Universidade Presbiteriana Mackenzie

Faculdade de Computação e Informática – FCI

PROJETO APLICADO III

Sistemas de Recomendação

COMPONENTES DO GRUPO:

Carlos Antonio Batista	- TIA 22021477
Erick Meyer Machado Terceiro	- TIA 22008225
Mauricio Henrique Leal Novakowski	- TIA 22015078
Pedro Costa Dias	- TIA 22010823

São Paulo

2023

Sumário

Universidade Presbiteriana Mackenzie

Faculdade de Computação e Informática – FCI

1 INTRODUÇÃO

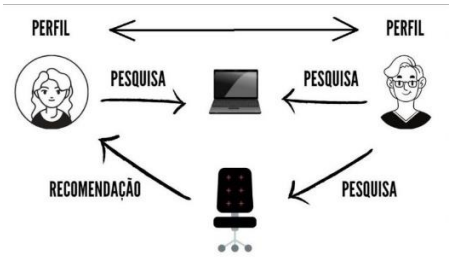
1.1. Contexto do Trabalho

Após várias pesquisas e análises de datasets no Kaggle e no repositório da Universidade da Califórnia, escolhemos o conjunto de dados “Receitas e interações do Food.com” para aplicarmos o Sistema de Recomendação, treinar e testar os métodos mais eficazes para gerar receitas personalizadas a partir do histórico de preferência do usuário (retirado do Kaggle).

Kaggle é uma plataforma para aprendizado de Ciências de Dados que possui inúmeros conjuntos de dados para todo o tipo de treinamento em Data Science e Machine Learning.

O repositório da Universidade da Califórnia possui uma infraestrutura técnica robusta, conectados a mais de 100 mil redes de pesquisa e educação de alto desempenho em todo o mundo. Esse repositório digital leva a segurança muito a sério e são implementadas medidas de proteção dos dados dos clientes.

Sistemas de recomendação são aplicações que conseguem sugerir algo a uma pessoa, com a ajuda de uma predição probabilística de que ele vai gostar daquilo. Envolve uma análise profunda que compreende padrões, correlações entre os dados e até mesmo a distância entre as variáveis existentes na base de dados.



Fonte: https://didatica.tech/sistemas-de-recomendacao/

Link dataset: https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions

1.2. Fonte da base de dados

Forma de coleta	Raspado por meio de solicitações Python/BeautifulSoup
Início da coleta	24/02/2000
Término da coleta	17/12/2018
Origem dos dados	Food.com
Proprietário da base	Shuyang Li
Editor primário da base	Shuyang Li
Editor secundário da base	Bodhisattwa Prasad Majumder
DOI	10.34740/kaggle/dsv/783630
Licença	Arquivo de dados © Autores originais
Frequência de atualização	Atualizado há 4 anos

Fonte: Elaborado pelo próprio autor com dados do Kaggle.

1.3. Motivação - Justificativas

A relevância do tema é, a partir de receitas consumidas pelo usuário anteriormente, atender em nível de técnica e receita, aquela que seja plausível e personalizada para cada tipo de usuário, tornando-o mais feliz e satisfeito além de colaborar com a sua saúde alimentar e física.

Sabemos que uma alimentação saudável é fundamental para o bom funcionamento do organismo. A alimentação saudável aliada a exercícios físicos contribui para a qualidade de vida, melhorando o sistema imunológico, a capacidade de concentração, prevenindo doenças entre outros benefícios.

A má alimentação é uma das principais causas de mortes no mundo. A alimentação inadequada está relacionada ao desenvolvimento de doenças e problemas de saúde como: obesidade e sobrepeso, doenças cardiovasculares, diabetes entre outas.

Nesse contexto, esse projeto pode ajudar todas as pessoas a prepararem uma refeição simples e saudável de acordo com os gostos e preferências individuais, sem excessos e exageros, incentivando a boa prática alimentar, experimentando alimentos diferentes e tornando-os mais receptivos a determinados alimentos que não costumam comer.

1.4. Objetivos

Este projeto tem por objetivo gerar receitas personalizadas para ajudar os usuários com preferências culinárias. Ajudar as pessoas a mudarem o seu comportamento alimentar, desenvolvendo sistemas capazes de recomendar receitas saudáveis e que levam em conta as necessidades e preferências (gostos) dos usuários e, também, a experimentar novos alimentos.

