,104,O,10
46,3,27,8,6,166,O,4
9,4,30,1,6,53,O,2
71,3,33,8,2,119,P,8
45,15,18,5,6,10,P,2
41,2,18,9,9,156,O,5
48,12,29,1,7,14,O,9
21,12,9,3,6,164,O,10
95,15,41,2,10,119,O,9
73,11,35,1,3,109,O,1
10,4,46,6,8,111,O,7
21,12,50,9,2,28,O,6
61,9,8,6,3,165,P,4
32,9,3,4,7,16,O,9
33,13,29,9,9,181,P,2
16,6,50,10,7,180,P,9
11,9,11,1,10,145,O,4
37,6,47,10,9,25,O,10
93,12,30,7,3,76,O,7
34,14,44,10,10,62,O,8
84,4,17,9,3,39,P,9
26,6,8,6,7,36,P,10
55,10,21,10,2,177,O,8
50,6,48,9,10,192,P,4
24,1,6,10,8,96,P,7
86,15,38,6,9,58,P,10
32,6,11,9,10,120,O,4
81,12,37,5,5,131,O,4
93,14,25,8,3,24,P,2
73,4,22,5,3,82,O,1
```

Figura 4. Entrada de dados

5.1 Experimento 1 (Filtragem baseada em conteúdo)

O primeiro experimento utilizou um sistema de recomendação baseado em conteúdo. O algoritmo calcula a similaridade de cosseno entre as características normalizadas dos cursos (course_duration e course_rating) para gerar recomendações. O processo foi realizado da seguinte forma:

- **Normalização de Atributos**: As variáveis course_duration e course_rating foram normalizadas para garantir que os valores estivessem no mesmo intervalo.

```
df['course_duration_norm'] = normalize(df['course_duration'].values)
df['course_rating_norm'] = normalize(df['course_rating'].values)
```

Figura 5. Código de normalização em Python

Resultando em:

	student_id	course_id	teacher_id	school_id	subject_id	course_duration	study_mode	course_rating	course_duration_norm	course_rating_norm
0	100	13	37	2	2	111	O	5	0.554945	0.444444
1	55	2	38	1	3	154	P	2	0.791209	0.111111
2	80	12	21	10	3	79	P	8	0.379121	0.777778
3	66	15	19	1	2	64	P	2	0.296703	0.111111
4	44	14	50	4	4	58	P	2	0.263736	0.111111
5	7	1	14	10	10	110	O	2	0.549451	0.111111
6	90	8	31	1	1	128	P	2	0.648352	0.111111
7	76	5	15	7	10	104	O	10	0.516484	1.000000
8	46	3	27	8	6	166	O	4	0.857143	0.333333
9	9	4	30	1	6	53	O	2	0.236264	0.111111
10	71	3	33	8	2	119	P	8	0.598901	0.777778
11	45	15	18	5	6	10	P	2	0.000000	0.111111
12	41	2	18	9	9	156	O	5	0.802198	0.444444
13	48	12	29	1	7	14	O	9	0.021978	0.888889
14	21	12	9	3	6	164	O	10	0.846154	1.000000
15	95	15	41	2	10	119	O	9	0.598901	0.888889
16	73	11	35	1	3	109	O	1	0.543956	0.000000
17	10	4	46	6	8	111	O	7	0.554945	0.666667
18	21	12	50	9	2	28	O	6	0.098901	0.555556
19	61	9	8	6	3	165	P	4	0.851648	0.333333
20	32	9	3	4	7	16	O	9	0.032967	0.888889
21	33	13	29	9	9	181	P	2	0.939560	0.111111
22	16	6	50	10	7	180	P	9	0.934066	0.888889
23	11	9	11	1	10	145	O	4	0.741758	0.333333
24	37	6	47	10	9	25	O	10	0.082418	1.000000
25	93	12	30	7	3	76	O	7	0.362637	0.666667
26	34	14	44	10	10	62	O	8	0.285714	0.777778
27	84	4	17	9	3	39	P	9	0.159341	0.888889
28	26	6	8	6	7	36	P	10	0.142857	1.000000
29	55	10	21	10	2	177	O	8	0.917582	0.777778
30	50	6	48	9	10	192	P	4	1.000000	0.333333
31	24	1	6	10	8	96	P	7	0.472527	0.666667
32	86	15	38	6	9	58	P	10	0.263736	1.000000
33	32	6	11	9	10	120	O	4	0.604396	0.333333
34	81	12	37	5	5	131	O	4	0.664835	0.333333
35	93	14	25	8	3	24	P	2	0.076923	0.111111
36	73	4	22	5	3	82	O	1	0.395604	0.000000

Figura 6. Saída do dataframe normalizado

- **Codificação One-Hot**: As colunas study_mode e subject_id foram transformadas em variáveis binárias.

