CSE508_Winter2024_A3_2021004_Report

Aditya Bindlish

PART-1,2,3

Approach:

1. load_and_filter_metadata: This function filters the metadata to find the ASINs (Amazon Standard Identification Numbers) of products categorized under "Headphones".
2. filter_reviews: Utilizes the headphone ASINs from the first function to filter the main reviews dataset, keeping only the reviews related to those ASINs.
3. preprocess_reviews: Cleans the filtered reviews by dropping rows with missing review texts or overall ratings and removing duplicates based on reviewer ID and ASIN.
4. save_reviews: Saves the cleaned review data to a specified output path in a JSON format, with each line representing a review.

Assumptions:

No assumptions were made.

Methodologies:

1. Chunk Processing: To handle large datasets efficiently, the script reads the data in chunks. This approach allows for the processing of datasets larger than the available memory.
2. Set for ASIN Storage: Using a set for storing headphone ASINs ensures that only unique ASINs are kept, eliminating potential duplicates efficiently.

Results:

- Headphone related product ASINs are extracted from the metadata.
- Reviews related to these ASINs are filtered from the main reviews dataset.
- The filtered reviews are then cleaned by removing entries with missing data and duplicates.

PART-4

Approach:

1. The total number of reviews were derived.
2. The average rating score across all reviews.
3. The count of unique products were reviewed.
4. The number of good ratings (defined as ratings greater than or equal to 3).
5. The number of bad ratings (defined as ratings less than 3).
6. The distribution of reviews across different rating scores.

Assumptions:

No assumptions were made.

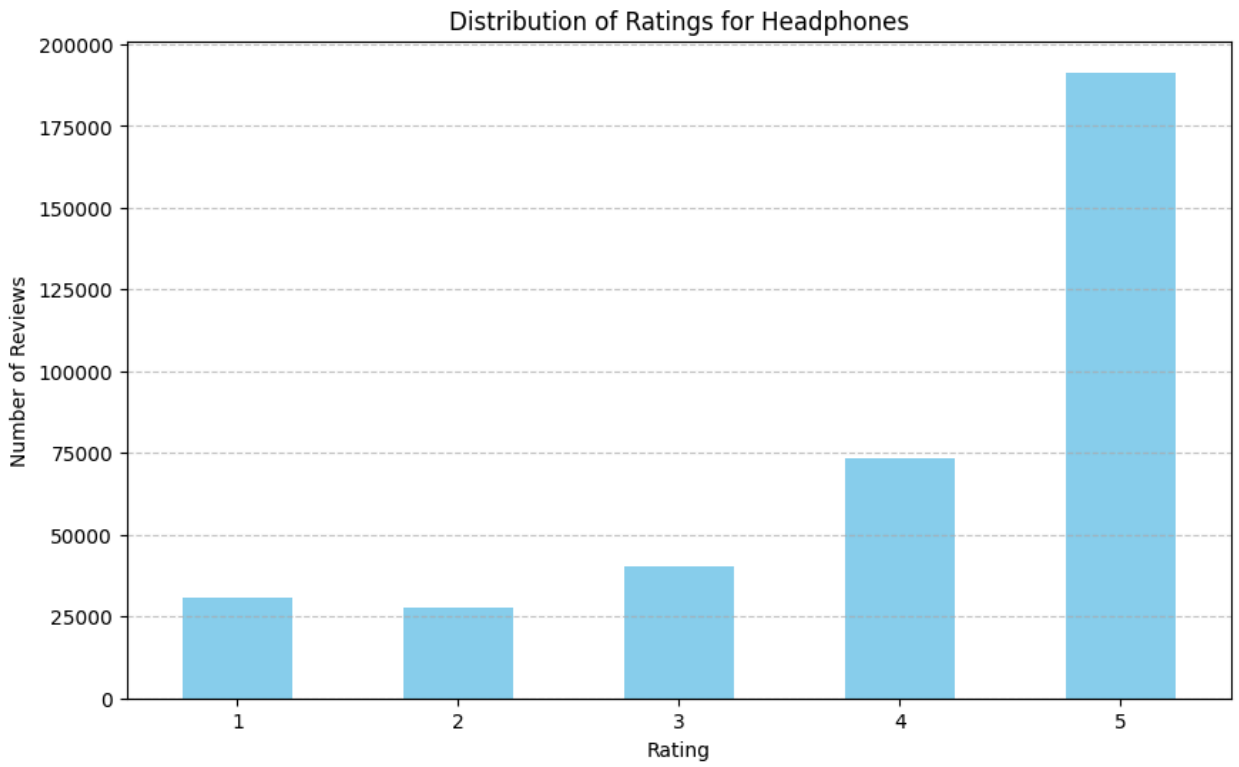
Methodologies:

