LIGHTWEIGHT RECOMMENDATION SYSTEM
Design Credit Project

GUIDED BY:
Dr. Binod Kumar

CONTRIBUTED BY:
Kaustabh Mishra (B20AI063)
Aman Srivastava (B20CS100)

Link to video presentation:
https://drive.google.com/file/d/1aJSSTF5mOA6DEgENuyaprURSWeYD0gFg/view?usp=sharing

INTRODUCTION:

This report presents a detailed analysis of a lightweight recommendation system developed to provide personalized meal recommendations to users. The objective of this recommendation system is to offer tailored suggestions based on users' preferences and dietary requirements, while maintaining a lightweight infrastructure that ensures efficiency and accessibility. By leveraging simplified algorithms and minimal computational resources, the system aims to deliver practical and relevant recommendations to enhance the user experience in meal selection.

The lightweight recommendation system addressed in this report takes a different approach by focusing on simplicity and efficiency. By leveraging content-based filtering and user-based collaborative filtering techniques, the system can generate personalized meal recommendations without the need for excessive computational complexity. This allows users to receive valuable insights and suggestions in a timely manner, helping them make informed decisions about their meals.

Throughout this report, we discuss the methodology employed in building the lightweight recommendation system. This includes data collection, preprocessing, and feature engineering techniques to create a comprehensive dataset for recommendation generation. We will also explore the implementation of content-based and user-based collaborative filtering techniques, highlighting the critical steps involved in generating personalized recommendations.

DATA COLLECTION:

The success of a recommendation system relies heavily on the availability of high-quality and diverse data. Regarding our lightweight recommendation system for meal suggestions, we employed web scraping to gather the necessary information.

Web scraping allowed us to extract data from various online sources that includes
Healthline (https://www.healthline.com)
NDTV Food (https://food.ndtv.com/recipes/)
Wikipedia (https://en.wikipedia.org/wiki/List_of_Indian_dishes)
We utilized Python-based web scraping tools BeautifulSoup to navigate through web pages, extract relevant information such as meal names, ingredients, nutrient details, and user ratings. By scraping multiple sources, we ensured a wide range of meal options and a comprehensive dataset for analysis and recommendation purposes.

After data collection, we had scrapped data (data.csv) with 1951 rows from almost 1951 links and this data can be processed to create a perfect dataset.

DATA PROCESSING: (.ipynb file)

Data processing is done for data.csv which contains scrapped data. The data processing tasks aimed to clean the data, handle missing values, add new columns, and prepare the dataset for further analysis. By reviewing the code, we can gain a comprehensive understanding of the implemented data processing steps and evaluate their effectiveness.

1. Data Cleaning:
   - Duplicate Removal: The dataset was checked for duplicate entries, and any duplicate rows were removed to ensure data integrity and avoid redundancy.
   - Noise Removal: In order to enhance data quality, any irrelevant or noisy data was identified and removed from the dataset.

2. Handling Missing Values:
   - Missing Value Identification: The dataset was examined to identify columns with missing values.

- Missing Value Treatment: Various strategies were employed to handle missing values, including:
- Removal: Rows containing missing values in critical columns were removed to ensure data completeness and accuracy.
- Imputation: Missing values in non-critical columns were filled using appropriate imputation techniques, such as mean, median, or mode.

3. Adding New Columns:
    - Price: A new column named "Price" was added to the dataset, representing the price of each item. Random price values within a specified range were generated and assigned to the items.
    - User_Id: Another new column named "User_Id" was added to represent the unique identifier for each user. Random user IDs were generated and assigned to the dataset.
    - Meal_Id: A column named "Meal_Id" was created to uniquely identify each meal item in the dataset. A mapping dictionary was used to assign a unique ID to each distinct meal name.

4. Dataset Preparation:
    - Category Processing: The code accurately processes the "catagory" column, ensuring consistency and standardization of categories. It uses keywords to identify and assign appropriate categories to the items. This step enhances the usability and interpretability of the dataset.
    - Nutrient Processing: The code correctly assigns specific nutrients to items based on their category and sub-category. It effectively fills in missing nutrient information, enhancing the completeness and relevance of the dataset.
    - User Profiles: The code extracts a subset of the dataset to create a user profile dataset. It includes relevant information such as User_Id, Veg_Non preference, Nutrient preference, Disease condition, and Diet preference. Duplicate user profiles are removed, and the resulting dataset is saved separately as "user_Profiles.csv". This dataset serves as a foundation for user-based analyses and personalized recommendations.
    - Recent Activity: Another subset of the dataset is used to create a dataset representing recent user activity. The code assigns random activity indicators such as rating, liking, searching, and purchasing to the user's interactions with meal items. The resulting dataset is saved as "recent_activity.csv" and captures recent user behavior for recommendation system development.

