MT23040
Jafreen Rizvi

IR Assignment 3

Q1. I tried to keep both the files in separate dataframe but I got error memory out of bound error.

Q2. I have chosen the product 'Laptop' and filtered out both files in

```
df=pd.read_csv("filtered_main_df.csv",low_memory=False)
```

```
df2=pd.read_csv("laptop_metadata.csv",low_memory=False)
```

Q3. Report the total number of rows for the product- for df =56756, for df2=287330

Perform appropriate pre-processing as handling missing values, duplicates and other.

Approach:

- Read two CSV files, "filtered_main_df.csv" and "laptop_metadata.csv", into Pandas DataFrames.
- Preprocess the data: drop rows with missing 'asin' values, and remove duplicate rows based on the 'asin' column from both DataFrames.
- Output the number of rows after preprocessing for each DataFrame.

Methodologies:

1. Data Loading:

- Utilize Pandas' `read_csv` function to load the CSV files into DataFrames (`df` and `df2`).

2. Data Preprocessing:

- Drop rows with missing 'asin' values using the `dropna` function.

- Remove duplicate rows based on the 'asin' column using the `drop_duplicates` function.

3. Reporting:

- Print the number of rows after preprocessing for each DataFrame.

Assumptions:

- 'asin' is the unique identifier for each item in the datasets.

- The preprocessing steps of dropping missing values and removing duplicates based on 'asin' are sufficient for data cleaning.

MT23040
Jafreen Rizvi

Results:

```
print("Number of rows after preprocessing in meta ", len(df2))
```

```
Number of rows after preprocessing in meta  56756
```

```
print("Number of rows after preprocessing ",len(df))
```

```
Number of rows after preprocessing  287330
```

Q4.

Approach:

- Calculate various statistics and metrics based on the data in the DataFrame df.
- Obtain the number of reviews, average rating score, number of unique products, number of good and bad ratings, and the distribution of reviews corresponding to each rating.

Methodologies:

- Calculate Number of Reviews:
- Determine the total number of reviews in the DataFrame df.
- Calculate Average Rating Score:
- Compute the mean rating score from the 'overall' column in the DataFrame df.
- Calculate Number of Unique Products:
- Determine the count of unique products based on the 'asin' column in the DataFrame df.
- Calculate Number of Good and Bad Ratings:
- Filter the DataFrame df to separate good and bad ratings based on a threshold (3 in this case).
- Count the number of good ratings and calculate the number of bad ratings as the difference between total reviews and good ratings.
- Calculate Number of Reviews Corresponding to Each Rating:
- Compute the frequency distribution of ratings using the 'overall' column in the DataFrame df.

Assumptions:

- 'overall' column represents the rating scores in the DataFrame df.
- Ratings equal to or above 3 are considered good ratings, while ratings below 3 are considered bad ratings.

Results:

```
a. Number of Reviews: 287330
b. Average Rating Score: 4.249778999756377
c. Number of Unique Products: 9262
d. Number of Good Ratings: 253378
e. Number of Bad Ratings: 33952
f. Number of Reviews corresponding to each Rating:
1.0     20668
2.0     13284
3.0     21837
4.0     49363
5.0    182178
Name: overall, dtype: int64
```

Q.5

Approach:

- Perform text preprocessing on the 'reviewText' column of the DataFrame df.
- Implement various steps to clean and normalize text data for analysis.

Methodologies:

- Removing HTML Tags:
- Utilize BeautifulSoup to parse and remove HTML tags from the text data.
- Removing Accented Characters:
- Normalize text by converting accented characters to their ASCII equivalents using unicodedata.normalize().
- Expanding Acronyms:
- Replace known acronyms with their expanded forms in the text data.
- Removing Special Characters:
- Implement regular expressions to remove special characters from the text data.
- Lemmatization:
- Utilize NLTK's WordNetLemmatizer to lemmatize verbs in the text data, reducing them to their base form.
- Text Normalizer:
- Lowercase all text and join lemmatized words to form normalized text.

Assumptions:

- The 'reviewText' column contains text data that needs to be cleaned and normalized.
- Lemmatization is performed specifically for verbs (wordnet.VERB), assuming that the text mainly consists of verbs that need to be lemmatized.

Results:

MT23040
Jafreen Rizvi

The 'reviewText' column in the DataFrame df is processed to remove HTML tags, accented characters, special characters, and lemmatize verbs. The text is then normalized to lowercase.

Q6.

