Documentação do Sistema de Recomendação com SQLite e Algoritmos KNN/SVD

# Visão Geral

Este sistema de recomendação de produtos locais foi reimplementado com as seguintes melhorias:

1. \*\*Banco de Dados SQLite\*\*: Substituição dos arquivos CSV por um banco de dados relacional SQLite

2. \*\*Algoritmos de Machine Learning\*\*: Implementação de dois algoritmos de recomendação

- KNN (k-Nearest Neighbors)

- SVD (Singular Value Decomposition)

3. \*\*Interface Interativa\*\*: Adição de um botão para alternar entre os algoritmos em tempo real

O sistema conecta consumidores a pequenos produtores rurais, considerando preferências, sazonalidade, proximidade geográfica e avaliações de outros usuários.

# Arquitetura do Sistema

## Banco de Dados SQLite

O sistema utiliza SQLite como banco de dados relacional, com as seguintes tabelas:

* \*\*usuarios\*\*: Armazena informações dos usuários, preferências e localização
* \*\*associacoes\*\*: Armazena dados das associações de produtores e suas localizações
* \*\*produtos\*\*: Armazena informações dos produtos, incluindo preço, categoria e sazonalidade
* \*\*avaliacoes\*\*: Armazena avaliações dos usuários para produtos
* \*\*compras\*\*: Armazena histórico de compras dos usuários
* \*\*Tabelas de associação\*\*: Para relacionamentos muitos-para-muitos

## Algoritmos de Recomendação

### KNN (k-Nearest Neighbors)

O algoritmo KNN recomenda produtos com base em usuários com preferências semelhantes:

1. Constrói uma matriz usuário-item com avaliações

2. Calcula a similaridade entre usuários usando distância de cosseno

3. Identifica os k vizinhos mais próximos (usuários com gostos similares)

4. Recomenda produtos que esses vizinhos avaliaram bem e que o usuário atual ainda não avaliou

5. Ajusta as recomendações considerando distância geográfica e sazonalidade

### SVD (Singular Value Decomposition)

O algoritmo SVD utiliza decomposição matricial para identificar padrões ocultos:

1. Constrói uma matriz usuário-item com avaliações

2. Decompõe a matriz em três componentes (U, Σ, V^T)

3. Reduz a dimensionalidade mantendo apenas os fatores mais importantes

4. Reconstrói a matriz para prever avaliações desconhecidas

5. Ajusta as recomendações considerando distância geográfica e sazonalidade

## Backend (Flask)

O backend foi implementado com Flask e oferece as seguintes APIs:

* `/api/mapa`: Retorna dados das associações para exibição no mapa
* `/api/produtos`: Retorna produtos filtrados por associação, sazonalidade e tipo
* `/api/avaliar`: Permite adicionar avaliações a produtos
* `/api/recomendacoes`: Retorna recomendações personalizadas usando o algoritmo selecionado
* `/api/alternar\_algoritmo`: Permite alternar entre os algoritmos KNN e SVD
* `/api/algoritmo`: Retorna informações sobre o algoritmo atualmente selecionado

## Frontend

A interface web permite:

* Visualizar associações em um mapa interativo
* Filtrar produtos por mês, tipo (orgânico) e associação
* Avaliar produtos com pontuação de 1 a 5 estrelas e comentários
* Alternar entre os algoritmos KNN e SVD com um botão
* Ver recomendações personalizadas baseadas no algoritmo selecionado

# Instalação e Execução

## Requisitos

* Python 3.8 ou superior
* Bibliotecas: Flask, SQLAlchemy, scikit-learn, scipy, pandas, geopy, folium

## Instalação

1. Clone o repositório ou extraia o arquivo zip

2. Instale as dependências:

```

pip install -r requirements.txt

```

## Execução

1. Execute o servidor Flask:

```

python -m src.main

```

2. Acesse no navegador: `http://localhost:5000`

# Uso do Sistema

## Alternando entre Algoritmos

Na parte superior da página, você encontrará um botão de alternância para escolher entre:

* \*\*KNN (K-Nearest Neighbors)\*\*: Recomenda produtos com base em usuários com preferências semelhantes
* \*\*SVD (Singular Value Decomposition)\*\*: Utiliza decomposição matricial para identificar padrões ocultos

Ao alternar o algoritmo, as recomendações são atualizadas automaticamente.

