Projeto 1 - Recomendação de produtores

Introdução à Inteligência Artifical

Rafael Dias Ghiorzi
232006144
Depart. de Ciência da Computação
Universidade de Brasília
Brasília, Brasil

I. Introdução

O presente relatório tem como objetivo detalhar o desenvolvimento do projeto de recomendação de produtores locais da matéria de introdução a inteligência artificial.

O projeto consiste na construção de uma aplicação capaz de filtrar e recomendar agricultores do Distrito Federal utilizando técnicas de filtragem comum e recomendação que utilizam algoritmos de *Machine Learning* fundamentais.

Para a realização do projeto, foi utilizado uma base de dados reais disponibilizados em uma iniciativa do Programa Nacional de Alimentação Escolar (PNAE) em conjunto com dados fictícios criados por modelos de linguagem de larga escala. Com base nesses dados, foi possível construir uma aplicação que serve o propósito hipotético de recomendar produtores para usuários da plataforma.

II. Desenvolvimento

O objetivo do projeto foi atender os seguintes requerimentos do roteiro: aplicação de filtros de distância, preferência, sazonalidade e aplicação de recomendações baseadas em filtragem colaborativa e uso de modelos de aprendizagem de máquina.

Para isso, foi utilizado um ambiente Python e algumas dependências necessárias. As bibliotecas utilizadas foram:

- Pandas para manipulação de tabelas e DataFrames;
- SQLAlchemy para estruturação de dados e organização de buscas;
- SciKit-Learn para aprendizagem de máquina;
- Streamlit para construção de uma interface simples e responsiva;
- Faker para criação de usuários fictícios
- Geopy para calcular distâncias geográficas

O código fonte do projeto pode ser encontrado no seguinte link do repositório do GitHub. Cada função e filtro será abordado de forma mais aprofundada nas seções a seguir.

A. Obtenção dos dados

O projeto tinha como requisito encontrar um dataset que contesse informações acerca dos produtores e seus produtos ofertados no Distrito Federal. O professor, no roteiro, disponibilizou uma lista de produtores junto com a fonte dessa lista. Nessa fonte, havia links para editais relacionados à compra e planejamento de compra de certos produtos. Os links seguiam para uma chamada pública de compras e o resultado desse processo público.

Tendo acesso a essas informações, foi utilizado o modelo de LLM mais recente da Google, *Gemini 2.5 Pro* para analisar os textos e tabelas presentes nos documentos e criar um arquivo CSV contendo todas as informações para cada produtor de cada região administrativa, com o objetivo de ser utilizado como base para o projeto. O arquivo de dados pode ser encontrado dentro do repositório do projeto no GitHub.

B. Filtro por distância

O filtro por distância presente na aplicação é uma simples aplicação de distância geográfica utilizando a biblioteca Geopy. O código estruturado fica da seguinte forma:

```python
# ================ funções de filtros ================
def filtro_distancia(user_lat: float, user_lon: float, raio: float) -> list[Produtor]:
    """Retorna os produtores dentro de um raio especificado a partir da localização do usuário."""
    with Session(engine) as session:
        # Busca todos os produtores
        produtores = session.query(Produtor).all()

        # Calcula a distância entre o usuário e cada produtor
        resultado = []
        for produtor in produtores:
            if produtor.lat is not None and produtor.lon is not None:
                # Calcula a distância usando geopy
                distancia = geodesic(
                    (user_lat, user_lon),
                    (produtor.lat, produtor.lon)
                ).km
                # Filtra a distancia para ver se adiciona nos resultados
                if distancia <= raio:
                    resultado.append(produtor)

        # Retorna os produtores dentro do raio ordenados pela distância
        return resultado
```

Fig. 1. Função de filtro de distância

Essa função simplesmente utiliza a latitude e longitude do usuário para calcular a distância entre ele e cada produtor presente no banco de dados, filtrando aqueles que possuem uma distância maior que o raio selecionado.