1.5. Cronograma e Diretório

Período:	**Atividade:**	**Agenda:**
01/08/23 a 27/08/23	**Etapa 1** • ☑ ~~Organização dos grupos de trabalho.~~ • ☑ ~~Escolha do tema do projeto.~~ • ☑ ~~Organização do repositório de materiais.~~ • ☑ ~~Cronograma do projeto.~~ • ☑ ~~Seção: Capa~~ • ☑ ~~Seção: Sumário (Parcial)~~ • ☑ ~~Seção: Introdução~~	21/08 Encontro com o Professor. 21/08 Reunião 1 - Levantamento de dúvidas. 23/08 Reunião 2 - Decisões finais e divisão de tarefas. 25/08 Prazo para tarefas individuais. 27/08 Prazo Final para Entrega
28/08/23 a 22/09/23	**Etapa 2** • ☐ Definir as bibliotecas Python. • ☐ Analisar, de forma exploratória, a base de dados. • ☐ Tratar e preparar a base de dados para o treinamento. • ☐ Definir a técnica para o treinamento do modelo de recomendação. • ☐ Realizar o treinamento de um modelo inicial, como prova de conceito. • ☐ Definir a forma de avaliação de desempenho do modelo. • ☐ Descrever o referencial teórico para a elaboração do projeto. • ☐ Seção: Referencial Teórico • ☐ Seção: Sumário (Update)	28/08 Reunião 1 - Levantamento de dúvidas. 05/08 Reunião 2 - Decisões finais e divisão de tarefas. 08/08 Análise Exploratória. 11/08 Treinamento do modelo. 17/08 Prazo para tarefas individuais. 18/08 Encontro com o Professor. 22/09 Prazo Final para Entrega.

23/09/23 a 23/10/23	**Etapa 3** • ☐ Analisar os resultados preliminares alcançados na etapa anterior. • ☐ Ajustar o pipeline de treinamento para melhoria do desempenho do modelo. • ☐ Reavaliar o desempenho do modelo. • ☐ Organizar, de forma sistemática, a descrição das técnicas utilizadas. • ☐ Descrever a metodologia aplicada no projeto. • ☐ Seção: Metodologia • ☐ Seção: Sumário (Update)	25/09 Reunião 1 - Levantamento de dúvidas. 27/09 Reunião 2 - Decisões finais e divisão de tarefas. ??/09 Encontro com o Professor. 20/10 Prazo para tarefas individuais. 23/10 Prazo Final para Entrega
24/10/23 a 1/11/23	**Etapa 4** • ☐ Organizar os resultados alcançados. • ☐ Analisar os resultados, identificando pontos positivos e negativos das técnicas utilizadas. • ☐ Descrever e documentar os resultados. • ☐ Descrever e documentar as conclusões e os trabalhos futuros. • ☐ Entregar a apresentação em vídeo do projeto. • ☐ Entregar os artefatos de software. • ☐ Seção: Resultados • ☐ Seção: Conclusão e trabalhos futuros • ☐ Seção: Resumo • ☐ Seção: Sumário (Update final) • ☐ Entregar a documentação do projeto.	25/10 Reunião - Levantamento de dúvidas, decisões finais e divisão das tarefas. ??/10 Encontro com o Professor. 28/10 Prazo para tarefas individuais. 01/11 Prazo Final para Entrega

Fonte: Elaborado pelo próprio autor.

Link para o diretório: https://github.com/PedroCosDi/ProjetoAplicadoMack3

2 REFERENCIAL TEÓRICO

Projeto de conclusão de curso da Universidade Federal da Fronteira Sul – Campos Chapecó – Ciência da Computação de 2022 sobre sistema de recomendação de receitas alimentares utilizando filtragem baseada em conteúdo.

O projeto trata da dificuldade, nos últimos anos, de se manter uma alimentação saudável devido a fatores como reflexo da pandemia, problemas psicológicos como ansiedade e depressão.

A internet nos traz uma quantidade de receitas alimentares disponíveis e que vem crescendo a cada dia. Por conta disso, o projeto propõe um desenvolvimento de um sistema de recomendação de receitas, onde o usuário transforma o ato de cozinhar em um hobby e a necessidade de manter uma vida mais saudável. O projeto usa a filtragem baseada em conteúdo para fazer a recomendação e que também é feita por meio de cinco receitas por maior similaridade utilizando a distância euclidiana.

A filtragem baseada em conteúdo seleciona itens considerando o conteúdo de cada um e comparando ao conteúdo dos itens previamente avaliados pelo usuário, e que parece ser a mais apropriada para o projeto.

O objetivo do desenvolvimento desse projeto é possibilitar a melhor compreensão dos métodos de extração de dados, ou web scraping, além de tratar e padronizar os dados que poderão ser utilizados como facilitador no desenvolvimento de projetos futuros.