```
df = ohe(df=df, enc_col='study_mode')
df = ohe(df=df, enc_col='subject_id')
```

Figura 7. Código one-hot-encoding em Python.

Como resultado temos a imagem abaixo, onde podemos ver que os valores da coluna ‘study_mode’, ‘o’ e ‘p’ representando online e presencial respectivamente viraram a colunas, e as id das disciplinas da coluna ‘subject_id’, que iam de 1 a 10. Foi gerado uma dataframe e que em seguida em concatenada a dataframe da imagem acima.

0	P	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0

Figura 8. Saída do dataframe após one-hot-encoding.

- **Similaridade de Cosseno**: A similaridade entre os cursos foi calculada utilizando a função de similaridade de cosseno.

Temos como resultado que para um determinado curso, o sistema foi capaz de recomendar 5 cursos mais semelhantes com base nas características mencionadas, sem considerar a interação entre os usuários. No experimento foi selecionado o curso 24, e como retorno foram os 5 mais similares, ou seja, o que o sistema recomendaria.

Recomendações:

course_id	student_id	teacher_id	school_id	course_rating	course_duration_norm	course_rating_norm	0	P	1	2	3	4	5	6	7	8	9	10	sim
12	81	37	5	4	0.664835	0.333333	True	False	False	False	False	False	True	False	False	False	False	False	1.285823
3	71	33	8	8	0.598901	0.777778	False	True	False	True	False	False	False	False	False	False	False	False	1.283937
11	73	35	1	1	0.543956	0.000000	True	False	False	False	True	False	False	False	False	False	False	False	1.283328
15	86	38	6	10	0.263736	1.000000	False	True	False	False	False	False	False	False	False	False	True	False	1.283185
15	95	41	2	9	0.598901	0.888889	True	False	False	False	False	False	False	False	False	False	False	True	1.281965

Figura 9. Resultado das recomendações baseado em item.

5.2 Experimento 2 (Filtragem colaborativa)

O segundo experimento utilizou a técnica de filtragem colaborativa baseada na decomposição em valores singulares (SVD) para prever as avaliações dos alunos para cursos que eles ainda não haviam avaliado.

- **Tabela Pivô**: Uma tabela foi criada onde os student_id formam as linhas, course_id formam as colunas e os valores são as avaliações dos cursos.

course_id	1	2	3	4	5	6	8	9	10	11	12	13	14
student_id													
7	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	0.0	0.0
24	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0	4.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0
37	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
41	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
46	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0
50	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
55	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0
61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
66	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
73	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
76	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
80	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	0.0	0.0
81	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0
84	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
86	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
90	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0
93	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	0.0	2.0
95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0

Figura 10. Tabela dinâmica (student_id,course_id).

course_id	15
student_id	
7	0.0
9	0.0
10	0.0
11	0.0
16	0.0
21	0.0
