Documentação - Recommender System

Um código de Rebeca Bivar e Mathews Alves.

Classe BDManagement

connect()

Função: Conectar com o servidor do postgres.

Input: Dicionário com informações necessárias para a conexão.

Output: Conector do postgres.

getOutputRecom()

Função: Retornar a tabela de recomendação de clientes do banco.

Input: -

Output: Tabela de recomendações de produtos para clientes.

getSalesTable()

Função: Retornar a tabela de vendas do banco.

Input: -

Output: Tabela de vendas.

getProductsTable()

Função: Retornar a tabela de produtos do banco.

Input: -

Output: Tabela de produtos.

getClientRecomTable()

Função: Retornar as colunas necessárias da tabela de vendas do banco para a recomendação utilizando TuriCreate.

Input: -

Output: Tabela utilizada para treino do modelo de clientes pelo TuriCreate.

updateProductTable()

Função: Apagar a tabela de produtos e depois atualiza com as novas informações de produtos obtidas.

Input: DataFrame de produtos a ser inserida no banco.

Output: -

updateRecomTable()

Função: Apagar a tabela de recomendações e depois atualiza com as novas informações de recomendações para clientes obtidas.

Input: DataFrame de recomendação de clientes a ser inserida no banco.

Output: -

Classe DMean

get_classif_dict()

Função: Retornar um dicionário de classificações de produtos.

Input: DataFrame com as vendas realizadas.

Output: Dicionário de classificações de produtos.

get_d_mean_classif()

Função: Calcular o valor d_mean para uma dada classificação.

Input: Classificação a ser calculada, DataFrame agrupado por classificação e código do cliente, com a contagem de compras.

Output: Valor d_mean para uma dada classificação.

create_class_cliente_df()

Função: Para cada cliente que realizou compra de item de uma das classificações, fazer o somatório de itens comprados e do número de compras do cliente. Aplicar também o cálculo do d_mean para cada classificação.

Input: DataFrame com as vendas realizadas.

Output: DataFrame agrupado e com os cálculos d_mean.

create_classif_dict()

Função: Criar o dicionário de classificações que indica se as classificações devem ter produtos de mesma classificação recomendados ou não, a partir de um *treshold* calculado em cima do comportamento de compras de produtos de mesma classificação.

Input: DataFrame com as vendas realizadas.

Output: Dicionário de classificações.

Classe ExtractDescription

create_df_product()

Função: Extrair descrições completas de cada produto do DataFrame de compras, e atualizar a tabela de produto, já com as descrições, no banco de dados.

Input: DataFrame de compras.

Output: -

Classe CartRecom

Possui uma única instância criada durante a execução do arquivo "**api.py**", possui o papel de uma interface entre as requisições de recomendações para o carrinho e para o treinamento do mesmo. Além disso, essa instância tem acesso as duas outras classes, a **SimilarityModel** e a **DMean**.

__ init __()

Função: Carregar os arquivos "**pickle/cart_convert_produto.pickle**" e "**pickle/cart_output.pickle** para suas respectivas variáveis.

Input: -

Output: -

get_products_to_recommend()

Função: Recomendar códigos de produtos baseado no input.

Input: - **code** (código do produto inserido no carrinho).

Output: Array com código de produtos recomendados.

create_cart_recommendation_output()

Função: Faz o treinamento da recomendação do carrinho para cada um dos produtos presentes na tabela **df_products** e salva os resultados em dois arquivos, complementares, no formato .pickle.

Input: - **df_compras** (tabela de todas as vendas presentes no banco de dados), **df_products** (tabela de todos os produtos presentes no banco de dados).

Output: -

create_matrix_u_c()

Função: Criar matriz esparsa de classificação por cliente, contendo os valores 0 ou 1, onde 1 corresponde que cliente comprou produto daquela classificação e 0 o oposto, através do **df_compras**.

Input: - **df_compras** (tabela de todas as vendas presentes no banco de dados).

Output: - Matriz esparsa de classificação por cliente.

purchase_similarity_recom()

Função: Calcula a similaridade dos códigos de produtos recebidos como input e o código do produto inserido no carrinho, baseado nas compras realizadas.