1. Statistical Analysis: Utilizes Pandas' built-in functions to calculate various statistics, such as the mean, count, and unique value counts. These functions enable efficient data summarization.
2. Conditional Filtering: To differentiate between good and bad ratings, conditional filtering is applied, showcasing the flexibility of Pandas in handling data based on specific criteria.
3. Visualization: Matplotlib is used to plot the distribution of ratings, offering a visual representation of the data that can highlight patterns and insights more intuitively than raw numbers alone.

Results:

- Total number of reviews.
- Average rating score, indicating the overall sentiment of the reviews.
- The count of unique products reviewed, reflecting the diversity of the dataset.
- The number of good and bad ratings, offering insights into customer satisfaction.

```
Number of Reviews: 364050
Average Rating Score: 4.01
Number of Unique Products: 8135
Number of Good Ratings: 305314
Number of Bad Ratings: 58736
Number of Reviews corresponding to each Rating:
overall
1     30910
2     27826
3     40550
4     73349
5    191415
```



PART-5

Approach:

Several NLP techniques were used to prepare the review texts for analysis:

1. Removing HTML tags to get clean text.
2. Normalizing text by converting it to a consistent encoding (ASCII).
3. Removing non-alphabetic characters to focus on textual content.
4. Lemmatizing words to their base or dictionary form.
5. Expanding acronyms using a predefined mapping to ensure clarity.
6. Converting text to lowercase for uniformity.

Assumptions:

No assumptions were made.

Methodologies:

1. BeautifulSoup for HTML Cleaning: Removes HTML tags that could be present in the review texts, ensuring that only the textual content is analyzed.
2. Unidecode for Text Normalization: Converts Unicode characters to their closest ASCII equivalents, addressing encoding inconsistencies.

3. Regular Expressions for Text Cleaning: Filters the text to retain only alphabetic characters, removing numbers and special characters that don't contribute to semantic analysis.
4. NLTK for Tokenization and POS Tagging: Tokenizes the text into words and assigns part-of-speech (POS) tags, which are necessary for accurate lemmatization.
5. Lemmatization with NLTK: Converts words to their base forms based on their POS tags, facilitating the comparison and analysis of semantic content.
6. Acronym Expansion: Expands common acronyms to their full forms based on a custom mapping, enhancing the understandability of the texts.

Results:

- Reviews are cleaned of HTML content and normalized to plain text.
- Textual content is refined by removing non-essential characters and converting words to their lemmatized forms.
- Acronyms are expanded to ensure clarity and uniformity in analysis.
- The processed review texts are stored in a new column within the dataset.

PART-6

Approach:

1. Brand Analysis: Identifies the most and least reviewed headphone brands by mapping ASINs to brands and counting the reviews per brand.
2. Sentiment Analysis: Uses average ratings to determine the most positively reviewed headphone brand and generates word clouds to visualize the common words in good and bad reviews.
3. Trend Analysis: Examines the distribution of ratings and the number of reviews over recent years to identify trends in customer feedback and engagement.

Assumptions:

No assumptions were made.