No que se refere aos algoritmos de aprendizado de máquina, são responsáveis por realizar tarefas desejadas de modo que produzam uma saída esperada; algoritmos esses que funcionam por meio de repetição, modo em que as saídas ficam mais precisas a cada repetição feita. Para que isso ocorra, o conjunto de dados foi dividido em treino e teste.

No projeto foi usado o KMeans que é um algoritmo de clusterização, processo de classificação de padrões na forma não supervisionada em grupos ou clusters. Para isso foi necessário determinar o número de clusters e o centro de cada um deles.

Utilizando as bibliotecas existentes para a linguagem de programação Python, a experimentação demonstrou que, no conjunto de dados extraído de um site chamado “Tudo Gostoso”, obteve-se um melhor resultado utilizando o algoritmo KMeans com três clusters.

Também possibilitou a verificação das recomendações usando a distância euclidiana entre os objetos como medida de similaridade.

A ferramenta teve um resultado razoável em relação às recomendações geradas.

Para trabalhos futuros, pensam em gerar um sistema de recomendação mais preciso, com receitas que possuam ingredientes mais específicos e desenvolver um sistema capaz de aperfeiçoar as recomendações, criando perfis, separados por grupos, condizentes ao gosto do usuário.

BEVILACQUA, Gustavo. Sistema de Recomendação de Receitas Alimentares Utilizando Filtragem Baseada em Conteúdo.2022.Trabalho de Conclusão de Curso(Bacharel em Ciências da Computação)- Universidade Federal da Fronteira Sul,Chapecó, SC,2022.Disponível em:https://rd.uffs.edu.br/bitstream/prefix/5761/1/BEVILACQUA.pdf. Acesso em:22/09/2023.

2.1 Descrição do Projeto

2.1.1 Definição da Linguagem de Programação Usada

A linguagem de programação usada no projeto será o Python, linguagem de alto nível, sintaxe relativamente simples e de fácil compreensão. Possui um grande número de bibliotecas, dentre elas usaremos pandas, numpy, seaborn, matplotlib, sklearn, csr_matrix(scipy, sparse), entre outros.

2.1.2 Análise Exploratória

Esse projeto tem como objetivo sistema de recomendação de usuários de “Receitas e Interações do Food.com” para gerar receitas personalizadas, ajudando o usuário com suas preferências culinárias, mudando seu comportamento alimentar levando em consideração as necessidades e preferências e, o mais importante, experimentar novos alimentos.

Para a análise exploratória, que consiste em toda e qualquer operação realizada com os dados com o objetivo de garantir uma base de dados limpa e organizada para facilitar a compreensão dos dados. Foi realizado o tratamento da base de dados com a limpeza dos dados, verificando e

substituindo valores ausentes (dropando esses valores – pois quando encontrado um valor em branco, o nosso modelo preditivo poderá dar erro), contagem de linhas e colunas, exclusão de colunas desnecessárias para o projeto, verificação e tratamento de valores duplicados tanto para o dataset receitas, como para o dataset interações que serão agrupados. Foi verificado os tipos de variáveis de ambos datasets. Depois do agrupamento, será novamente verificado o número de linhas e colunas, removemos a duplicidade de usuários que fizeram rewiew de receitas duas ou mais vezes, verificamos novamente os valores duplicados para que não tenhamos problemas de um usuário avaliar a mesma receita diversas vezes.

Foi removido o “ID_RECEITA” para que a receita seja recomendada pelo “NOME_RECEITA”.

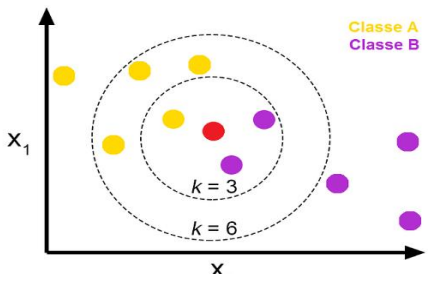
Criamos um novo dataset com as receitas em linhas e os usuários em colunas, fazendo um pivô para que cada “ID_USUARIO” seja uma variável com o respectivo valor de avaliação para cada receita avaliada. Importamos o csr_matrix (pacoteScipy) que é um método para criar a matriz esparsa, que é uma matriz na qual a grande maioria de seus elementos possui um valor padrão, que no caso do nosso projeto, esse valor é o zero. Assim, foi criado e testado o modelo K Vizinhos Mais Próximos (KNN) e as recomendações foram realizadas.