Input: - **products_codes** (array de códigos dos produtos possíveis para a recomendação), **code** (código do produto inserido no carrinho), **df_compras_pivot** (dataframe que possui o cliente, o produto e o somatório da quantidade que o respectivo cliente comprou do respectivo produto).

Output: Array de tuplas do score de similaridade do produto e o código do produto.

description_similarity_recom()

Função: Calcula a similaridade dos códigos de produtos recebidos como input e o código do produto inserido no carrinho, baseado nas descrições de cada produto.

Input: - **products_codes** (array de códigos dos produtos possíveis para a recomendação), **code** (código do produto inserido no carrinho), **df_products** (tabela de todos os produtos presentes no banco de dados), **stopwords** (palavras chaves para o calculo da similaridade da descrição).

Output: Array de tuplas do score de similaridade do produto e o código do produto.

get_products_list()

Função: Remove os códigos de produtos duplicados, inclui o código do produto inserido no carrinho e transforma em array

Input: - **code** (código do produto inserido no carrinho), **products** (lista de possíveis produtos para serem recomendados, pode incluir duplicatas)

Output: Array de códigos dos produtos possíveis para a recomendação

calculate_recommendations_similarity()

Função: Calcula os 10 produtos mais semelhantes ao do produto inserido no carrinho, baseado na similaridade de compras e de descrição do produto

Input: - **code** (código do produto inserido no carrinho), **df_compras** (tabela de todas as vendas presentes no banco de dados), **df_products** (tabela de todos os produtos presentes no banco de dados), **sim_results** (matriz de similaridade entre classificações), **df_compras_pivot** (dataframe que possui o cliente, o produto e o somatório da quantidade que o respectivo cliente comprou do respectivo produto), **max_recom** (numero máximo de classificações semelhantes ao do produto inserido no carrinho, atualmente 2 classificações semelhantes)

Output: Array dos 10 produtos mais semelhantes ao do código do produto inserido no carrinho

Classe ClientRecom

Possui uma única instância criada durante a execução do arquivo "**api.py**", possui o papel de uma interface entre as requisições de recomendações para um cliente e para o treinamento do mesmo. Além disso, essa instância tem acesso as duas outras classes, a **BdManagement** e a **Model**.

__ init __()

Função: Carregar os arquivos "**pickle/client_convert_filial.pickle**" e "**pickle/new_client_output.pickle** para suas respectivas variáveis e pegar do banco de dados as recomendações salvas para cada um dos clientes.

Input: -

Output: -

get_client_to_recommend()

Função: Recomendar códigos de produtos, baseado no código do cliente que já consta na base de dados de vendas.

Input: - **user_id** (código do cliente a qual será recomendado).

Output: Array com código de produtos recomendados.

recommend_to_new_client()

Função: Recomendar códigos de produtos, baseado no código da filial onde o cliente novo se encontra.

Input: - **cod_filial** (código da filial)

Output: Array com código de produtos recomendados.

recommendations()

Função: Muda o índice do DataFrame que possui a recomendação para cada um dos clientes, no código do cliente.

Input: -

Output: DataFrame contendo recomendações de produtos para cada um dos clientes.

get_clients_output()

Função: Retorna o DataFrame contendo recomendação para cada um dos clientes.

Input: -

Output: - DataFrame contendo recomendações de produtos para cada um dos clientes.

train_new_clients()

Função: Faz o treinamento da recomendação para clientes novos, baseado nos produtos mais populares em cada uma das filiais existentes e salva os resultados em dois arquivos, complementares, no formato .pickle.

Input: - **db_cart** (tabela de todas as vendas presentes no banco de dados).

Output: -

train_cliente_recom()

Função: Faz o treinamento da recomendação de clientes, para cada um dos clientes presente na tabela de vendas **db_purchase**, baseado na similaridade de compras entre cada cliente e salva os resultados no banco de dados.

Input: - **db_purchase** (tabela contendo o cliente, o produto e quantidade vendida do produto para o cliente em cada uma das venda).