Methodologies:

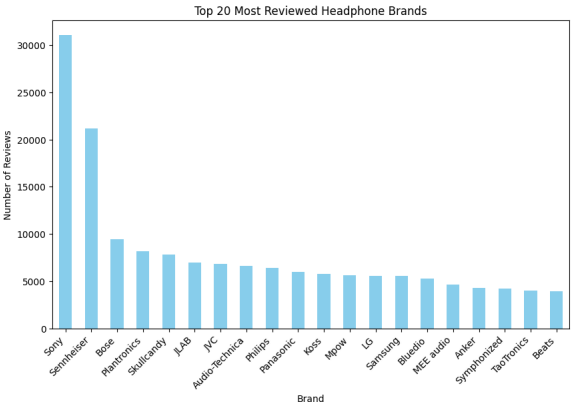
1. Chunk Processing: For efficient memory use, the metadata and reviews datasets are processed in chunks. This approach enables the handling of large datasets that might not fit into memory otherwise.

2. Data Aggregation: Aggregates data to compute statistics such as review counts by brand and average ratings by ASIN, facilitating the identification of top and bottom brands by popularity.
3. Textual Analysis: Generates word clouds for visual representation of common terms in good and bad reviews, providing insights into customer sentiments and preferences.
4. Time Series Analysis: Analyzes reviews over time to identify trends in customer engagement and satisfaction.

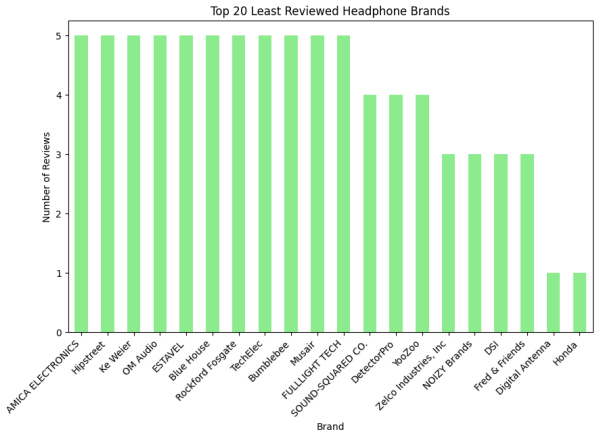
Results:

- Brand Preferences: The analysis of the most and least reviewed brands offers insights into customer preferences, highlighting which brands dominate the market in terms of review count.
- Customer Sentiments: Word clouds for good and bad reviews visually depict prevalent themes in customer feedback, with distinct differences in the focus of positive and negative reviews.
- Market Trends: The analysis over recent years reveals trends in the number of reviews, the year with the maximum reviews, and the year with the highest number of unique customers, indicating shifts in market engagement and possibly in consumer electronics purchasing behavior.

a.