2.1.3 Definição da técnica de treinamento do modelo de recomendação

A técnica de treinamento do modelo de recomendação utilizada nesse projeto será o K Vizinhos Mais Próximos (KNN) modelo bem simples de ser compreendido, apenas é necessário que tenha uma noção de distância e que quanto maior é a proximidade dos pontos, maior é a similaridade entre eles.

Para fazer a recomendação, o filtro colaborativo e o filtro por conteúdo busca por usuários com dados similares e a partir desses dados faz a recomendação de itens ainda não consumidos pelo usuário que está recebendo a recomendação cruzada com os gostos de outro usuário com perfil parecido, com isso se evita que o usuário fique dentro de uma bolha de conteúdos, mas que tem que ser medido para não diversificar demais o conteúdo e o usuário não se identificar com o que foi apresentado.

O KNN faz a predição da vizinhança K do usuário solicitante da recomendação com base em suas classificações positivas ou negativas, para então fazer a previsão das avaliações do usuário.



Fonte: https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e

2.1.4 Treinamento do Modelo

O sistema de filtro colaborativo usa as ações dos usuários para recomendar outras receitas. Já o sistema de filtro por conteúdo usa uma lista de receitas inspirada pelo histórico de navegação do usuário (baseada nas últimas receitas adquiridas pelo usuário).

Logo, o filtro precisa conhecer o domínio em que ele atua de forma que consiga fazer as comparações e retornar receitas semelhantes.

O algoritmo ideal para se utilizar em um sistema de recomendação, são os algoritmos que se calculam e que se baseiam em distância, algoritmo esse que é o K Vizinhos Mais Próximos (KNN) que é um excelente algoritmo quando se deseja fazer modelos preditivos referente a cálculos de distância.

Como cada usuário virou uma variável, para criar o modelo preditivo usaremos o conceito de matriz esparsa, uma matriz com muitos valores zero e que conseguimos compactar essa matriz, e onde houver valores zero, haverá uma função que irá gravar essas posições de zeros.

Para fazer uma conversão para uma matriz esparsa é bem simples, ou seja, no pacote scipy utilizar método csr_matrix e a partir daí já podemos criar o nosso modelo preditivo com uma matriz compactada.

2.1.5 Definição da Forma de Avaliação de Desempenho do Modelo

A análise de grandes quantidades de dados torna-se uma tarefa desafiadora, e para essa tarefa é imprescindível a utilização de ferramentas computacionais que, de forma inteligente, processem as informações dos bancos de dados para auxiliarem na tomada de decisão.

O objetivo desse projeto é buscar o melhor resultado do desempenho do algoritmo K Vizinhos Mais Próximos (KNN) no sistema de recomendação.

Para avaliar o desempenho dos algoritmos é fundamental entender as metodologias de avaliação através da comparação das predições realizadas e as respectivas avaliações reais de usuários para as instâncias preditas.

A obtenção de métricas para aferir o desempenho de um sistema de recomendação é fundamental para verificarmos se as predições realizadas são adequadas para o projeto.

Após importação do algoritmo KNN do Scikit Learn, iremos criar e treinar nosso modelo preditivo. Para isso criamos um objeto chamado “modelo” que receberá o pacote do KNN (NearestNeighbors) usando também um hiperparâmetro chamado “algorithm = brute”, que é um método direto de resolver um problema que depende de poder computacional puro e de tentar todas as possibilidades em vez de técnicas avançadas para melhorar a eficiência.

O “modelo.fit” é o comando para se fazer o treino do modelo preditivo. Ao executarmos, ele criará e treinará nosso modelo.

A partir daí é só realizar as previsões sugerindo ao usuário as receitas. O Kneighbors é o que irá retornar de previsão dos vizinhos mais próximos (que são as recomendações) através da distância e das sugestões.

E por fim, os parâmetros é quem irão ser passados para o nosso modelo fazer as previsões.

3 METODOLGIA DO PROJETO

3.1 Definição do K Nearest Neighbors (KNN) para sistema de recomendação:

O algoritmo K Nearest Neighbors (KNN) pode ser usado em sistema de recomendação, embora seja mais comumente associado a tarefas de classificação e regressão. Quando aplicado em sistema de recomendação é utilizado para encontrar itens ou recomendações similares com base no comportamento de outros usuários ou itens.

