Data Analysis on Big Basket Products

Reyhaneh Saffar_99222064

1. Data Exploration

This dataset is a collection of products's details.
Loading the dataset: I used pandas to read the csv file and store it in a data frame . The dataset has 27555 rows and 9 columns. Here is a brief description of each column and its data type:
Product: Name of the product. **String**
Category: Category of the product. **String**
Sub-Category: Sub-category of the product. **String**
Sale Price: The discounted price of the product**. Float**
Market Price: The original price of the product before discount**. Float**
Type: The type of the product**. String.**
Description: A brief description of the product**. String**
Brand: Brand of the product. **String**
Rating: Average rating of the product. Data type: **Float**

2. Data Processing

In the data processing phase, a comprehensive analysis of the dataset's unique values is conducted across key columns. This exploration provides valuable insights into the diversity and uniqueness of information within each column.
The output reveals the distinctiveness of entries in critical columns, such as 'product,' 'category,' 'sub_category,' 'brand,' 'sale_price,' 'market_price,' 'type,' 'rating,' and 'description.' The number of unique values in each column is presented, offering a clear picture of the dataset's variety and richness.
By examining the unique values, it becomes apparent how many different products, categories, sub-categories, brands, and other essential attributes are present in the dataset.

- **Null Values:**
  In the data processing phase, a thorough examination of null values is conducted to identify and quantify missing data in key columns.
  A separate dataframe is created to mirror the original dataset, allowing for a comprehensive analysis without altering the original dataset.
  A summary report is generated, presenting the count and percentage of null values for each relevant column.
  This report serves as a valuable diagnostic tool, shedding light on the extent of missing information in critical fields such as product, brand, rating, and description. To address missing values effectively, a multi-step strategy is implemented. First, rows with null values in certain crucial columns, like 'product' and 'description', are systematically removed to ensure data integrity. For the 'rating' column, where a significant proportion of data is missing, a strategic imputation method is employed. The missing values in 'rating' are replaced with the median value, a robust measure that minimizes the impact of outliers.

- **Encoding Categorical Columns:**
  In the data processing section dedicated to encoding categorical columns, several techniques were employed to transform non-numeric data into a format suitable for statistical analysis. For the 'category' column, a one-hot encoding method was applied, creating binary columns to represent different categories. The 'sub_category' column underwent label encoding using the sklearn library's LabelEncoder, assigning numerical labels to each unique subcategory. In the case of the 'brand' column, frequency encoding was implemented by mapping each brand to its corresponding frequency percentage within the dataset. Finally, the 'type' column was also label encoded to facilitate numerical representation. These encoding techniques contribute to the preparation of the dataset for further analysis, enabling statistical models to interpret and derive meaningful insights from the categorical information present in the original data.
  In the data processing section related to the 'description' column, the code performs several operations to transform textual data into a format suitable for machine learning models. The length analysis of the descriptions reveals that there are 1895 unique lengths, with a minimum length of 7 and a maximum length of 4486 characters. The code then utilizes the Keras library to tokenize the text and convert it into sequences, forming a word index that captures 32,831 unique tokens within the descriptions. The function `word_embedding` is designed to facilitate this process, returning the word index and sequences.
  The 'description_sequences' column is then created in the DataFrame, containing the tokenized sequences. Subsequently, a conversion function is implemented to ensure uniform length across all sequences, padding with zeros where necessary. This step is crucial for compatibility with neural network models. Finally, after the conversion is complete, the 'description_sequences' column is dropped from the DataFrame, ensuring that the dataset is appropriately preprocessed and ready for further analysis.

3. **Recommender System:**
   In the recommender system section, the process begins with creating a copy of the encoded DataFrame. The data processing involves vectorizing the rows of the DataFrame based on specified columns, excluding certain columns related to product information. The vectorized rows are then organized into a dictionary, where each product value is associated with its corresponding vectorized row.
   The primary method used for similarity calculation is the Jaccard Similarity. This method computes the similarity between two sets by measuring the intersection and union of their elements. The goal is to find similar products for a given customer's selected product. The similarity threshold is set at 0.325, and the algorithm iterates through the products, comparing the Jaccard Similarity with the customer's product set. The system prompts the user to input their considered product, and based on the Jaccard Similarity, it outputs a list of products that are deemed similar to the user's selection. The process repeats until the user decides to disconnect. The dictionary size, representing the number of unique products in the dataset, is also displayed during the process.

The End.