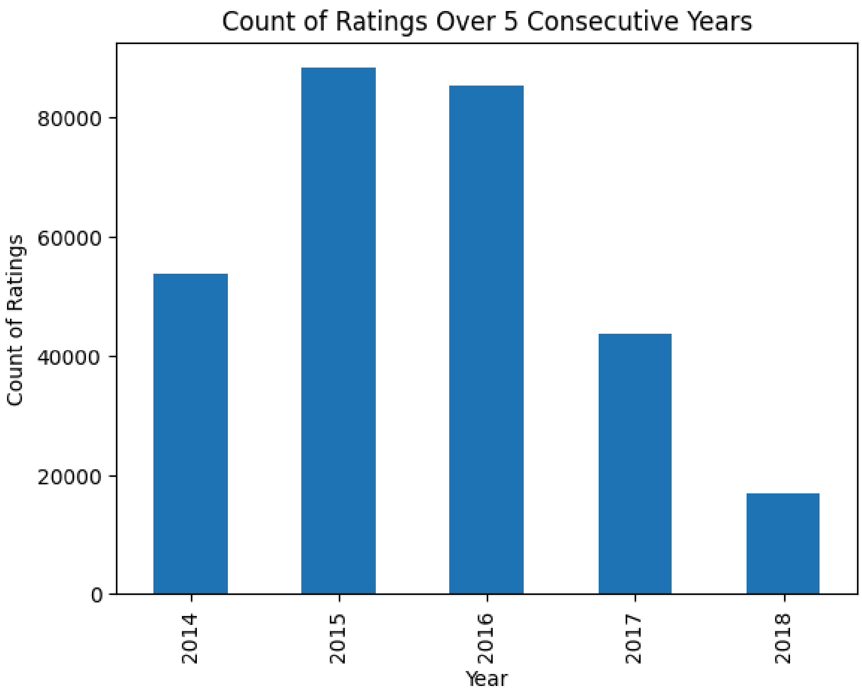


b.



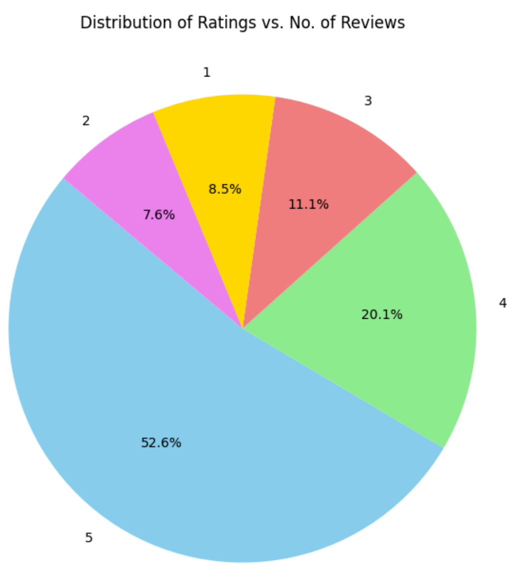
c.

```
The most positively reviewed headphone is from the brand: Sony
```


Count of Ratings Over 5 Consecutive Years
Count of Ratings
0
20000
40000
60000
80000
2014
2015
2016
2017
2018
Year

d.


Word Cloud for Good Ratings
Word Cloud for Bad Ratings


Distribution of Ratings vs. No. of Reviews
1
2
3
4
5
8.5%
7.6%
11.1%
20.1%
52.6%

f.

g. The year with the maximum reviews is: 2015

h. The year with the highest number of customers is: 2015

PART-7

Approach:

1. Data Preparation: The script loads the preprocessed dataset, selecting the 'processed_reviewText' as features (X) and the 'overall' ratings as the target variable (y).
2. Feature Extraction: It employs two vectorization methods to transform the text data into numerical data that machine learning models can work with:
    a. Count Vectorization: Converts text documents to a matrix of token counts.
    b. TF-IDF Vectorization: Converts text documents to a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features.
3. Model Training and Prediction: A Logistic Regression model is trained using the TF-IDF features, chosen for its effectiveness in high-dimensional sparse data, which is typical of text data.
4. Evaluation: The model's accuracy is evaluated by comparing the predicted ratings against the actual ratings in the test set.

Assumptions:

No assumptions were made.

Methodologies:

1. Train-Test Split: The dataset is split into training and testing subsets to evaluate the model's performance on unseen data, using a test size of 20%.
2. Vectorization: The CountVectorizer and TfidfVectorizer from Scikit-learn are used to transform text into a format that the Logistic Regression model can work with.
3. Logistic Regression: This model is chosen for its robustness and efficiency in dealing with binary and multiclass classification problems, including text classification tasks.

Results:

A Logistic Regression model is trained using TF-IDF features of the processed review texts and predicts the ratings on the test set.

```
Accuracy with Logistic Regression: 0.6517786018404066
```

PART-8

Approach:

Each review is classified into one of three categories based on its rating:

- Ratings above 3 are labeled as 'Good'.
- Ratings exactly equal to 3 are labeled as 'Average'.
- Ratings below 3 are labeled as 'Bad

Assumptions:

No assumptions were made.

Methodologies:

Custom Categorization Function: Demonstrates how to integrate domain-specific logic into data preprocessing, converting numerical ratings into categorical labels that are more intuitive for certain types of analysis.