## Filtrando Produtos

Você pode filtrar produtos por:

* \*\*Mês\*\*: Para considerar a sazonalidade dos produtos
* \*\*Tipo\*\*: Apenas produtos orgânicos
* \*\*Associação\*\*: Produtos de uma associação específica

## Avaliando Produtos

Para avaliar um produto:

1. Clique no botão "Avaliar" no card do produto

2. Selecione uma pontuação de 1 a 5 estrelas

3. Adicione um comentário (opcional)

4. Clique em "Enviar Avaliação"

As avaliações são persistidas no banco de dados e afetam as recomendações futuras.

## Atualizando sua Localização

Você pode atualizar sua localização para receber recomendações baseadas na proximidade:

1. Insira as coordenadas de latitude e longitude

2. Clique em "Atualizar Localização"

# Detalhes Técnicos

## Estrutura de Arquivos

projeto\_sqlite/

├── data/ # Banco de dados SQLite

├── src/

│ ├── models/ # Modelos de dados

│ ├── routes/ # Rotas da API

│ ├── templates/ # Templates HTML

│ ├── static/ # Arquivos estáticos (CSS, JS)

│ └── main.py # Ponto de entrada da aplicação

├── models.py # Definições ORM para SQLAlchemy

├── recommenders.py # Implementação dos algoritmos KNN e SVD

└── requirements.txt # Dependências do projeto

## Modelo de Dados

O sistema utiliza SQLAlchemy ORM com as seguintes classes:

* `Usuario`: Representa um usuário do sistema
* `Associacao`: Representa uma associação de produtores
* `Produto`: Representa um produto disponível
* `Avaliacao`: Representa uma avaliação de um produto por um usuário
* `Compra`: Representa uma compra realizada por um usuário

## Algoritmos de Recomendação

### Classe Base `RecommenderSystem`

Implementa funcionalidades comuns a todos os recomendadores:

* Preparação de dados
* Construção da matriz usuário-item
* Filtragem por distância e sazonalidade

### Classe `KNNRecommender`

Implementa o algoritmo KNN:

* Treinamento do modelo usando scikit-learn
* Recomendação baseada em vizinhos mais próximos
* Fallback para recomendações baseadas em popularidade

### Classe `SVDRecommender`

Implementa o algoritmo SVD:

* Decomposição da matriz usando scipy.sparse.linalg.svds
* Reconstrução da matriz para prever avaliações
* Ajuste de recomendações com base em fatores latentes

# Comparação dos Algoritmos

## KNN (k-Nearest Neighbors)

\*\*Vantagens:\*\*

* Intuitivo e fácil de entender
* Bom para usuários com perfis bem definidos
* Recomendações explicáveis (baseadas em usuários similares)

\*\*Desvantagens:\*\*

* Pode ter desempenho ruim com dados esparsos
* Computacionalmente intensivo para grandes conjuntos de dados
* Requer um número mínimo de avaliações por usuário

## SVD (Singular Value Decomposition)

\*\*Vantagens:\*\*

* Lida bem com dados esparsos
* Captura padrões latentes não óbvios
* Geralmente oferece recomendações mais precisas

\*\*Desvantagens:\*\*

* Menos intuitivo e mais difícil de explicar
* Pode sofrer de overfitting
* Requer ajuste cuidadoso de hiperparâmetros

# Melhorias Futuras

Algumas possíveis melhorias para o sistema:

1. \*\*Autenticação de usuários\*\*: Implementar sistema de login para personalização completa

2. \*\*Mais algoritmos\*\*: Adicionar outros algoritmos como Matrix Factorization ou modelos híbridos

3. \*\*Explicabilidade\*\*: Adicionar explicações sobre por que um produto foi recomendado

4. \*\*Avaliação offline\*\*: Implementar métricas de avaliação como RMSE, precisão e recall

5. \*\*Processamento em lote\*\*: Pré-calcular recomendações periodicamente para melhorar desempenho

6. \*\*Interface móvel\*\*: Otimizar a interface para dispositivos móveis

7. \*\*Integração com pagamentos\*\*: Permitir a compra direta de produtos recomendados

# Conclusão

Este sistema demonstra como técnicas de machine learning podem ser aplicadas para conectar produtores rurais e consumidores, promovendo o consumo de produtos locais e frescos. A combinação de banco de dados relacional com algoritmos de recomendação avançados permite uma experiência personalizada e relevante para os usuários.