Approach:

Perform exploratory data analysis (EDA) on the merged DataFrame merged_df to extract relevant statistics regarding reviews of electronic products. The analysis includes identifying top and least reviewed brands, most positively reviewed product, rating distribution over consecutive years, word cloud analysis for good and bad ratings, distribution of ratings versus number of reviews, year with maximum reviews, and year with the highest number of customers.

Methodologies:

a. Top 20 most reviewed brands:

Group the data by brand and aggregate the sum of overall ratings.

Sort the brands based on the sum of ratings in descending order to identify the top 20 most reviewed brands.

	brand	overall
1152	Logitech	49817.0
158	Asus	45874.0
363	Case Logic	42210.0
506	Dell	37015.0
109	AmazonBasics	29091.0
140	Apple	21146.0
842	HP	19197.0
77	Acer	18356.0
1383	PWR+	18336.0
430	Corsair	16966.0
1729	Targus	16946.0
1127	Lenovo	15846.0
649	Evecase	15031.0
124	Anker	13667.0
766	Generic	13604.0
1817	UGREEN	12543.0
1884	VicTsing	12301.0
1742	TeckNet	12154.0
1790	Toshiba	11159.0
1562	Samsung	9681.0

b. Top 20 least reviewed brands:

Similar to the approach for the most reviewed brands, but sorting in ascending order to identify the least reviewed brands.

MT23040
Jafreen Rizvi

	brand	overall
1704	THZY	10.0
704	Fly Infotech	11.0
1141	LinDon-Tech	11.0
730	GE	12.0
115	Ammorn	13.0
343	CSRET	13.0
1971	Xorastra	13.0
540	Ducti	13.0
33	AIKONSOUND	13.0
1366	Osurce	13.0
326	COMEHERE	14.0
1532	SODIAL(TM)	14.0
428	Cooper Cases	14.0
752	GSAstore TM	15.0
1875	Vensmile	15.0
1356	Ona	15.0
136	Aokland	15.0
1399	Pesp	15.0
1444	QualityArt	15.0
970	Jopuzia	15.0

c. Most positively reviewed product:

Calculate the mean overall rating for each product brand.

Identify the brand with the highest mean rating as the most positively reviewed product.

```
most reviewed brand
```

	brand	overall
1152	Logitech	49817.0

d. Count of ratings over 5 consecutive years:

Extract the year from the 'reviewTime' column.

Filter the data for the years 2012 to 2017.

Count the ratings for each year to observe trends over consecutive years.

```
Count of Ratings for Laptop from 2012 to 2017:
```

	year	overall
0	2012	10390
1	2013	22074
2	2014	38368
3	2015	60879
4	2016	71854
5	2017	47936

e. Word Cloud for good and bad ratings:

Preprocess the text data by converting to lowercase, removing punctuation, and stopwords.

Separate reviews into good and bad based on specific keywords.

Generate word clouds to visualize the most commonly used words in good and bad reviews.