C. Filtro por preferência

O filtro por preferência recebe o nome dos produtos que o usuário deseja, e retorna os produtores que contém, no mínimo, os produtos escolhidos utilizando querys SQL e filtragem simples. Segue a seguir o código da função:

```python
def filtro_preferencia(preferencia: list) -> list[Produtor]:
    """Retorna os produtores que oferecem todos os produtos da preferência."""
    with Session(engine) as session:
        if not preferencia:
            return []

        # Busca todos os ids dos produtos da preferência
        produtos_id = session.query(Produto.id).filter(
            Produto.nome.in_(preferencia)).all()
        produtos_id = [produto[0] for produto in produtos_id]

        if not produtos_id:
            return []

        from db import produtor_produto
        # Busca os produtores que oferecem todos os produtos da preferência
        produtores = session.query(Produtor).join(
            produtor_produto).filter(
                produtor_produto.c.produto_id.in_(produtos_id)).group_by(
                    Produtor.id).having(
                        func.count(
                            produtor_produto.c.produto_id
                            ) >= len(produtos_id)).all()

        return produtores
```

Fig. 2. Função de preferência

D. Filtro por sazonalidade

O filtro por sazonalidade retorna os produtores que possuem produtos disponíveis na estação do ano atual. Para isso, foi criado outro arquivo CSV com as sazonalidades dos produtos e outras características manualmente, assim como uma função que retorna a estação do ano. Com essa coleta, as funções ficaram da seguinte forma:

```python
def get_estacao():
    mes = time.localtime().tm_mon
    estacao = {
        1: "Verão", 2: "Verão",
        3: "Outono", 4: "Outono",
        5: "Outono", 6: "Inverno",
        7: "Inverno", 8: "Inverno",
        9: "Primavera", 10: "Primavera",
        11: "Primavera", 12: "Verão"
    }
    return estacao[mes]
```

```python
def filtro_sazonalidade() -> list[Produtor]:
    """Retorna os produtos disponíveis na estação atual."""
    estacao = get_estacao()

    with Session(engine) as session:
        # Busca os produtos disponíveis na estação atual
        produtos = session.query(Produto).filter(
            Produto.sazonalidade == estacao).all()

        if not produtos:
            return []
        produtos_id = [produto.id for produto in produtos]

        from db import produtor_produto
        produtores = session.query(Produtor).join(
            produtor_produto).filter(
                produtor_produto.c.produto_id.in_(produtos_id)).distinct().all()

        return produtores
```

Fig. 3. Função de sazonalidade

E. Recomendação utilizando aprendizagem de máquina

A recomendação com aprendizagem de máquina é a função mais complexa e importante do projeto. A função utiliza o algoritmo KNN para recomendar produtores para o usuário da seguinte forma:

Primeiro, é criado uma matriz pivotada em que cada linha é um usuário, cada coluna é um produtor e os valores são as notas fictícias dadas para os produtores pelos usuários. Essa matriz é utilizada para treinar o modelo KNN utilizando similaridade de cossenos e 5 como o K da função.

A partir desse modelo, encontramos os n-ésimos vizinhos do usuário e os produtores avaliados por eles. Comparamos os produtores avaliados com nota acima de 3 pelos vizinhos e recomendamos a diferença entre os produtores avaliados pelo usuário, ou seja, os produtores que os vizinhos avaliaram bem, mas que o usuário nunca avaliou.

Essa função retorna diferentes produtores baseado nas avaliações do próprio usuário e pode ser usado a retornar diversos produtores baseado tanto na atividade do usuário quanto dos seus vizinhos, ou seja, é uma recomendação dinâmica, diferente das outras filtragens presentes no projeto.

Segue o código completo da função de recomendação baseada em KNN:

```python
def recomendar_produtores():
    """Cria ou treina um modelo KNN para recomendações de produtores."""
    with Session(engine) as session:
        # Busca as informações necessárias
        query = session.query(Avaliacao.usuario_id, Avaliacao.produtor_id, Avaliacao.nota)
        dataframe = pd.read_sql(query.statement, session.connection())

    # Cada linha dessse dataframe é uma avaliação
    # Colunas usuario_id, produtor_id, nota
    matriz = dataframe.pivot_table(
        index='usuario_id',
        columns='produtor_id',
        values='nota',
        fill_value=0
    )
    matriz.fillna(0)
    # cada coluna é um produtor, cada linha é um usuario

    # Criar e treinar o modelo KNN
    modelo = NearestNeighbors(metric='cosine', n_neighbors=5, algorithm='brute')
    modelo.fit(matriz.values)

    # Criar um dicionario com os ids na ordem que aparecem na matriz
    usuarios_idx = {id: i for i, id in enumerate(matriz.index.tolist())}

    idx_usuario = usuarios_idx[usuario.id]
    # criar uma lista de avaliacoes do usuario
    # essa lista mostra o valor da nota para cada produtor na ordem das colunas
    # da matriz pivotada
    avaliacoes_usuario = matriz.iloc[idx_usuario].to_numpy().reshape(1, -1)

    # knn retorna a distancia dos vizinhos [[]] e o indice deles na matriz [[]]
    _ , vizinhos_k = modelo.kneighbors(avaliacoes_usuario, n_neighbors=20)

    # inverte o usuarios_idx para encontrar o usuario com base no indice
    usuarios_idx = {value: key for key, value in usuarios_idx.items()}

    vizinhos = []
    # encontrar os vizinhos para colocar na lista
    for i in range(1, len(vizinhos_k[0])):
        idx = vizinhos_k[0][i]
        id = usuarios_idx.get(idx)

        if id is not None:
            vizinhos.append(id)
```