Output: -

Classe Model

Possui uma única instância criada durante a execução da classe **ClientRecom**, possui o papel de realizar o treinamento do **ClientRecom**, através de manipulação dos dados de vendas e calcula a similaridade entre as compras de cada cliente. Além disso, possui outra classe dentro da mesma, **ClientProductMap**, responsável por fazer o mapeamento da frequência de compras de cada cliente para cada um dos produtos comprados pelos mesmos.

split_data()

Função: Transforma o DataFrame em SFrame da biblioteca turicreate.

Input: - **data** (DataFrame contendo o cliente, o produto e a frequência de compras do cliente para aquele respectivo produto).

Output: SFrame contendo o cliente, o produto e a frequência de compras do cliente para aquele respectivo produto.

recom_model()

Função: Calcula a similaridade entre a frequência de compras entre cada um dos clientes e cria tabela contendo recomendação de produtos para cada um dos clientes.

Input: - **train_data** (SFrame contendo o cliente, o produto e a frequência de compras do cliente para aquele respectivo produto), **user_id** (String do nome da coluna que corresponde ao código do cliente), **item_id** (String do nome da coluna que corresponde ao código do produto), **users_to_recommend** (lista de todos os usuários presentes nos dados de vendas), **n_rec** (Número de produtos a ser recomendado para cada um dos clientes).

Output: SFrame contendo recomendação de produtos para cada um dos clientes.

create_output()

Função: Transformar a tabela SFrame em DataFrame e unir os produtos recomendados, para cada cliente, num único registro por cliente na tabela.

Input: - **model** (SFrame contendo recomendação de produtos para cada um dos clientes), **user_id** (String do nome da coluna que corresponde ao código do cliente), **item_id** (String do nome da coluna que corresponde ao código do produto), **users_to_recommend** (lista de todos os usuários presentes nos dados de vendas), **n_rec** (Número de produtos a ser recomendado para cada um dos clientes).

Output: DataFrame contendo recomendação de produtos para cada um dos clientes.

ClientProductMap.__ init __()

Função: Inicializa as variáveis utilizadas durante o processamento, ao criar uma instância da classe **ClientProductMap**.

Input: -

Output: -

ClientProductMap.__ del __()

Função: Deleta as variáveis utilizadas após o processamento das função da instância da classe **ClientProductMap**

Input: -

Output: -

ClientProductMap.add()

Função: Adiciona o código de cliente, o código do produto e a quantidade comprada no mapeamento de listas de adjacências e calcula o máximo vendido para cada um dos produtos.

Input: - **cod_cli** (Código do cliente da compra), **cod_prod** (Código do produto da compra), **freq** (quantidade comprada do produto pelo cliente).

Output: -

ClientProductMap.getNormalizedFreq()

Função: Cria um array contendo a frequência de compras, normalizada por máximo e mínimo, de cada um dos clientes para cada um dos produtos comprados pelos mesmos.

Input: -

Output: Array de triplas contendo o código do cliente, o código do produto e frequência normalizada de compras do cliente para o produto.

Classe Bicluster

Possui uma única instância criada durante a execução da classe "**api.py**", possui o papel de realizar o treinamento da recomendação de cliente, utilizando o método de biclusterização, e de recomendar para o cliente em si.

__ init __()

Função: Inicializar o carregamento de variáveis utilizadas para realizar a recomendação.

Input: -

Output: -

get_adjacency_list()

Função: Carrega as variáveis responsáveis pela manipulação das listas adjacentes que serão utilizadas durante a recomendação.

Input: - **txt_file_name** (String do caminho onde se encontra o arquivo .txt).

Output: -

get_biclusters()

Função: Carrega as variáveis responsáveis pela recomendação do cliente utilizadas durante a recomendação.

Input: -

Output: -

recomenda_cliente()

Função: Recomenda produtos para clientes que possuem mais de 2 compras na base de dados das vendas.

Input: - **cliente** (código do cliente a qual será recomendado).