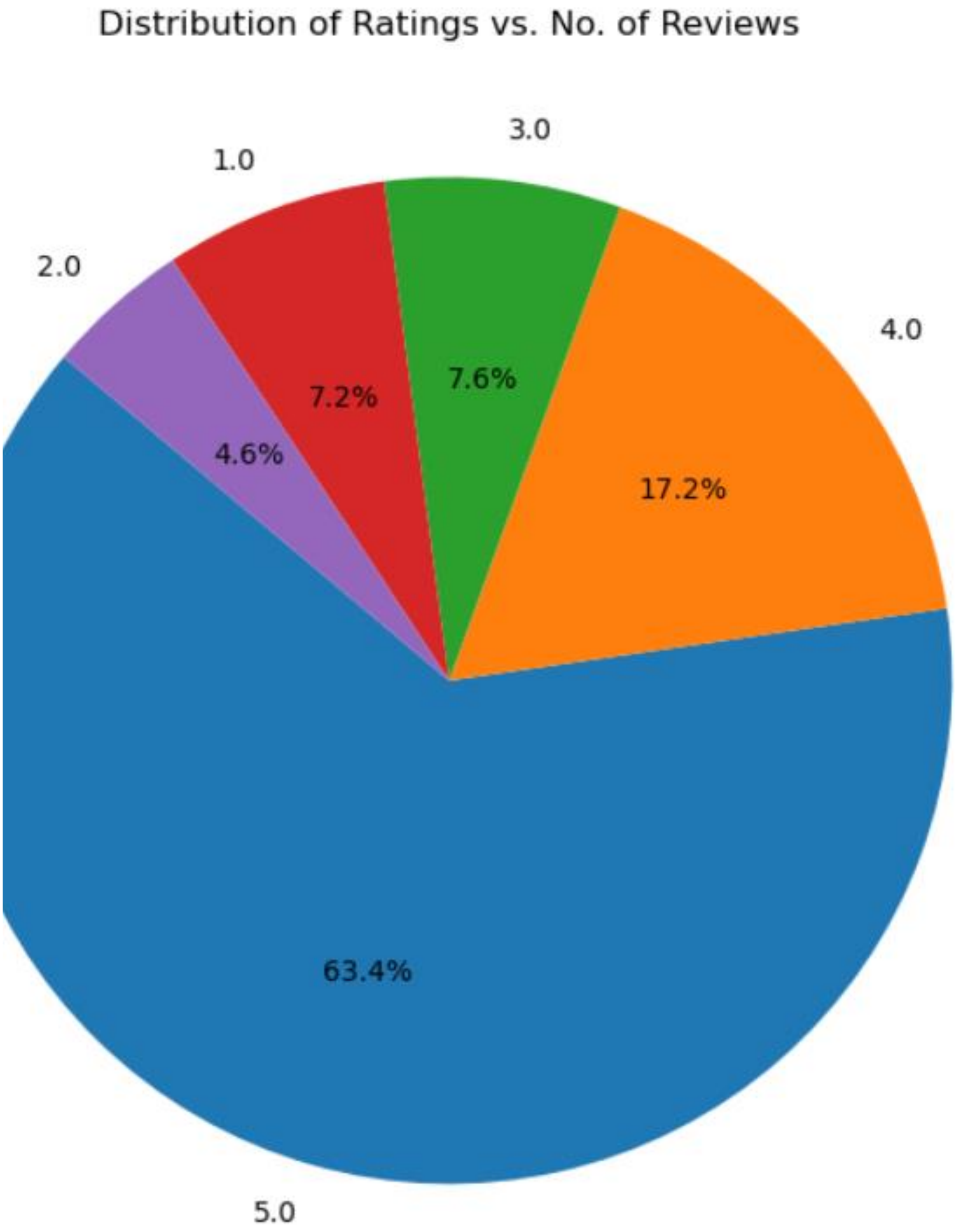
MT23040
Jafreen Rizvi

f. Distribution of Ratings vs. No. of Reviews:

Count the number of reviews for each rating.

Plot a pie chart to visualize the distribution of ratings.

MT23040
Jafreen Rizvi



g. Year with maximum reviews:

Extract the year from the 'reviewTime' column.

Count reviews for each year and identify the year with the maximum reviews.

```
The product got maximum reviews in the year: 2016
```

MT23040
Jafreen Rizvi

h. Year with the highest number of customers:

```
The year with the highest number of customers is: 2016
```

Group the data by year and count the number of unique customers (products) for each year.

Identify the year with the highest number of customers.

Assumptions:

The 'merged_df' DataFrame contains the merged data of reviews and product metadata.

The 'reviewTime' column is in datetime format and contains review timestamps.

Q7.

Approach:

Utilize CountVectorizer from scikit-learn to convert the text data in the 'reviewText' column of the DataFrame df into a bag-of-words (BoW) representation. This will enable the extraction of features from the text data, which can be used for further analysis or modeling tasks.

Methodologies:

- Initialize CountVectorizer:
- Import CountVectorizer from the scikit-learn library.
- Fit and Transform the Data:
- Initialize an instance of CountVectorizer.
- Fit the CountVectorizer on the 'reviewText' column to learn the vocabulary and transform the text data into a sparse matrix representation.
- Get Feature Names:
- Retrieve the feature names from the CountVectorizer instance. These feature names correspond to the unique words in the text data.

Assumptions:

- The 'reviewText' column contains textual data that needs to be converted into a bag-of-words representation.
- No specific preprocessing steps (such as tokenization or stopword removal) are applied before using CountVectorizer. If needed, additional preprocessing can be done before fitting the CountVectorizer.

Results:

MT23040
Jafreen Rizvi

- The text data in the 'reviewText' column of DataFrame df is converted into a bag-of-words representation using CountVectorizer.
- The feature names (unique words) extracted from the text data are obtained.

Q8, Q9 and Q10

Approach:

In this task, we aim to compare the performance of five different machine learning models in classifying the sentiment of reviews into three categories: Good, Average, and Bad. We'll use the review text as the input feature and the rating class as the target variable. The data will be divided into training and testing sets in a 75:25 ratio for evaluation.

