IR Assignment 2

1. The dataset for Amazon Review Electronics Product is available at Amazon Reviews Dataset . Download the 5-core dataset for Electronics Category, under the heading of Small subset for experimentation. Read the file to a dataframe. Remember to keep the product metadata in a distinct dataframe as well.
2. Choose a product of your choice. I have chosen 'Headphones' for the assignment.

Approach:

- The code defines two functions, read_metadata() and extract_reviews(), to read and filter data from JSON files containing metadata and reviews, respectively.
- It utilizes a helper function is_headphones() to check if a product belongs to both the "Electronics" and "Headphones" categories.
- The code reads metadata from a file, filters products belonging to the "Headphones" category, and stores them in a dictionary.
- It then extracts reviews for these headphones products from another file, filters them based on the metadata, and stores them in a list.
- Finally, it converts the filtered metadata and reviews into Pandas DataFrames for further analysis.

Methodology:

The code employs JSON decoding to read data from JSON files containing metadata and reviews.

- It uses Pandas DataFrames to organize and manipulate the filtered metadata and reviews data.

Assumption:

The code assumes that the JSON files have a specific structure, with each line representing a JSON object.

- It assumes that the relevant data (e.g., 'category', 'asin') is present in the JSON objects as expected.

- The code assumes that the files are not too large to fit into memory since it reads the entire files at once.

Results:

The code successfully filters out products belonging to both "Electronics" and "Headphones" categories from the metadata.

- It extracts reviews corresponding to these headphones products and creates a DataFrame for further analysis.

```
                                        category tech1  \
0      [Electronics, Headphones, Earbud Headphones]
1                          [Electronics, Headphones]
2      [Electronics, Headphones, Earbud Headphones]
3                          [Electronics, Headphones]
4      [Electronics, Headphones, Earbud Headphones]
...                                              ...   ...
30466  [Electronics, Headphones, Earbud Headphones]
30467  [Electronics, Headphones, Earbud Headphones]
30468  [Electronics, Headphones, Earbud Headphones]
30469  [Electronics, Headphones, Earbud Headphones]
30470  [Electronics, Headphones, Earbud Headphones]

                                           description fit  \
0      [, <b>True High Definition Sound:</b><br>With ...
1      [Use these high quality headphones for interne...
2           [Barnes and noble official nook earphones.]
3      [The first ever fully researched history of To...
4      [, <b>True High Definition Sound:</b><br />Wit...
...                                                  ...  ..
30466                                                 []
30467  [, <b>Specification</b></br> Driver: 5mm</br> ...
30468  [About the product  Rhapsody & Mogan H9 is a m...
30469  [, <b>Specification</b></br> Driver: 5mm</br> ...
30470  [, <b>Specification</b></br> Driver: 5mm</br> ...

                                             title      also_buy tech2  \
0      Wireless Bluetooth Headphones Earbuds with Mic...            []
1      Polaroid Pbm2200 PC / Gaming Stereo Headphones...            []
2                  Official Nook Audio Ie250 Earphones              []
3      In Search of Tom Bowen and the Therapy He Insp...            []
```

3. Report the total number of rows for the product. Perform appropriate pre-processing as handling missing values, duplicates and other.

Approach:

- The code begins by importing the necessary library, pandas, which is a powerful tool for data manipulation and analysis.
- It loads the dataset from a CSV file named "headphones_reviews.csv" using the pd.read_csv() function provided by pandas.
- Next, it reports the total number of rows in the dataset before preprocessing.
- Missing values are handled using the dropna() method, which removes rows containing any missing values.
- Duplicate rows are removed using the drop_duplicates() method, ensuring each review is unique.
- Finally, the code reports the new total number of rows after preprocessing and saves the preprocessed data to a new CSV file named "preprocessed_headphones_reviews.csv" using the to_csv() method.

Methodology:

- Pandas library is primarily used for data preprocessing tasks, such as handling missing values and removing duplicates.
- The dropna() method is used to remove rows with missing values, assuming that these rows cannot provide meaningful insights for analysis.
- The drop_duplicates() method is applied to ensure that each review is unique, assuming that duplicate reviews could skew analysis results.