Fig. 4. Primeira parte da função

```python
# ====================================================
# Com a lista de vizinhos ordenada e feita, encontrar
# os vizinhos, seus produtores avaliados e pegar os que
# o usuário não avaliou e tem uma nota boa para recomendar

# encontrar os produtores dos vizinhos
produtores = []
with Session(engine) as session:
    for id in vizinhos:
        avaliacoes = session.query(
            Avaliacao.produtor_id).filter(
                Avaliacao.usuario_id == id).all()
        # Adicionar os ids na lista de produtores
        produtores.extend(aval[0] for aval in avaliacoes)

    # pegar os produtores avaliados pelo usuário
    produtores_usuario = session.query(
        Avaliacao.produtor_id).filter(
            Avaliacao.usuario_id == usuario.id).all()
    produtores_usuario = [produtor[0] for produtor in produtores_usuario]

    produtores_finais = [produtor for
                         produtor in produtores if
                         produtor not in produtores_usuario]
    # remover duplicados
    produtores_finais = list(set(produtores_finais))

    recomendacoes = session.query(Produtor).filter(
        Produtor.id.in_(produtores_finais)).all()
    # pegar apenas os com nota maior que 3
    produtores_finais = []
    for produtor in recomendacoes:
        avg_nota = session.query(func.avg(Avaliacao.nota)).filter(
            Avaliacao.produtor_id == produtor.id).scalar()
        if avg_nota is not None and avg_nota >= 3:
            produtores_finais.append(produtor)

    print(len(produtores_finais), "produtores recomendados")
    return produtores_finais[:10]  # Retorna os 10 primeiros produtores recomendados
```

Fig. 5. Segunda parte da função

F. Interface gráfica

A interface do projeto foi implementada diretamente em Python, utilizando o framework Streamlit, que é uma forma moderna de implementar projetos relacionados a ciência de dados, mas também pode ser utilizado para projetos mais simples como o explicado nesse relatório. A interface é composta por 3 páginas: a página de filtragem simples, a página de recomendações utilizando aprendizagem de máquina e uma página para ver as informações de cada produtor (suas avaliações e produtos ofertados).

O código fonte completo da interface pode ser encontrado dentro do repositório do projeto.

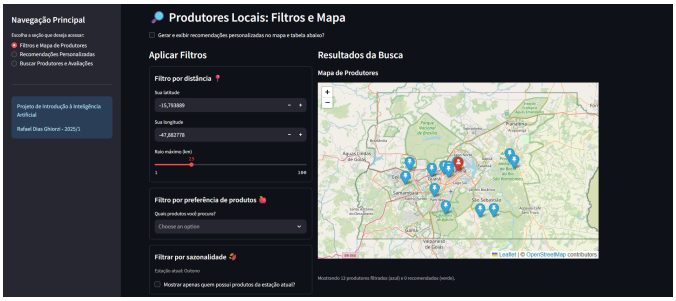

Fig. 6. Página inicial da interface

III. Conclusão

Para a realização desse projeto, foi utilizado conceitos fundamentais estudados até o momento na disciplina. Foi possível colocar em prática algoritmos de filtragem simples e algoritmos de aprendizagem de máquina para recomendação personalizada, analisando suas complexidades e compreendendo em quais contextos cada tipo de função é apropriada.

Além disso, tivemos a oportunidade de trabalhar com dados reais e a criação de dados fictícios que simulam a vida real, explorando conceitos de aplicações utilizadas pelo mundo inteiro. Recomendações personalizadas utilizando aprendizagem de máqina podem ser encontradas em todo lugar, e muitas delas acabam sendo extremamente similares à função aplicada nesse projeto.

Assim, esse projeto proporcionou uma valiosa experiência prática, demonstrando como os conceitos teóricos aprendidos podem ser aplicados a problemais reais e relevantes, como a recomendação de produtores locais para usuários de uma plataforma social.