24	0.0
26	0.0
32	0.0
33	0.0
34	0.0
37	0.0
41	0.0
44	0.0
45	2.0
46	0.0
48	0.0
50	0.0
55	0.0
61	0.0
66	2.0
71	0.0
73	0.0
76	0.0
80	0.0
81	0.0
84	0.0
86	10.0
90	0.0
93	0.0
95	9.0
100	0.0

Figura 11. Tabela dinâmica (student_id, course_id) (complemento).

- **Matriz esparsa**: a tabela de interações entre alunos e cursos é transformada em uma matriz esparsa, que é uma estrutura de dados eficiente para representar matrizes grandes e esparsas

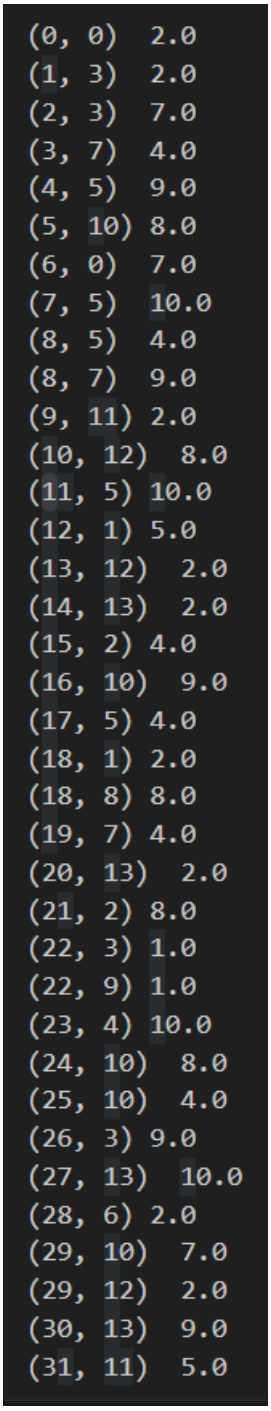


Figura 12. Matriz esparsa.

A primeira linha (0,0) 2.0, indica que a linha 0 e coluna 0 da matriz original, ou seja, (estudante id 7 e curso id 1) a nota foi 2.0, e assim por diante.

- **Decomposição SVD**: A matriz resultante foi decomposta utilizando a técnica de SVD para gerar previsões de avaliação.

course_id	7	9	10	11	16	21
1	2.000000e-01	2.323162e-15	2.322326e-15	2.323496e-15	2.323496e-15	2.323496e-15
2	2.537104e-15	2.350184e-15	2.430966e-15	2.363124e-15	2.407106e-15	2.239079e-15
3	2.238658e-15	2.355045e-15	2.433917e-15	2.594064e-15	2.181833e-15	2.342158e-15
4	2.467910e-15	1.999889e-01	6.999610e-01	2.338815e-15	2.211906e-15	2.335663e-15
5	2.338264e-15	2.295661e-15	2.226072e-15	2.439509e-15	2.308943e-15	2.400835e-15
6	2.265525e-15	2.332656e-15	2.351922e-15	2.133644e-15	9.000000e-01	2.222798e-15
8	2.329047e-15	2.384556e-15	2.537207e-15	2.312033e-15	2.429987e-15	1.975670e-15
9	2.384822e-15	2.230980e-15	1.999080e-15	4.000000e-01	2.143900e-15	2.279702e-15
10	2.371571e-15	2.345009e-15	2.435015e-15	2.306276e-15	2.485547e-15	2.117358e-15
11	2.369397e-15	1.492371e-03	5.223299e-03	2.378874e-15	2.123238e-15	2.232458e-15
12	2.240859e-15	2.323881e-15	2.317534e-15	2.342612e-15	2.069099e-15	8.000000e-01
13	2.367905e-15	2.356802e-15	2.440066e-15	2.311166e-15	2.455126e-15	2.368850e-15
14	2.375309e-15	2.337378e-15	2.437167e-15	2.279403e-15	2.345965e-15	2.282611e-15
15	2.366237e-15	2.256274e-15	2.088233e-15	2.335178e-15	2.473033e-15	2.279893e-15

Figura 13. Resultado da decomposição em valores singulares.

Os resultados normalizados são convertidos para um DataFrame dos pandas, com os índices e colunas correspondendo à tabela dinâmica original (student_id e course_id), os valores são interpretados como previsões de classificação, ou seja, essas previsões indicam o quanto o sistema espera que um aluno goste de um curso com base em dados de interações anteriores. O experimento foi feito com número de características latentes = 10.

- **Recomendações**: Foram feitas recomendações com base nas previsões de avaliação, retornando os cursos com as maiores avaliações previstas.

	course_id	sim
0	12	8.000000e-01
1	5	2.400835e-15
2	13	2.368850e-15
3	3	2.342158e-15
4	4	2.335663e-15

Figura 14. Resulta das recomendações filtragem colaborativa.

Que no caso foram os cursos 12, 5, 13, 3 e 4 recomendadas para o aluno de id 21.

Como resultado, a filtragem colaborativa foi eficaz para capturar padrões de interação aluno-curso, recomendando 5 cursos com base em avaliações passadas de outros usuários com perfis similares.

5.3 Experimento 3 (Filtragem híbrida)

O terceiro experimento combinou as abordagens de filtragem baseada em conteúdo e colaborativa, utilizando um sistema híbrido. O processo seguiu o seguinte fluxo:

- **Baseado em Conteúdo**: Primeiro, foi calculada a similaridade de cosseno entre cursos, retornando os 10 cursos mais semelhantes ao curso de entrada. Onde o primeiro membro é o índice do curso e o segundo é o valor de similaridade