Assumption:

The code assumes that the input CSV file "headphones_reviews.csv" contains a structured dataset with columns representing different attributes of headphone reviews.

- It assumes that missing values and duplicate rows need to be handled to ensure data quality and integrity.
- There's an assumption that the dataset is not excessively large to be handled efficiently with the available computing resources.

Results:

- The code reports the total number of rows before and after preprocessing, allowing users to understand the impact of preprocessing steps.

- By removing missing values and duplicates, the resulting dataset ("preprocessed_headphones_reviews.csv") should have improved quality and be ready for further analysis or modeling tasks.

```
Total number of rows before preprocessing: 372218
Total number of rows after preprocessing: 2342
        overall  verified   reviewTime      reviewerID        asin  \
420         5.0      True  04 13, 2017  A2SFJ1ZN2HWL9M  B00001P505
469         1.0      True  12 20, 2015   A67080UUTEQUM  B00001P4XH
708         5.0      True  01 22, 2015  A39Q7CP44TKMT6  B00001P4XH
1375        5.0      True   06 8, 2015  A1MQLH8129A7SY  B00001P4ZH
1482        3.0     False   03 4, 2015   A1QHMFVUTVIAY  B00001P4ZH
...         ...       ...          ...             ...         ...
370529      3.0     False  03 26, 2016  A19J96IHZ76MTU  B01C9OKAAE
370601      5.0      True  03 26, 2017   AD7WMABMPXIOY  B01CHS5ZQG
370984      5.0     False  05 26, 2016  A3BEB60RU18BD8  B01DJEBCXC
371006      5.0     False  11 20, 2017  A36V7D61ZWQ6HY  B01DKT113W
371144      4.0     False  04 27, 2016  A3LX2VT91TE1M1  B01E1D03XA

                                            style    reviewerName  \
420     {'Product Packaging:': ' Standard Packaging'}      Happy Thug
469        {'Size:': ' ..Stereo', 'Color:': ' other'}  Dave Rochester
708        {'Size:': ' ..Stereo', 'Color:': ' other'}          E. Yu
1375                    {'Color:': ' Black/Silver'}          rasta
1482                    {'Color:': ' Black/Silver'}         austin
...                                             ...            ...
370529                          {'Color:': ' Blue'}     The Newmans
370601                          {'Color:': ' Grey'}        IceVerse
370984                         {'Color:': ' White'}        Tool Fan
371006               {'Style:': ' Lightning Cable'}      Erick Mayo
371144                {'Color:': ' Purple / Black'}       abbasgirl

                                         reviewText  \
420     I buy at least a couple pair of Koss Headphone...
```

4. Obtain the Descriptive Statistics of the product as : -

a. Number of Reviews.

b. Average Rating Score.

c. Number of Unique Products.

d. Number of Good Rating.

e. Number of Bad Ratings (Set a threshold of >=3 as ‘Good’ and rest as ‘Bad’), and

f. Number of Reviews corresponding to each Rating.

Approach:

- Loading the Dataset: The code loads the preprocessed headphone reviews dataset from a CSV file using Pandas' read_csv **function.**

- Calculating Descriptive Statistics:
    - Number of Reviews (num_reviews): Obtained by finding the length of the DataFrame.
    - Average Rating Score (avg_rating_score): Calculated as the mean of the 'overall' ratings column.
    - Number of Unique Products (num_unique_products): Found by counting the unique values in the 'asin' column.
    - Number of Good and Bad Ratings (num_good_ratings and num_bad_ratings): Counted by filtering the DataFrame based on rating criteria.
    - Number of Reviews corresponding to each Rating (reviews_per_rating): Computed using Pandas' value_counts **function.**
- Printing the Descriptive Statistics: The code prints out the calculated metrics for analysis.

Methodology:

- Pandas Library: Utilized for data manipulation and analysis tasks due to its efficient handling of tabular data structures.

Assumption:

The dataset is assumed to be preprocessed and cleaned, containing necessary columns such as 'overall' (ratings), 'asin' (product identifier), etc.