Results:

```
Index(['overall', 'vote', 'verified', 'reviewTime', 'reviewerID', 'asin',
       'style', 'reviewerName', 'reviewText', 'summary', 'unixReviewTime',
       'image', 'processed_reviewText', 'rating_category'],
      dtype='object')
```

PART-9

Approach:

1. Dataset Preparation: The dataset is first organized into features (X) represented by the 'processed_reviewText' and target labels (y) represented by the 'rating_category', which were previously derived from the 'overall' ratings.
2. Train-Test Split: Utilizes Scikit-learn's train_test_split function to partition the dataset into training and test sets, allocating 75% of the data for training and the remaining 25% for testing.
3. Data Saving: Saves the split datasets into separate JSON files ('X_train.json', 'X_test.json', 'y_train.json', 'y_test.json') for both features and labels.

Assumptions:

No assumptions were made.

Methodologies:

- Scikit-learn for Data Splitting: This library is a standard in the machine learning community for tasks such as data partitioning, offering reliable and customizable functions like train_test_split.

Results:

- Training set size: Represents 75% of the original dataset, allocated for training the machine learning models.
- Test set size: Constitutes the remaining 25% of the data, intended for evaluating model performance.

```
Training set size: 273037
Test set size: 91013
```

PART-10

Approach:

1. Model Selection: 5 models were selected, Logistic Regression, Multinomial Naive Bayes, Linear SVC, SGD Classifier, and Passive Aggressive Classifier.
2. Evaluation: The performance of each model is evaluated using the classification_report from Scikit-learn, which provides precision, recall, f1-score, and accuracy for each category ('Good', 'Average', 'Bad'), as well as overall accuracy.

Assumptions:

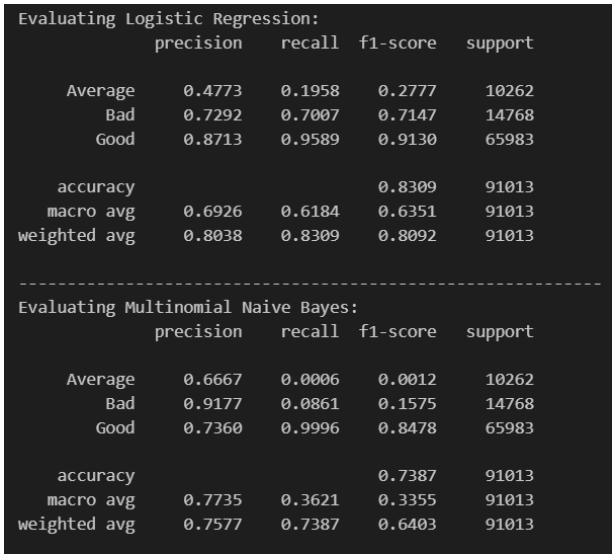
No assumptions were made.

Methodologies:

1. Training and Predicting: Each model is trained on the TF-IDF vectorizer training data and then used to make predictions on the test set.
2. Performance Metrics: Utilizes precision, recall, f1-score, and support to assess the models' performance.

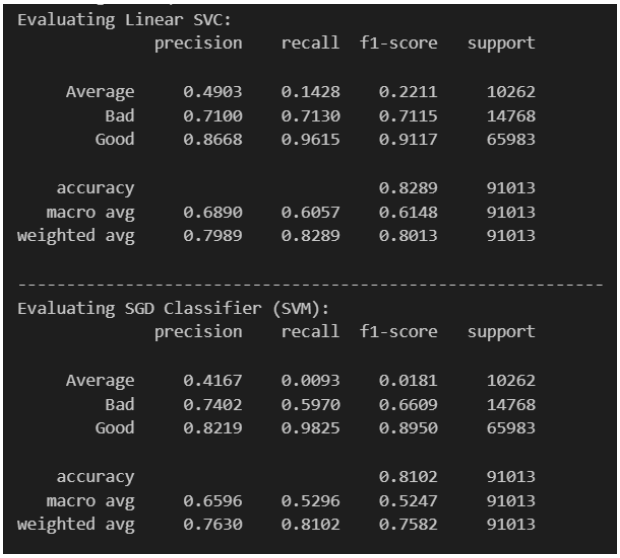
Results:

Precision, recall, f1-score, and support for the categories, along with overall accuracy for each model is output.