```
[(3, 1.0), (19, 1.0), (8, 0.9138115486202573), (0, 0.0), (1, 0.0), (2, 0.0), (4, 0.0), (5, 0.0), (6, 0.0), (7, 0.0)]
```

Figura 15. Resultado da similaridade dos cossenos na filtragem Híbrida

- **Filtragem Colaborativa (SVD)**: Para os 10 cursos mais semelhantes, foi utilizado o modelo SVD para prever as avaliações que os alunos poderiam atribuir a esses cursos.

	course_id	course_rating	course_duration	study_mode	student_id	est
3	15	2	64	P	66	5
19	9	4	165	P	61	5
8	3	4	166	O	46	5
0	13	5	111	O	100	5
1	2	2	154	P	55	5
2	12	8	79	P	80	5
4	14	2	58	P	44	5
5	1	2	110	O	7	5
6	8	2	128	P	90	5
7	5	10	104	O	76	5

Figura 16. Resultado da decomposição em valores singulares na filtragem Híbrida

- **Recomendações**: Os cursos foram classificados com base nas avaliações previstas, e os melhores cursos foram recomendados.

	course_id	course_rating	course_duration	study_mode	student_id	est
3	15	2	64	P	66	5
19	9	4	165	P	61	5
8	3	4	166	O	46	5
0	13	5	111	O	100	5
1	2	2	154	P	55	5

Figura 17. Resultado das recomendações filtragem híbrida

O algoritmo escolheu as 5 primeiras recomendações para o aluno de id 100 e de semelhança com o curso que foi exigido, que foi do id 13.

5.4 Análises

Os resultados obtidos nos experimentos com os três modelos de recomendação — colaborativo, baseado em item e híbrido — foram analisados com base no RMSE (Root Mean Squared Error), que avalia a precisão das previsões feitas por cada modelo em relação às classificações reais dos alunos.

O modelo híbrido apresentou o menor valor de RMSE, com uma média de 3.06, destacando-se como o mais preciso em termos de previsão. Isso ocorre porque ele combina tanto informações sobre as interações anteriores entre alunos e cursos quanto as características dos próprios cursos. Essa combinação permitiu ao modelo superar as limitações enfrentadas pelos métodos individuais, como o problema de dados esparsos na filtragem colaborativa e a ausência de informações dos alunos no modelo baseado em item.

O modelo de filtragem colaborativa apresentou um RMSE ligeiramente maior (4.47), mas ainda assim competitivo. No entanto, seu desempenho foi inferior ao modelo híbrido, especialmente em cenários onde os dados de interação entre alunos e cursos eram limitados, o que reforça o problema do cold start. A principal vantagem desse modelo é que ele é altamente eficiente quando há dados suficientes sobre as interações dos usuários, mas enfrenta dificuldades quando isso não ocorre.

Por outro lado, o modelo baseado em item apresentou o maior RMSE (6.29), indicando uma menor precisão nas previsões. Isso era esperado, visto que o modelo se baseia apenas nas características dos cursos, ignorando as preferências explícitas dos alunos. Embora tenha a vantagem de não depender das interações dos usuários, sua capacidade de fazer previsões personalizadas é limitada, o que resulta em uma maior discrepância entre as classificações previstas e as reais.

6 Conclusões e Trabalhos Futuros

6.1 Conclusão

Neste trabalho, foram implementados e avaliados três diferentes modelos de sistemas de recomendação aplicados ao contexto educacional: filtragem colaborativa, filtragem baseada em item e um modelo híbrido que combina as duas abordagens. A métrica escolhida para a avaliação dos modelos foi o RMSE (Root Mean Squared Error), que mede a diferença entre as classificações previstas pelos modelos e as classificações reais dos alunos.