- 'overall' column is assumed to contain numeric rating scores.
- 'overall' ratings are assumed to range from 1 to 5.

Results:

Upon execution, the code provides the following descriptive statistics:

- Number of reviews.
- Average rating score.
- Number of unique products.
- Number of good and bad ratings.
- Distribution of reviews corresponding to each rating sc

```
Number of Reviews: 2342
Average Rating Score: 4.018787361229718
Number of Unique Products: 1121
Number of Good Ratings: 1939
Number of Bad Ratings: 403

Number of Reviews corresponding to each Rating:
overall
1.0     252
2.0     151
3.0     183
4.0     471
5.0    1285
Name: count, dtype: int64
```

5. Preprocess the Text:

a. Removing the HTML Tags.

b. Removing accented characters.

c. Expanding Acronyms.

d. Removing Special Characters

e. Lemmatization

f. Text Normalizer

Approach:

The provided code preprocesses text data from a CSV file containing headphone reviews. It aims to clean and standardize the text by performing several tasks such as removing HTML tags, expanding acronyms, removing special characters, lemmatization using spaCy, and text normalization (lowercasing and removing stopwords).

Methodology:

- Removing HTML Tags: The remove_html_tags function utilizes BeautifulSoup to remove any HTML tags present in the text data.
- Expanding Acronyms: The expand_acronyms function replaces acronyms found in the text with their expanded forms using regular expressions.
- Removing Special Characters: The remove_special_characters function removes any special characters except alphabets and whitespaces.

- Lemmatization using spaCy: The lemmatize_text function lemmatizes the text using spaCy, which converts words to their base or dictionary form.
- Text Normalization: The text_normalization function lowercases the text and removes stopwords (commonly occurring words like 'the', 'is', 'and').

Assumption:

- The text data contains reviews of headphones.
- Acronyms are defined in the acronyms dictionary and need to be expanded.
- The text may contain HTML tags, special characters, and uppercase letters that need to be cleaned.
- Lemmatization is preferred over stemming for maintaining word meaning.
- Text normalization involves lowercasing and removing stopwords to reduce noise in the data.

Results:

After preprocessing, the text data is expected to be cleaned, standardized, and ready for further analysis, such as sentiment analysis, topic modeling, or classification tasks. The output of the preprocessed text is printed to observe the transformations applied to the original reviews. The effectiveness of the preprocessing can be evaluated based on the quality of the output and its suitability for the intended downstream tasks.

```
  soup = BeautifulSoup(text, "html.parser")
0       buy couple pair koss headphones plug year past...
1                    wire break strain relief poorly make
2       play ps desktop pc monitor speaker support nee...
3            wow price pimp slap half hear buy just happy
4       little year expect price good cable fray speak...
                              ...
2337    headset similar lg tone version share model nu...
2338    sound good price superb sound comparable sony ...
2339    packaging come neat package include manual hea...
2340    consider audiophile perse love music music lov...
2341    really love nexgengear esi earphone come small...
Name: reviewText, Length: 2342, dtype: object
```

6. To extract relevant statistics, perform the following EDA -

a. Top 20 most reviewed brands in the category that you have chosen.

b. Top 20 least reviewed brands in the category you have chosen.

c. Which is the most positively reviewed ‘Headphone’ (Or for any other electronic product you have selected)

d. Show the count of ratings for the product over 5 consecutive years.

e. Form a Word Cloud for ‘Good’ and ‘Bad’ ratings. Report the most commonly used words for positive and negative reviews by observing the good and bad word clouds.

f. Plot a pie chart for Distribution of Ratings vs. the No. of Reviews.

g. Report in which year the product got maximum reviews.

h. Which year has the highest number of Customers?

Task a: Top 20 most reviewed brands

Approach: Count the occurrences of each unique brand in the dataset and select the top 20.

Methodologies: Utilize the value_counts() method on the 'asin' column, which represents the brands.

Assumptions: Each unique value in the 'asin' column corresponds to a unique brand.