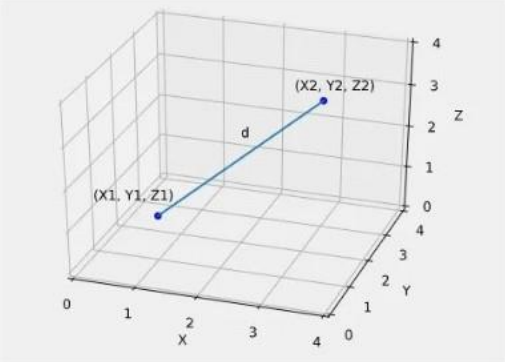
Existem duas abordagens principais para utilizar o KNN em sistemas de recomendação:

1) Sistema de Recomendação de Usuário-Baseado (User-Based) – é usado para encontrar usuários similares com base em seu histórico de interações com itens. Uma vez identificados, os itens preferidos por esses usuários podem ser recomendados ao usuário de destino.

2) Sistema de Recomendação de Item-Baseado (Item-Based) – é usado para encontrar itens similares com base nas preferências dos usuários.

Para implementar o KNN em sistema de recomendação, a métrica de similaridade desempenha um papel fundamental. A similaridade entre usuários ou itens é medida usando métricas de distância como a distância euclidiana, que matematicamente pode ser expressa como:

$$\text{Distância Euclidiana} = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \ldots)}$$



Fonte: https://medium.com/@natalia.gcvs/entenda-o-funcionamento-do-k-nearest-neighbors-knn-616dadad34e

Outra métrica de distância é a correlação de Pearson que mede o quanto duas variáveis mudam juntas dividida pelo produto do quanto elas mudam individualmente. Quanto mais as variáveis mudam juntas em relação ao quanto mudam individualmente, maior a correlação.

Fórmula para cálculo do coeficiente de Pearson entre dois objetos é:

$$Pearson(x, y) = \frac{\Sigma xy - \frac{\Sigma x \Sigma y}{N}}{\sqrt{(\Sigma x^2 - \frac{(\Sigma x)^2}{N})(\Sigma y^2 - \frac{(\Sigma y)^2}{N})}}$$

Fonte: https://uit.br/sicit/images/Documentos/ARTIGOS/APO9.pdf

Outra métrica que podemos citar é a similaridade de cosseno, que é a métrica que calcula o ângulo gerado entre dois vetores com relação à origem, entre outras.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fonte: https://medium.com/data-hackers/deep-learning-para-sistemas-de-recomenda%C3%A7%C3%A3o-parte-3-recomenda%C3%A7%C3%A3o-por-similaridade-d788c126d808

A escolha do valor de “K” é importante nesse contexto. Um valor de “K” maior pode levar a recomendações mais diversificadas, mas também, pode resultar em recomendações menos personalizadas. Um valor de “K” menor tente a produzir recomendações mais focadas, mas pode não levar em consideração informações suficientes.

O KNN em sistemas de recomendação pode ser eficaz em cenário onde a informação de usuário ou item é escassa, uma vez que aproveita informações de comportamento semelhante de outros usuários ou itens. No entanto, ele pode sofrer com problemas da “maldição da dimensionalidade” em conjunto de dados muito grandes, onde a eficiência computacional e a seleção de recursos relevantes se tornam preocupações importantes.

Podemos citar, também, o sistema de recomendação usando o pacote surprise. Surprise é um pacote Python para construir e analisar sistemas de recomendação que lidam com dados de classificação explícitos. Não é uma biblioteca nativa do Python, mas disponibiliza diversos algoritmos prontos para previsão e, além disso, possui ferramentas para avaliar modelos.

O surprise tem implementado vários algoritmos, mas no nosso projeto fizemos a análise com três deles (segue abaixo) e aquele que tiver a melhor performance na avaliação, usaremos no nosso modelo.

1) KNN Baseline – algoritmo básico de filtragem colaborativa que leva em consideração uma classificação básica. A previsão é definida como (segue abaixo) dependendo do user_based campo do sim_options parâmetro.

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - b_{vi})}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

ou

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - b_{uj})}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

Fonte: https://dadosaocubo.com/sistemas-de-recomendacoes-com-surprise/

2) SlopeOne – é uma família de algoritmos usados para a filtragem colaborativa, não trivial, baseada em itens com base em classificações. Um algoritmo simples, mas preciso. Sua simplicidade torna especialmente fácil implementá-los e de forma eficiente, enquanto sua precisão costuma estar no mesmo nível de algoritmos complicados e computacionalmente caros. A previsão dos itens relevantes é definida como:

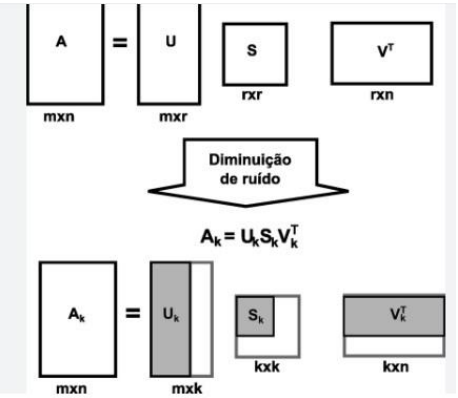
$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} \text{dev}(i, j),$$

Fonte: https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBaseline

3) CoClustering – é um algoritmo de filtragem colaborativa. Também possui implementação simples. Basicamente, usuários e itens recebem alguns clusters e alguns CoClusters. Os clusters são atribuídos usando um método de otimização simples e direta, semelhante ao K-Means. Após isso, os modelos são criados e treinados, bastando somente avaliá-los.

E por fim, podemos avaliar os modelos pela métrica da Raiz do Erro Quadrático Médio (RMSE) e pelo Erro Médio Absoluto (MAE) das previsões e fazer o cross-validate usando o método SVD do pacote surprise.

O SVD (Singular Value Decomposition), ou em português, Decomposições de Valores Singulares, não é uma métrica de avaliação em si, mas uma técnica de decomposição matricial amplamente usada na análise de dados e no processamento de informações.



Fonte: https://www.researchgate.net/figure/Figura-48-A-decomposicao-por-valores-singulares-e-a-posterior-reducao-de-posto-Baseado_fig5_234027762

SVD é um método numérico pertencente ao campo de estudo de Álgebra Linear, utilizado para fatorar matrizes retangulares. É um dos resultados mais importantes da Álgebra Linear, tanto computacional quanto teórico.

O SVD é usado em várias aplicações, incluindo redução de dimensionalidade, compressão de dados, filtragem colaborativa em sistemas de recomendação e análise de componentes principais.

Em métricas de avaliação, como na avaliação de modelos de Machine Learning, o SVD pode ser usado como uma técnica para realizar uma transferência de dimensionalidade nos dados, o que pode ajudar a melhorar a interpretação e desempenho dos modelos. No entanto, o SVD por si não é uma métrica de avaliação.

Em vez disso, métricas de avaliação comuns incluem coisas como precisão, recall, F1_Score, AUC (área sob a curva ROC) e RMSE (Raiz do Erro Quadrático Médio), dependendo do tipo de problema e modelo que está sendo avaliado.

O MAE (Erro Médio Absoluto) é a medida da diferença entre o valor previsto pelo recomendado e o valor real fornecido pelo usuário, ou seja, as avaliações feitas pelos usuários. O MAE mostra o quanto a pontuação prevista está longe da pontuação real. O MAE igual a zero significa que não houve diferença entre a avaliação prevista e real e que o modelo previu com precisão. Portanto, quanto menor o MAE, melhor.

Abaixo, seguem as representações matemáticas do RMSE e do MAE:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (P_i - O_i)^2}{n}} \qquad \text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Fonte: https://ichi.pro/pt/uma-lista-exaustiva-de-metodos-para-avaliar-sistemas-de-recomendacao-183670080086494

3.2 Descrição da Técnica Utilizada

Utilizamos um sistema de recomendação item-baseado em receitas, que é um sistema projetado para recomendar receitas aos usuários com base em suas preferências e histórico de consumo. Portanto, este sistema pode ser útil para ajudar as pessoas a descobrirem novas receitas, planejarem refeições e usarem ingredientes que já possuem em casa e até mesmo provarem novos ingredientes.

Segue abaixo uma visão geral de como esse sistema de recomendação do nosso projeto pode funcionar:

1) Coleta de dados – o sistema coleta dados sobre receitas, ingredientes, avaliações de usuários, preferências alimentares e histórico de consumo.

2) Pré-processamento dos dados – os dados coletados são processados para eliminar ruídos e padronizar as informações. Isso pode incluir a normalização de ingredientes, remoção de recitas duplicadas entre outras.