Os resultados indicaram que o modelo híbrido apresentou o menor valor de RMSE, indicando maior precisão nas recomendações. Isso era esperado, pois o modelo híbrido aproveita os pontos fortes de ambos os métodos — colaborativo e baseado em conteúdo — ao incorporar informações tanto das interações passadas dos usuários quanto das características dos itens (cursos).

O modelo de filtragem colaborativa, apesar de eficiente, apresentou um RMSE um pouco maior do que o modelo híbrido. Essa abordagem depende exclusivamente das interações entre os usuários e os itens, o que limita a qualidade das recomendações quando há pouca informação disponível sobre as preferências dos usuários (problema conhecido como cold start).

Por fim, o modelo de filtragem baseada em item apresentou o maior RMSE dos três. Isso pode ser atribuído ao fato de que esse método considera apenas as características dos cursos, ignorando completamente as interações entre usuários e itens. Embora seja eficaz quando temos muitos atributos descritivos sobre os cursos, a ausência de informações sobre as preferências dos usuários pode prejudicar a qualidade das recomendações.

Em resumo, o modelo híbrido demonstrou ser o mais robusto, obtendo melhores resultados no contexto de sistemas de recomendação educacionais, seguido pela filtragem colaborativa e, por último, pela filtragem baseada em item. O valor do RMSE como indicador

de erro de previsão reforça a importância de combinar técnicas diferentes para aumentar a precisão das recomendações, especialmente em ambientes com dados limitados ou desbalanceados.

6.2 Trabalhos Futuros

Como visto neste trabalho, é apenas uma ideia simples de como um pequeno sistema de recomendação seria implementado para um sistema educacional online. Uma ideia futura seria de um sistema completo, com front-end interativo que permita aos usuários acessar as recomendações de cursos ou materiais diretamente por meio de uma interface web/mobile amigável. O sistema será integrado a um banco de dados robusto, que armazenará as informações dos cursos, usuários e suas interações (como avaliações e preferências), permitindo a personalização dinâmica das recomendações. A arquitetura proposta incluirá a criação de APIs para a comunicação entre o banco de dados e a camada de apresentação, além de utilizar tecnologias modernas para garantir escalabilidade e desempenho.

7 Referências bibliográficas

[1] Lee, Yena, et al. “Personalized Recommendation System Based on Recommendation Indicator in Education Platform.” Proceedings of the International Conference on Research in Adaptive and Convergent Systems, ACM, 2023, pp. 1–6. DOI.org (Crossref), https://doi.org/10.1145/3599957.3606224.

[2] MAGRANI, Eduardo. O que são Sistemas de Recomendação e qual a sua importância? 2023. Disponível em: https://www.itchannel.pt/news/opiniao/o-que-sao-sistemas-de-recomendacao-e-qual-a-sua-importancia. Acesso em: 25 abr. 2024.

[3] Sharma, Richa, et al. “Machine Learning Algorithms for Building Recommender Systems.” 2019 International Conference on Intelligent Computing and Control Systems (ICCS), IEEE, 2019, pp. 785–90. DOI.org (Crossref), https://doi.org/10.1109/ICCS45141.2019.9065538.

[4] RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha; KANTOR, Paul B. (Eds.). Recommender Systems Handbook. 2. ed. New York: Springer, 2015.

[5] Referência: SU, Xiaoyuan; KHOSHGOFTAAR, Taghi M.. A Survey of Collaborative Filtering Techniques. Advances In Artificial Intelligence, [S.L.], v. 2009, p. 1-19, 27 out. 2009. Hindawi Limited. http://dx.doi.org/10.1155/2009/421425.

[6] ADOMAVICIUS, G.; TUZHILIN, A.. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Ieee Transactions On Knowledge And Data Engineering, [S.L.], v. 17, n. 6, p. 734-749, jun. 2005. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tkde.2005.99.