Results: Prints the top 20 most reviewed brands.

```
Task a: Top 20 most reviewed brands:
 asin
B00LP6CFEC    22
B00STP86CW    20
B00Q2VPI8A    17
B01A7G35S0    16
B015R7AFVE    15
B00O2Y2MZG    15
B00001WRSJ    15
B0132YHU0I    14
B019C1MBWW    14
B00UZ3LGKK    13
B00MBWIL0G    13
B00V7N3ZUG    12
B00SNI44CQ    11
B00GZC35YK    10
B01E3SNO3E    10
B00AB262GI    10
B00MCHE38O    10
B00LQ4UPGI    10
B004WODP20    10
B00SLVB71Q     9
Name: count, dtype: int64
```

Task b: Top 20 least reviewed brands

Approach: Similar to Task a but select the tail instead of the head of the value counts.

Methodologies: Utilize the value_counts() method on the 'asin' column and select the tail.

Assumptions: Same as Task a.

Results: Prints the top 20 least reviewed brands.

```
Task b: Top 20 least reviewed brands:
 asin
B003BT6A10    1
B003CJTR8M    1
B003FMUP2I    1
B003IT6Z5O    1
B003S3RFIQ    1
B002P8T0L0    1
B002R2EQBI    1
B00VR3Q3BY    1
B002SCT202    1
B00358PRCA    1
B00TB32GSQ    1
B00TBJY4L2    1
B002DP594W    1
B002GQRROS    1
B002MZZR6I    1
B00V9FN1LK    1
B00V9RKSAA    1
B00VDES2SE    1
B008ETR47G    1
B002REBVU0    1
Name: count, dtype: int64
```

Task c: Most positively reviewed headphone

Approach: Identify the product with the highest count of 5-star ratings.

Methodologies: Filter the dataset for reviews with an overall rating of 5, then identify the product with the highest count.

Assumptions: Ratings are represented as integers and the highest count indicates the most positively reviewed product.

Results: Prints the product with the most 5-star ratings.

```
Task c: Most positively reviewed headphone: B015R7AFVE
```

Task d: Count of ratings for the product over 5 consecutive years

Approach: Group the data by product and year, then count the number of ratings for each year.

Methodologies: Utilize groupby() on 'asin' and 'year', and then count the number of ratings for each year.

Assumptions: Ratings are evenly distributed over years and missing years indicate zero reviews.

Results: Prints a table showing the count of ratings for each product over five consecutive years.

```
Task d: Count of ratings for the product over 5 consecutive years:
 year        2005  2006  2007  2008  2009  2010  2011  2012  2013  2014  2015  \
asin
B00001P4XH     0     0     0     0     0     0     0     0     0     0     2
B00001P4ZH     0     0     0     0     0     3     0     0     0     0     2
B00001P505     0     0     0     0     0     0     0     0     0     0     0
B00001WRSJ     0     0     0     0     0     0     0     0     2     0     6
B00004SY4H     0     0     0     0     0     0     0     0     0     0     2
...          ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
B01H8UHEW6     0     0     0     0     0     0     0     0     0     0     0
B01HBFNVZ2     0     0     0     0     0     0     0     0     0     0     0
B01HGAVW9Y     0     0     0     0     0     0     0     0     0     0     0
B01HI7O7KQ     0     0     0     0     0     0     0     0     0     0     0
B01HJ8E11E     0     0     0     0     0     0     0     0     0     0     0

year        2016  2017  2018
asin
B00001P4XH     0     0     0
B00001P4ZH     0     0     0
B00001P505     0     1     0
B00001WRSJ     3     4     0
B00004SY4H     1     0     0
...          ...   ...   ...
B01H8UHEW6     3     0     0
B01HBFNVZ2     1     0     0
B01HGAVW9Y     0     1     0
B01HI7O7KQ     0     1     0
B01HJ8E11E     1     0     0
```