- Recommendation Dataset: The code refines the original dataset by selecting relevant columns such as Meal_Id, Name, catagory, description, Veg_Non preference, Nutrient, Disease, Diet, and Price. The refined dataset is saved as "dataset.csv" and serves as a comprehensive dataset for recommendation system implementations.

USER PROFILE GENERATION:

The Profile class allows users to create a personalized profile based on their dietary preferences, nutrient requirements, dietary restrictions, food type preferences, and favorite foods. By providing these inputs, users can generate a customized profile that matches their specific needs. The class filters the main dataset based on the inputs and stores the filtered results in separate dataframes. The get_profile method returns the merged and filtered dataframe, containing relevant information such as meal names, nutrients, food types, prices, reviews, and descriptions.

RECOMMENDATION:

1. Recommendation by 'KNN' (Content - based filtering) (.ipynb file)

   We first implemented a recommendation system based on content-based filtering using the K Nearest Neighbors (KNN) algorithm. The system takes user preferences, such as diet, disease, nutrient requirements, and favorite food, as input. It then matches these preferences with the available items in the dataset. The dataset contains information about various meals, including their names, nutrients, vegetarian/non-vegetarian status, prices, reviews, diets, diseases, and descriptions. The system uses this information to find similar meals that match the user's preferences.
   To achieve this, the code first preprocesses the dataset by converting categorical variables into binary representations. It then applies the KNN algorithm to find the k nearest neighbors based on the user's preferences. The algorithm considers the similarity of features such as nutrients, diseases, and diets to determine the most relevant recommendations.
   The recommended items, along with their attributes, are returned as output. These attributes provide information about the recommended meals, allowing the user to make informed choices.

2. User based Recommendation (Collaborative filtering) (.ipynb file)

Next we implemented a user-based recommendation system using collaborative filtering. The system aims to recommend meals to a target user based on the preferences and activities of similar users.
The Recommender class is then defined, which has methods to calculate user similarity and provide recommendations. The get_features method converts the user profiles into binary representations using dummy variables. The k_neighbor method uses the K Nearest Neighbors algorithm to find similar users based on their profiles.
To make recommendations, the user_based method takes a user profile and user ID as input. It converts the profile into binary form and finds similar users. It then retrieves the activity data of those similar users and filters out the meals that the target user has not reviewed. Finally, the recommendations, along with their attributes, are returned.
To demonstrate the functionality, an example user profile (profile A) and user ID are provided. The code calculates the similarity between profile A and other users, retrieves their activity data, and recommends meals that have been reviewed by those users but not by user A.
The recommendations include information such as meal name, nutrients, vegetarian/non-vegetarian status, price, reviews, diet restrictions, diseases, and descriptions. These recommendations can help the target user discover new meals based on the preferences and activities of similar users.

3. Final Collaborative Recommender (.ipynb file)

The recommender utilizes a user-based collaborative filtering approach to identify similar users and suggest meals that have been enjoyed by those users but not yet rated or reviewed by the target user.
The code begins by loading the user profiles, recent activity data, and the main dataset containing meal information. The Recommender class is then defined, which incorporates methods for feature extraction, finding similar users or meals using the k-nearest neighbors algorithm, and generating recommendations based on user profiles and recent activity.
The recommend method combines user-based and recent activity-based recommendations to create a comprehensive list of meal suggestions. By considering the user's profile features and analyzing their recent activity, the system offers personalized recommendations that cater to the user's preferences and dietary needs.
By integrating user profiles and recent activity, the final collaborative recommender system delivers tailored meal recommendations, enhancing the

user's dining experience and promoting exploration of new and enjoyable culinary options.

CONCLUSION:

In conclusion, the lightweight recommendation system developed for personalized meal suggestions offers a practical and efficient solution. The system generates tailored meal recommendations based on users' preferences and recent activity by leveraging content-based and user-based collaborative filtering techniques. The system's simplicity and efficiency ensure timely and relevant suggestions without excessive computational complexity. The data collection process provides a diverse dataset, enabling comprehensive analysis and accurate recommendation generation. Overall, the system enhances the user experience in meal selection by providing personalized and relevant suggestions based on individual preferences and dietary requirements.