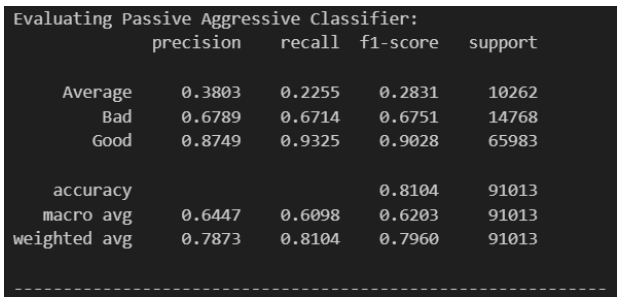
Evaluating Logistic Regression:

	precision	recall	f1-score	support
Average	0.4773	0.1958	0.2777	10262
Bad	0.7292	0.7007	0.7147	14768
Good	0.8713	0.9589	0.9130	65983
accuracy			0.8309	91013
macro avg	0.6926	0.6184	0.6351	91013
weighted avg	0.8038	0.8309	0.8092	91013



Evaluating Linear SVC:

	precision	recall	f1-score	support
Average	0.4903	0.1428	0.2211	10262
Bad	0.7100	0.7130	0.7115	14768
Good	0.8668	0.9615	0.9117	65983
accuracy			0.8289	91013
macro avg	0.6890	0.6057	0.6148	91013
weighted avg	0.7989	0.8289	0.8013	91013



Evaluating Multinomial Naive Bayes:

	precision	recall	f1-score	support
Average	0.6667	0.0006	0.0012	10262
Bad	0.9177	0.0861	0.1575	14768
Good	0.7360	0.9996	0.8478	65983
accuracy			0.7387	91013
macro avg	0.7735	0.3621	0.3355	91013
weighted avg	0.7577	0.7387	0.6403	91013

Evaluating SGD Classifier (SVM):

	precision	recall	f1-score	support
Average	0.4167	0.0093	0.0181	10262
Bad	0.7402	0.5970	0.6609	14768
Good	0.8219	0.9825	0.8950	65983
accuracy			0.8102	91013
macro avg	0.6596	0.5296	0.5247	91013
weighted avg	0.7630	0.8102	0.7582	91013

Evaluating Passive Aggressive Classifier:

	precision	recall	f1-score	support
Average	0.3803	0.2255	0.2831	10262
Bad	0.6789	0.6714	0.6751	14768
Good	0.8749	0.9325	0.9028	65983
accuracy			0.8104	91013
macro avg	0.6447	0.6098	0.6203	91013
weighted avg	0.7873	0.8104	0.7960	91013

PART-11

Approach:

1. Data Preparation: A subset of the first 1000 reviews is selected from the preprocessed dataset and constructs a sparse matrix representation for user-item interactions.
2. Normalization: Ratings are normalized on a per-user basis using min-max scaling to handle the variability in user rating scales.
3. Collaborative Filtering: Both user-user and item-item collaborative filtering models are built. For user-user CF, a similarity matrix between users is computed based on their rating patterns. For item-item CF, a similarity matrix between items is generated based on how similarly they have been rated by users.
4. Evaluation: The models are evaluated using k-fold cross-validation, and the MAE is computed to measure the average deviation of predicted ratings from the actual ratings. The process is repeated for different numbers of similar users/items (k) to observe the impact on prediction accuracy.

Assumptions:
No assumptions were made.