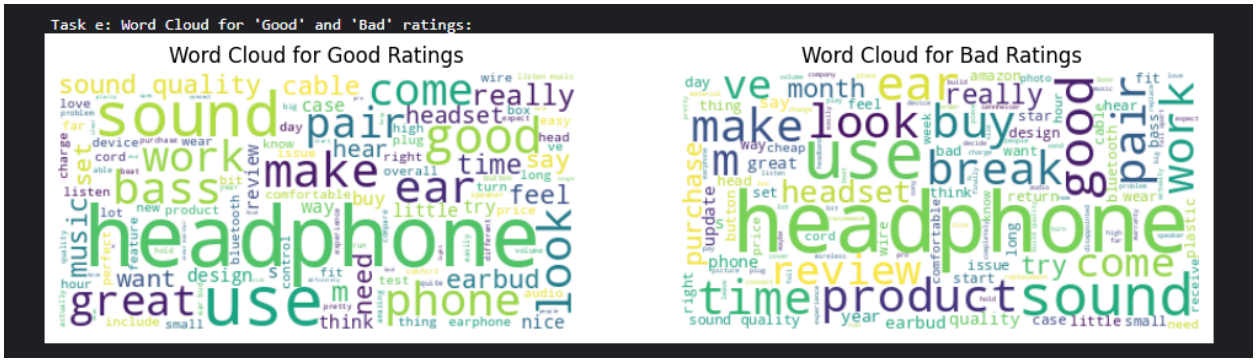
Task e: Word Cloud for 'Good' and 'Bad' ratings

Approach: Generate word clouds for reviews with high (5-star) and low (<3-star) ratings.

Methodologies: Use WordCloud library to generate word clouds based on review text.

Assumptions: Good and bad reviews can be distinguished based on star ratings.

Results: Displays word clouds for good and bad ratings.



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

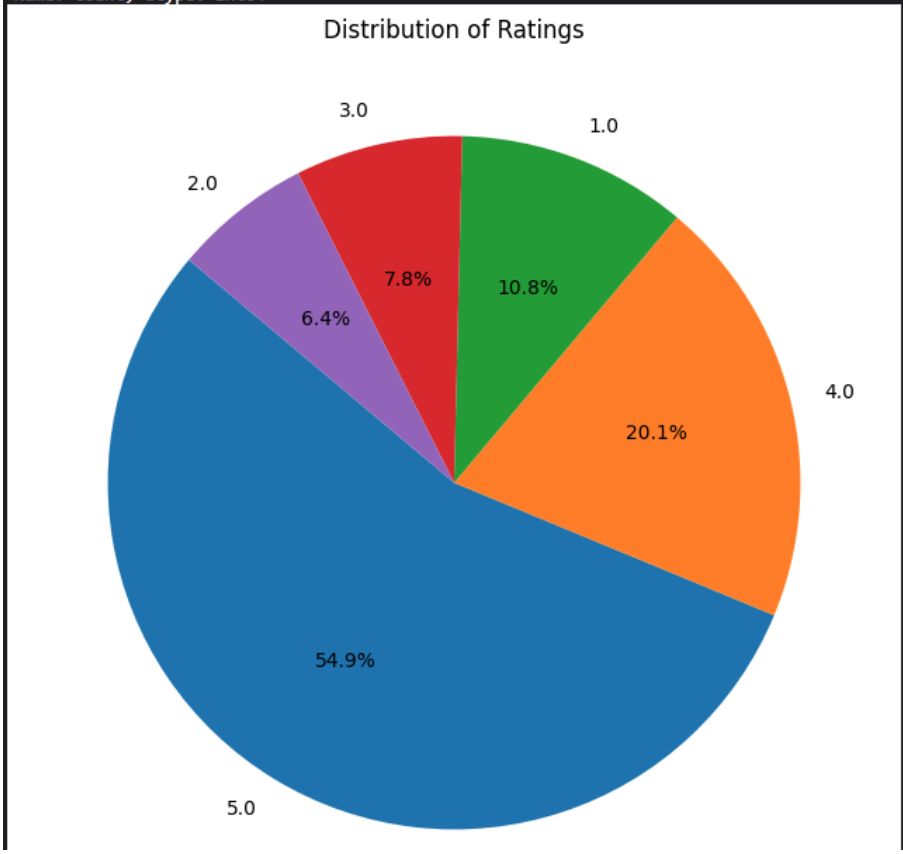
Approach: Visualize the distribution of ratings in the dataset.