[7] BURKE, Robin. Hybrid Recommender Systems: Survey and Experiments. User Modeling And User-Adapted Interaction, [S.L.], v. 12, n. 4, p. 331-370, 2002. Springer Science and Business Media LLC. http://dx.doi.org/10.1023/a:1021240730564.

[8] AHUJA, Rishabh; SOLANKI, Arun; NAYYAR, Anand. Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. 2019 9Th International Conference On Cloud Computing, Data Science & Engineering (Confluence), [S.L.], jan. 2019. IEEE. http://dx.doi.org/10.1109/confluence.2019.8776969.

[9] GARCIA, C. A.; FROZZA, R. Sistema de recomendação de produtos utilizando mineração de dados. Tecno-Lógica, v. 17, n. 1, p. 78–90, 20 jul. 2013.

[10] SANTOS, Rafael Vargas Mesquita; LORENÇÃO, Herik Santos. ELABORAÇÃO DE UM SERVIÇO DE RECOMENDAÇÃO HÍBRIDO IMPLANTADO EM WEBSERVICE RESTFUL. Tecnologia da Informação e Comunicação: pesquisas em inovações tecnológicas, [S.L.], p. 146-156, 2021. Editora Científica Digital. http://dx.doi.org/10.37885/211106635.

[11] MAYANG APRILIANTI; MAHENDRA, R.; INDRA BUDI. Implementation of weighted parallel hybrid recommender systems for e-commerce in Indonesia. 1 out. 2016.

[12] VIANA, C. F. Elaboração de um sistema de recomendação híbrido baseado em combinação sequencial. Ifes.edu.br, 2022.

[13] https://pandas.pydata.org/

[14] TOWARDS AI. How, when, and why should you normalize, standardize, rescale your data? Disponível em: https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff. Acesso em: 11 jul. 2024.

[15] BUILT IN. One-hot encoding: Definition, how it works, and how to use it. Disponível em: https://builtin.com/articles/one-hot-encoding#:~:text=Hot%20Encoding%20Definition-,One%20hot%20encoding%20is%20a%20machine%20learning%20technique%20that%20encodes,in%20a%20linear%20regression%20model. Acesso em: 15 jul. 2024.

[16] GOMES, Naomy. The Cosine Similarity and Its Use in Recommendation Systems. Medium, 11 julho 2024. Disponível em: https://naomy-gomes.medium.com/the-cosine-similarity-and-its-use-in-recommendation-systems-cb2ebd811ce1. Acesso em: 11 julho 2024.

[17] PANDITHU, Deepa. Recommender System — User Collaborative Filtering. Medium, 13 fev. 2020. Disponível em: https://medium.com/@deepapandithu/recommender-system-user-collaborative-filtering-37613f0c6a9 . Acesso em: 14 out. 2024.

[18] CHONG, D. Why We Use Sparse Matrices for Recommender Systems. Disponível em: <https://davidcjw.hashnode.dev/why-we-use-sparse-matrices-for-recommender-systems>. Acesso em: 10 set. 2024.

[19] PEREIRA FILHO, João Bosco A.; FILETO, Renato; BARBETTA, Pedro; BITTENCOURT, Guilherme. Explorando decomposição por valor singular em filtragem colaborativa. Florianópolis: Universidade Federal de Santa Catarina, 2008. Disponível em: https://clei.org/proceedings_data/CLEI2009/57406.pdf Acesso em: 09 aug. 2024.

[20] BREESE, J. S.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pp. 43-52, Morgan Kaufmann Publishers Inc., 1998

[21] HERLOCKER, J. et al. Evaluating Collaborative Filtering Recommender Systems,ACM Transaction on Information Systems, v. 22, n. 1, pp. 5-53, 2004.

[22] PYTHON, R. **Split Your Dataset With scikit-learn's train_test_split() – Real Python**. Disponível em: <https://realpython.com/train-test-split-python-data/>.

[23] MARIO MONTALVO GARCÍA. **Leveraging Surprise Library for Recommender Systems in Python**. Disponível em: <https://medium.com/@mariomg85/leveraging-surprise-library-for-recommender-systems-in-python-915e699f7a99>. Acesso em: 30 set. 2024.

Apêndice A – Código de gerador de dados