Output: Array de produtos recomendados, String informando se operação foi sucedida ou não, Inteiro informando o código de status em HTML.

recomenda_produto()

Função: Recomenda códigos de clientes, através do código de um produtos, caso o cliente possua mais de 2 compras na base de dados das vendas.

Input: - produto(código do produto a qual será recomendado).

Output: Array de códigos de clientes recomendados

create_biclusters()

Função: Pega o bicluster criado, através da execução do código em C++, e cria as variáveis responsáveis na recomendação do cliente. Além disso, salva o resultado em arquivos .pickle.

Input: -

Output: -

create_adjacency_list()

Função: Cria as variáveis responsáveis pela manipulação de listas de adjacência, utilizadas como auxiliadoras para a recomendação do cliente.

Input: - **db_cart** (tabela de todas as vendas presentes no banco de dados).

Output: -

get_exitcode_stdout_stderr()

Função: Executa comandos no terminal da máquina, através de bibliotecas do python, e retorna resposta desses comandos.

Input: - **cmd** (String do comando a ser executado no terminal).

Output: Inteiro do código de saída da execução do comando, String contendo o resultado da operação, String informando o erro da operação

execute_terminal_command()

Função: Faz a execução de três comandos no terminal: Compilar o código do bicluster, tornar o executável do bicluster acessível a todos e executar o código do bicluster.

Input: -

Output: - String contendo o resultado da operação da execução do código do bicluster.

Classe Recommendation

Possui uma única instância criada durante a execução da classe "**api.py**", possui o papel de interface para o treinamento de todas as instâncias de classes responsáveis pela recomendação. Além disso, essa instância tem acesso as duas outras classes, a **BdManagement** e a **ExtractDescription**.

__ init __()

Função: Inicializar o carregamento de variáveis responsáveis por possuir os dados tabela de vendas, os dados tabela de vendas do treinamento de cliente e os dados da tabela de produtos existentes.

Input: -

Output: -

retrain_model()

Função: Atualiza as variáveis dos dados da tabela de vendas, os dados tabela de vendas do treinamento de cliente e os dados da tabela de produtos existentes e inicializa o treinamento da recomendação feita pelo Bicluster, recomendação do Carrinho, recomendação de clientes do ClientRecom e recomendação de clientes novos do ClientRecom.

Input: - **bicluster_recom** (instância da classe Bicluster), **cart_recom** (instância da classe CartRecom), **client_recom** (instância da classe ClientRecom)

Output: - String afirmando o fim do treinamento de todas as instâncias.

SimilarityModel

create_cosine_similarity_matrix()

Função: Calcular a matriz de similaridades entre classificações (categorias) de produtos.

Input: **matrix_u_c** - Matriz transposta de usuario por classe de produto

Output: **sim_results** - Dicionário com as classes e suas respectivas similaridades com as demais classes. O dicionário é da forma: key: classe, value: [classe, similaridade].

get_product_classif()

Função: Retornar a classificação (categoria) de um determinado produto .

Input: **code** - código do produto a ser analisado, **df_compras** - tabela de todas as vendas presentes no banco de dados.

Output: **classif** - String com a classificação de um determinado produto.

recommendation()

Função: Retornar as classificações de produtos que podem ser recomendados a um determinado produto.

Input: **code** - código do produto a ser analisado, **max_recom** - número máximo de recomendações, **df_compras** - tabela de todas as vendas presentes no banco de dados, **sim_results** - dicionário com as classes e suas respectivas similaridades com demais classes.

Output: **sim_results** - Dicionário com as classes e suas respectivas similaridades com as demais classes. O dicionário é da forma: key: classe, value: [classe, similaridade].

get_products_recom_array()

Função: Pegar os produtos a serem recomendados para um dado produto.

Input: **code** - código do produto a ser analisado, **max_recom** - número máximo de recomendações, **df_compras** - tabela de todas as vendas presentes no banco de dados, **sim_results** - dicionário com as classes e suas respectivas similaridades com demais classes.

Output: **df_products_recoms** - Lista com produtos a serem recomendados para o produto de entrada.