3) Análise de similaridade – o sistema calcula a similaridade entre receitas com base em ingredientes comuns. Isso é feito usando a métrica de distância euclidiana. A distância euclidiana pode ser considerada uma forma simples e comum de calcular a similaridade entre vetores. Através da aplicação deste cálculo é possível obter a distância geométrica entre dois conjuntos em um plano multidimensional. Como resultado obtém-se um número real e não negativo, sendo que, quanto maior o resultado, maior a distância, ou seja, a lógica é que, descobre-se que um usuário A é parecido com um usuário B. Se o usuário A gostou de uma receita X, então podemos concluir que o usuário B poderá gostar da receita X também. O critério para definir a similaridade neste caso, é definido por um histórico do usuário A e com o cálculo da distância euclidiana com relação ao usuário B, cujas características servem de entrada para o sistema de recomendação. Em uma recomendação por avaliação de receitas, por exemplo, a distância euclidiana é calculada com as avaliações concedidas pelos usuários a cada receita, de modo a chegar no valor de similaridade. Quando um usuário C chega a receber uma recomendação, o sistema se pergunta quem é o usuário mais próximo de C. Se o mais próximo for o usuário B, por exemplo, o sistema recomenda para o usuário C o que o usuário B gostou e assim por diante.

4) Perfil do usuário – o sistema cria um perfil de usuário com base nas suas preferências, histórico de consumo e outras informações relevantes obtidos por meio das avaliações anteriores de receitas, ingredientes que o usuário possui em casa, restrições alimentares e preferências culinárias.

5) Geração de recomendações – com base na análise de similaridade e no perfil do usuário, o sistema gera recomendação de receitas, ou seja, já identifica receitas semelhantes às que o usuário gostou no passado e sugere novas receitas com base no seu perfil.

6) Classificação e filtragem – o sistema classifica as recomendações com base em relevância e aplica filtros, como restrições alimentares ou ingredientes disponíveis em casa.

7) Apresentação das recomendações – são apresentadas ao usuário por meio de interface de usuários como um aplicativo ou site. Os usuários podem visualizar detalhes das receitas, ver ingredientes necessários, passos e tempo de preparo, calorias entre outras.

8) Feedback do usuário – o sistema coleta o feedback do usuário como avaliações e comentários sobre as receitas recomendadas, ajudando a melhorar as recomendações futuras.

9) Aprendizado contínuo – o sistema usa técnicas de Machine Learning para melhorar suas recomendações ao longo do tempo, a medida em que se obtém mais dados de usuários e feedbacks.

No nosso projeto de recomendação de receitas, o primeiro passo foi fazer uma análise exploratória bem robusta, tanto para o dataset receitas, como para o dataset interações. Juntamos ambas as informações e a partir daí, podemos analisar as receitas mais bem avaliadas, a quantidade de avaliações por usuários, quais usuários e qual as avaliações dadas por eles pela(s) receita(s) escolhida(s). Essas são algumas das informações que podem ser interessantes para uma análise.

No primeiro instante, após a análise exploratória, foi implementado o algoritmo do KNN para sistemas de recomendação.

O KNN não faz suposições sobre a distribuição de dados subjacentes, mas depende da semelhança entre os itens. Quando KNN faz inferência sobre uma receita, ele calcula a distância entre a receita de destino e todas as outras receitas em seu banco de dados. Depois classifica suas distâncias e retorna K receitas vizinhas mais próximas como as recomendações mais semelhantes.

Para isso é necessário transformarmos o quadro de dados das classificações em um formato adequado para que possa ser consumido por um modelo KNN, ou seja, uma matriz m x n, onde m é o número de receitas e n é o número de usuários. Para modelar esse quadro com as receitas como linhas e usuários como colunas, preenchendo as observações ausentes com zeros, transformando os valores do quadro de dados numa matriz esparsa.

Após a criação da matriz esparsa, foi importado do Scikit Learn do KNN para criação do modelo preditivo e a partir daí podermos fazer as previsões de receitas baseadas em receitas já utilizadas pelo usuário. Fizemos, também, alguns testes, usando algumas receitas do banco de dados como itens para recomendações de novas receitas.

Quanto às recomendações com o surprise, para as configurações dos modelos, começamos com a função ”reader”, onde vamos informar a nossa escala de avaliação que varia de 0 a 5.

Para preparar os dados para os modelos, utilizamos a função “Dataset.load_from_df”, onde definimos as colunas do dataset ratings que são as avaliações das receitas que serão utilizadas. E, a função “train_test_split” para dividir os dados em treino e teste e, assim conseguirmos avaliar os modelos com dados não vistos no treinamento. Utilizamos 30% dos dados para teste e 70% dos dados para treino.

Para finalizar as configurações, definimos como irão se comportar as medidas de similaridade. Definimos o nome da semelhança que utilizamos, a “pearson_baseline” que leva em conta a Correlação de Pearson para fazer a similaridade. A Correlação de Pearson é um teste que mede a relação estatística entre duas variáveis contínuas, e pode ter um intervalo de valores +1 a -1. O valor zero indica que não há associação entre essas duas variáveis. Um valor maior que zero, indica associação positiva, ou seja, a medida em que o valor de uma variável aumenta, o mesmo acontece com a outra variável. Já um valor menor que zero, indica associação negativa, ou seja, a medida em que o valor de uma variável aumenta, o valor da outra variável diminui.