```python
import pandas as pd
from random import choice, randint

def generate_educational_data(n_courses=15, n_subjects=10, n_teachers=50, n_schools=10, n_students=100, dataset_size=37):


    d = pd.DataFrame(
        {
            'student_id' : [randint(1, n_students) for _ in range(dataset_size)],
            'course_id' : [randint(1, n_courses) for _ in range(dataset_size)],
            'teacher_id' : [randint(1, n_teachers) for _ in range(dataset_size)],
            'school_id' : [randint(1, n_schools) for _ in range(dataset_size)],
            'subject_id' : [randint(1, n_subjects) for _ in range(dataset_size)],
            'course_duration' : [randint(10, 200) for _ in range(dataset_size)],
            'study_mode' : [choice(['P', 'O']) for _ in range(dataset_size)], # P para presencial e O para online
            'course_rating' : [randint(1, 10) for _ in range(dataset_size)]
        }
    ).drop_duplicates()
    return d

d = generate_educational_data()
d.to_csv('educational_data.csv', index=False)
```

Apêndice B – Filtragem baseado em item

```python
import pandas as pd
import numpy as np
from numpy import dot
from numpy.linalg import norm
from sklearn.metrics import mean_squared_error
from math import sqrt

# Função para normalizar os dados
def normalize(data):
    min_val = min(data)
    max_val = max(data)
    return [(x - min_val) / (max_val - min_val) for x in data]

# Função para aplicar One-Hot Encoding (OHE)
def ohe(df, enc_col):
    ohe_df = pd.get_dummies(df[enc_col])
    ohe_df.reset_index(drop=True, inplace=True)
    return pd.concat([df, ohe_df], axis=1)

# Classe para recomendar com base na similaridade de cosseno
class CBRecommend():
    def __init__(self, df):
        self.df = df

    def cosine_sim(self, v1, v2):
        return sum(dot(v1, v2)/(norm(v1)*norm(v2)))

    def recommend(self, course_id, n_rec):
        inputVec = self.df.loc[course_id].values
        self.df['sim'] = self.df.apply(lambda x: self.cosine_sim(inputVec, x.values), axis=1)
        return self.df.nlargest(columns='sim', n=n_rec)
```

```python
# Função para calcular o RMSE
def calculate_rmse(real_ratings, predicted_ratings):
    real_vals = np.array(real_ratings)
    predicted_vals = np.array(predicted_ratings)
    rmse = sqrt(mean_squared_error(real_vals, predicted_vals))
    return rmse

if __name__ == '__main__':
    # Carregar os dados
    df = pd.read_csv('educational_data.csv')

    # Normalizar as colunas de course_duration e course_rating
    df['course_duration_norm'] = normalize(df['course_duration'].values)
    df['course_rating_norm'] = normalize(df['course_rating'].values)

    # Extrair classificações reais dos cursos antes de remover a coluna
    real_ratings = df['course_rating'].loc[df.index.isin([21,24, 25, 26, 27])].values

    # Aplicar One-Hot Encoding nas colunas 'study_mode' e 'subject_id'
    df = ohe(df=df, enc_col='study_mode')
    df = ohe(df=df, enc_col='subject_id')

    # Remover colunas redundantes (exceto 'course_rating' por enquanto)
    cols = ['course_duration', 'study_mode', 'subject_id']
    df.drop(columns=cols, inplace=True)
    df.set_index('course_id', inplace=True)

    # Criar uma cópia do dataframe
    t = df.copy()
```

```python
# Instanciar a classe de recomendação
cbr = CBRecommend(df=t)

# Fazer recomendações para o curso especificado
#curso de indice 21 = id 13
resultado = cbr.recommend(course_id=t.index[21], n_rec=5)
print("Recomendações: \n", resultado.to_string())

# Prever classificações com base na similaridade
predicted_ratings = resultado['sim'].values * 10  # Convertendo a similaridade em classificações (escala de 1 a 10)

# Calcular o RMSE
rmse = calculate_rmse(real_ratings, predicted_ratings)
print("RMSE: ", rmse)
```

Apêndice C – Filtragem colaborativa

```python
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import svds
from sklearn.metrics import mean_squared_error
from math import sqrt

# Função para normalizar as previsões
def normalize(pred_ratings):
    return (pred_ratings - pred_ratings.min()) / (pred_ratings.max() - pred_ratings.min())

# Gerar DataFrame com previsões
def generate_prediction_df(mat, pt_df, n_factors):
    if not 1 <= n_factors < min(mat.shape):
        raise ValueError("Deve ser 1 <= n_factors < min(mat.shape)")

    # Decomposição em valores singulares
    u, s, v = svds(mat, k=n_factors)
    s = np.diag(s)

    # Calcular as previsões
    pred_ratings = np.dot(np.dot(u, s), v)
    pred_ratings = normalize(pred_ratings)

    # Converter para DataFrame
    pred_df = pd.DataFrame(
        pred_ratings,
        columns=pt_df.columns,
        index=list(pt_df.index)
    ).transpose()
    return pred_df
```