Methodologies:

- Data Preparation:
- Define a function categorize_rating() to categorize ratings into three classes: Good, Average, and Bad based on the given criteria.
- Apply this function to create the 'rating_class' column in the DataFrame df.
- Train-Test Split:
- Split the data into training and testing sets using the bag-of-words representation (X_bow) as the input feature and the 'rating_class' column as the target variable. The ratio of the train-test split is set to 75:25.
- Model Training and Evaluation:
- Initialize and train five machine learning models: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).
- For each model, predict the target variable on the test set and calculate the classification report, including precision, recall, F1-score, and support for each target class (Good, Average, Bad).

Assumptions:

- The 'rating_class' column is created based on the categorization function provided.
- The bag-of-words representation (X_bow) is used as the input feature for model training.
- The target variable (y) consists of three classes: Good, Average, and Bad.
- The performance of the models is evaluated using precision, recall, F1-score, and support for each target class.

Results:

- Five machine learning models are trained and evaluated based on their performance metrics.
- The classification report for each model is displayed, providing insights into the precision, recall, F1-score, and support for each target class (Good, Average, Bad).

```
  n_iter_i = _check_optimize_result(

Classification Report for Logistic Regression:
              precision    recall  f1-score   support

        Good       0.43      0.12      0.19      5508
     Average       0.71      0.57      0.63      8416
         Bad       0.89      0.97      0.93     57909

    accuracy                           0.86     71833
   macro avg       0.68      0.55      0.58     71833
weighted avg       0.83      0.86      0.84     71833

Classification Report for Decision Tree:
              precision    recall  f1-score   support

        Good       0.24      0.19      0.21      5508
     Average       0.51      0.49      0.50      8416
         Bad       0.89      0.91      0.90     57909

    accuracy                           0.81     71833
   macro avg       0.55      0.53      0.54     71833
weighted avg       0.79      0.81      0.80     71833

Classification Report for Random Forest:
              precision    recall  f1-score   support

        Good       0.80      0.06      0.11      5508
     Average       0.86      0.28      0.42      8416
         Bad       0.84      1.00      0.91     57909

    accuracy                           0.84     71833
   macro avg       0.83      0.44      0.48     71833
weighted avg       0.84      0.84      0.79     71833

Classification Report for Support Vector Machine:
              precision    recall  f1-score   support

        Good       0.45      0.03      0.06      5508
     Average       0.75      0.56      0.64      8416
         Bad       0.88      0.99      0.93     57909

    accuracy                           0.86     71833
   macro avg       0.69      0.53      0.54     71833
weighted avg       0.83      0.86      0.83     71833

Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support
```

Q11.

Approach:

MT23040
Jafreen Rizvi

In this task, we aim to create a user-item rating matrix, normalize the ratings using min-max scaling, and then build a user-user collaborative filtering recommender system. We will find the top N similar users based on cosine similarity and evaluate the recommendation performance using k-fold cross-validation. Finally, we will report the Mean Absolute Error (MAE) for different values of N (number of similar users) in the recommendation system.

Methodologies:

Collaborative Filtering:

a) Create User-Item Rating Matrix:

Use the provided DataFrame merged_df containing user-item-rating data to create a user-item rating matrix.

b) Normalize Ratings using Min-Max Scaling:

Use MinMaxScaler to normalize the ratings in the user-item matrix.

c) User-User Recommender System:

i) Find Top N Similar Users:

Calculate cosine similarity between users to find the top N similar users for each user.

ii) K-Fold Cross-Validation (K=5):

Divide the dataset into 5 subsets for k-fold cross-validation.

iii) Predict Missing Values and Calculate Error:

For each fold, predict the missing ratings using the training set and calculate the MAE using the validation set.

iv) Report MAE for Different Values of N:

Evaluate the recommendation system's performance for N = 10, 20, 30, 40, and 50 similar users.

Assumptions:

- The DataFrame merged_df contains user-item-rating data.
- The user-item rating matrix is created from the DataFrame.
- Ratings are normalized using min-max scaling.
- Cosine similarity is used to find similar users.
- K-fold cross-validation with K=5 is applied for evaluation.
- MAE is used as the evaluation metric for recommendation performance.

Results:

- User-item rating matrix and normalized ratings are created.
- Top N similar users are found using cosine similarity.
- K-fold cross-validation is performed to evaluate the recommendation system's performance.
- MAE is reported for different values of N (number of similar users) in the recommendation system.