Após termos os dados de treino e teste, vamos treinar e criar nossos modelos. Para criarmos o modelo é bem simples. Instalamos o modelo em uma variável e passamos os parâmetros desejados. Com o modelo criado, fizemos o fit (treinamento) dos três modelos citados acima (KNN Baseline, SlopeOne e CoClustering).

A métrica que escolhemos para avaliar os modelos é a Raiz do Erro Quadrático Médio (RMSE) das previsões, definida na função “accuracy.rmse”. Não podemos nos esquecer que, quanto menor o erro, melhor a qualidade da solução, ou seja, quanto mais próximo de zero, menor o erro do modelo. Outra métrica é a MAE (Erro Médio Absoluto) das previsões, que é calculado a partir da média dos erros absolutos, ou seja, utilizamos o modo de erro para evitar a subestimação. Isso porque, o valor é menos afetado por pontos, especialmente os extremos (outliers).

Paraisso, foi necessário instalar o surprise, importar bibliotecas necessárias, carregar os dados, usar a função train_test_split e validação cruzada, dividir os dados para treino (70%) e dados para teste (30%). A validação cruzada no surprise é bem simples de realizar. Usamos a função cross_validate, onde o algoritmo SVD é avaliado cruzadamente em 5 partições, e as métricas de erro e erro absoluto são calculadas. Essas etapas nos permitiram fazer as divisões de treinamento e teste, bem como a validação cruzada com a biblioteca surprise para avaliar o desempenho dos algoritmos de recomendação.

3.3 Resultados Preliminares

Resultados preliminares é tudo aquilo que antecede o assunto ou o objeto principal e serve para esclarecer ou para facilitar a sua compreensão. Logo, neste projeto, embora em andamento, é possível constatar que o sistema de recomendação de recitas por item-baseado é uma ferramenta que ajuda os usuários a encontrarem receitas com base em seus gostos pessoais e histórico de consumo. Ele utiliza a análise de similaridade e o perfil do usuário para gerar recomendações relevantes, tornando a experiência culinária mais agradável e diversificada.

Como resultados preliminares, podemos dizer que o nosso sistema de recomendações de receitas está funcionando pelo modelo do KNN usando a matriz esparsa.

Apesar de alguns percalços, como uma base de dados gigantesca como essa, no qual a base de dados original das receitas possui 267783 linhas e 12 colunas e, a base de interações possuem 1048576 linhas e 4 colunas, não estávamos conseguindo implementar o K-Means. Por outro lado, tivemos que abrir mão do mapa de calor referente as avaliações feitas pelos usuários, do próprio K-Means, pois o consumo de memória RAM do Colab chegou ao seu limite. Tivemos que reduzir bastante a base de dados e utilizar o surprise para apresentarmos métricas e avaliações. Mesmo assim, tivemos problemas ao ler os códigos da criação e treinamento do SlopeOne, o qual ainda desejamos avaliar na próxima etapa do projeto.

Abaixo, seguem os resultados dos treinamentos dos modelos:

KNN – matriz esparsa – o usuário escolhe uma determinada receita, como por exemplo: ”lemonade made with stevia” , em seguida é sugerida algumas outras receitas. Foram realizados testes e obtivemos resultados.

KNN Baseline		CoClustering	
RMSE	0,8672	RMSE	0,8954
accuracy	0,867215...	accuracy	0,89538..

Cross validate

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8019  0.8063  0.8065  0.8019  0.7972  0.8028  0.0034
MAE (testset)     0.4742  0.4742  0.4754  0.4746  0.4729  0.4743  0.0008
Fit time          13.85   13.98   14.40   14.21   14.13   14.11   0.19
Test time         1.47    1.42    0.81    0.79    0.78    1.05    0.32
{'test_rmse': array([0.80190759, 0.80630023, 0.80648495, 0.8018823 , 0.79719194]),
 'test_mae': array([0.47424578, 0.47419884, 0.47544744, 0.47459645, 0.47287267]),
 'fit_time': (13.847593307495117,
  13.977767944335938,
  14.403170347213745,
  14.207783937454224,
  14.130545377731323),
 'test_time': (1.468259572982788,
  1.4157934188842773,
  0.807732105255127,
  0.7853982448577881,
  0.7766664028167725)}
```