```python
# Função para recomendar cursos
def recommend_courses(pred_df, student_id, n_recs):
    student_pred = pred_df[student_id].sort_values(ascending=False).reset_index().rename(columns={student_id: 'sim'})
    rec_df = student_pred.sort_values(by='sim', ascending=False).head(n_recs)
    return rec_df

# Função para calcular o RMSE
def calculate_rmse(real_ratings, predicted_ratings):
    real_vals = np.array(real_ratings)
    predicted_vals = np.array(predicted_ratings)
    rmse = sqrt(mean_squared_error(real_vals, predicted_vals))
    return rmse

if __name__ == '__main__':
    # Importar os dados
    df = pd.read_csv('educational_data.csv')

    # Criar a tabela dinâmica (Pivot Table)
    pt_df = df.pivot_table(
        columns='course_id',
        index='student_id',
        values='course_rating'
    ).fillna(0)


    # Converter para matriz esparsa
    mat = pt_df.values
    mat = csr_matrix(mat)
```

```python
# Gerar as previsões
pred_df = generate_prediction_df(mat, pt_df, 10)

# Extrair as classificações reais do estudante 21
real_ratings = df['course_rating'].loc[df['student_id'] == 21].values

# Extrair as classificações previstas do estudante 21
predicted_ratings = recommend_courses(pred_df, 21, len(real_ratings))['sim'].values * 10  # Multiplicar para escala 1-10

# Calcular o RMSE
rmse = calculate_rmse(real_ratings, predicted_ratings)
print("RMSE: ", rmse)

# Gerar as recomendações
print(recommend_courses(pred_df, 21, 5))
```

Apêndice D – Filtragem Híbrida

```python
import pandas as pd
import numpy as np
from random import randint
from sklearn.metrics.pairwise import cosine_similarity
from surprise import SVD, Reader, Dataset, accuracy
from surprise.model_selection import train_test_split

def hybrid(student_id, course_id, n_recs, df, cosine_sim, svd_model):


    # Ordenar valores de similaridade em ordem decrescente e pegar os 10 resultados principais
    sim = list(enumerate(cosine_sim[int(course_id)]))
    sim = sorted(sim, key=lambda x: x[1], reverse=True)
    sim = sim[0:10]



    # Obter metadados dos cursos
    course_idx = [i[0] for i in sim]
    courses = df.iloc[course_idx][['course_id', 'course_rating', 'course_duration', 'study_mode', 'student_id']]

    # Prever usando o modelo SVD
    courses['est'] = courses.apply(lambda x: svd_model.predict(student_id, x['course_id'], x['course_rating']).est, axis=1)



    # Ordenar previsões em ordem decrescente e retornar os top n_recs
    courses = courses.sort_values('est', ascending=False)
    return courses.head(n_recs)
```

```python
if __name__ == '__main__':
    # Gerar dados educacionais
    df = pd.read_csv('educational_data.csv')

    # Baseado em conteúdo
    rmat = df.pivot_table(
        columns='course_id',
        index='student_id',
        values='course_rating'
    ).fillna(0)

    # Calcular similaridade de cosseno
    cosine_sim = cosine_similarity(rmat, rmat)
    cosine_sim = pd.DataFrame(cosine_sim, index=rmat.index, columns=rmat.index)



    # Filtragem colaborativa
    reader = Reader()
    data = Dataset.load_from_df(df[['student_id', 'course_id', 'course_rating']], reader)

    # Dividir dados em treino e teste
    trainset, testset = train_test_split(data, test_size=0.3, random_state=10)

    # Treinar o modelo SVD
    svd = SVD()
    svd.fit(trainset)
```

```
# Previsões de teste
test_pred = svd.test(testset)

# Obter RMSE
accuracy.rmse(test_pred, verbose=True)

# Gerar recomendações, prestar atenção no dataframe gerado, nem todos os cursos estão lá, por conta do calculo
student_id = df['student_id'].values[0]
course_id = df['course_id'].values[16]
n_recs = 5


print(hybrid(student_id, course_id, n_recs, df, cosine_sim, svd))
```