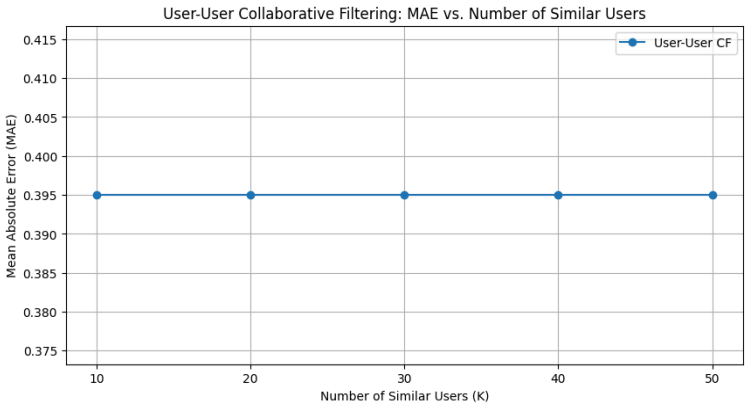
Methodologies:

1. Sparse Matrix Representation: Utilizes scipy.sparse matrices for efficient handling of the sparse user-item interaction data.
2. Min-Max Scaling: Normalizes the ratings for each user/item, ensuring that the rating scale's variability does not bias the similarity calculations.
3. Cosine Similarity: Cosine similarity is computed from scratch between users or items based on the cosine of the angle between their rating vectors.
4. Mean Absolute Error (MAE): Evaluates the accuracy of the predictions, providing a straightforward metric to compare the performance of different models or configurations.

Results:

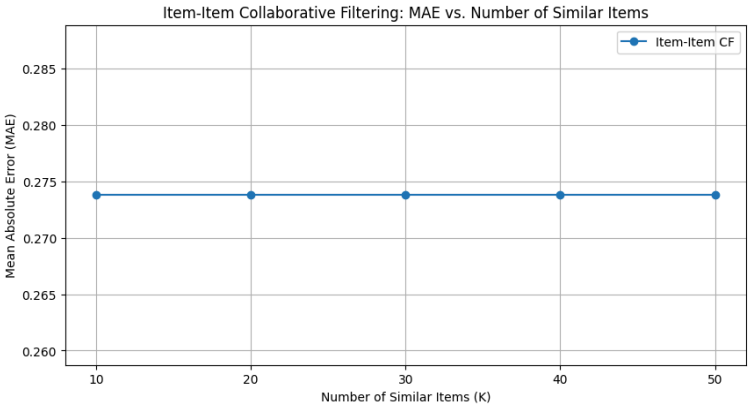
```
MAE for 10 similar users: 0.39498534607037056
MAE for 20 similar users: 0.39498534607037056
MAE for 30 similar users: 0.39498534607037056
MAE for 40 similar users: 0.39498534607037056
MAE for 50 similar users: 0.39498534607037056
```

```
MAE for 10 similar items: 0.27378538725979434
MAE for 20 similar items: 0.27378538725979434
MAE for 30 similar items: 0.27378538725979434
MAE for 40 similar items: 0.27378538725979434
MAE for 50 similar items: 0.27378538725979434
```



- The MAE values are relatively consistent, around 0.39, regardless of the number of similar users considered.

- This suggests that the number of similar users does not significantly affect the prediction accuracy of the user-user CF model in this scenario.



- The MAE values are consistent, averaging around 0.27, and show little variation with different values of K.
- This indicates a stable prediction accuracy for the item-item CF model across the tested values of K.

PART-12

Approach and Methodologies:

1. The ‘groupby’ function is used to aggregate the reviews by the 'asin' field, which represents unique identifiers for the products.
2. The 'overall' rating for each group is summed to get a total rating score per product. These scores are then sorted in descending order to identify the products with the highest cumulative ratings.
3. The head function selects the top 10 products from the sorted list, based on their total rating scores.

Assumptions:

No assumptions were made.

Results:

Lists the top 10 products by their ASINs alongside their respective total rating scores.

asin	
B00001P4XH	1584
B00001P505	832
B00001P4ZH	356
B00000JBHP	271
B00001P4XA	246
B00000JCTO	231
B000001OMI	146
4126895493	144
B00000J1EJ	140
B000001OMR	21