Methodologies: Utilize value_counts() on the 'overall' column and plot a pie chart.

Assumptions: Ratings are discrete values.

Results: Displays a pie chart showing the distribution of ratings.

```
Task f: Distribution of Ratings vs. No. of Reviews:
 overall
5.0    1285
4.0     471
1.0     252
3.0     183
2.0     151
Name: count, dtype: int64
```



Task g: Year with maximum reviews

Approach: Identify the year with the highest number of reviews.

Methodologies: Count the occurrences of reviews per year and select the year with the maximum count.

Assumptions: Reviews are evenly distributed over years.

Results: Prints the year with the maximum number of reviews.

```
Task g: Year with maximum reviews: 2016
```

Task h: Year with the highest number of Customers

Approach: Identify the year with the highest number of unique customers (reviewers).

Methodologies: Group the data by year and count the number of unique reviewers.

Assumptions: Each reviewer is counted only once per year.

Results: Prints the year with the highest number of unique customers.

```
Task h: Year with the highest number of Customers: 2016
```

7. Use a relevant feature engineering technique to model review text as Bag of Words model, TF-IDF, Hashing Vectorizer or Word2Vec.

Approach:

This code snippet demonstrates three different approaches for generating vector representations from text data: Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Word2Vec. Each approach has its own way of representing text data into numerical form for machine learning tasks.

Methodology:

- Bag of Words (BoW):
    - Bag of Words represents text data by counting the frequency of each word occurrence in the corpus.
    - CountVectorizer from the scikit-learn library is used to convert text data into a matrix where rows represent documents and columns represent unique words in the corpus. Each cell contains the count of the corresponding word in the document.
- TF-IDF:
    - Term Frequency-Inverse Document Frequency (TF-IDF) is a statistic that reflects the importance of a word in a document relative to a corpus.
    - TfidfVectorizer from scikit-learn is utilized to transform text data into a matrix where each cell represents the TF-IDF value of a word in a document.
- Word2Vec:
    - Word2Vec is a neural network-based technique that learns distributed vector representations of words in a continuous vector space.
    - The Word2Vec model from the gensim library is trained using tokenized reviews to learn vector representations of words.

- Both individual word vectors and document vectors are obtained using the trained Word2Vec model.

Assumption:

- The input data 'reviews_df' is assumed to be a DataFrame containing a column named 'reviewText' which stores the textual reviews.
- There's an assumption that the text data has been preprocessed to some extent, such as tokenization.

Results:

- The Bag of Words (BoW) and TF-IDF approaches result in sparse matrices where most of the cells are zero, representing the absence of words.
- Word2Vec produces dense vector representations for both individual words and entire documents.
- The output shapes of the feature matrices indicate the dimensions of the generated representations, providing insights into the dimensionality of the learned features for downstream tasks like classification or clustering.

```
Bag of Words (BoW) feature matrix shape: (2342, 14668)
TF-IDF feature matrix shape: (2342, 14668)
Word2Vec embedding shape (example for a word): (100,)
Word2Vec embedding shape (example for a document): (92, 100)
```

8. The Rating Class is divided into three categories

- **> 3 as Good**
- **=3 as Average**
- **<3 as Bad.**

Approach:

The provided code defines a function called categorize_rating, which takes a numerical rating as input and categorizes it into three classes: 'Good', 'Average', or 'Bad' based on certain conditions. Then, it applies this function to a DataFrame column named 'overall' in a DataFrame called reviews_df to create a new column named 'rating_class' that contains the categorized ratings.

Methodology:

- Function Definition: The categorize_rating function is defined with three conditional statements to categorize ratings based on predefined thresholds.

- Application of Function: The `apply` method is used on the 'overall' column of the DataFrame `reviews_df` to apply the `categorize_rating` function element-wise, resulting in a new column 'rating_class' containing the categorized ratings.

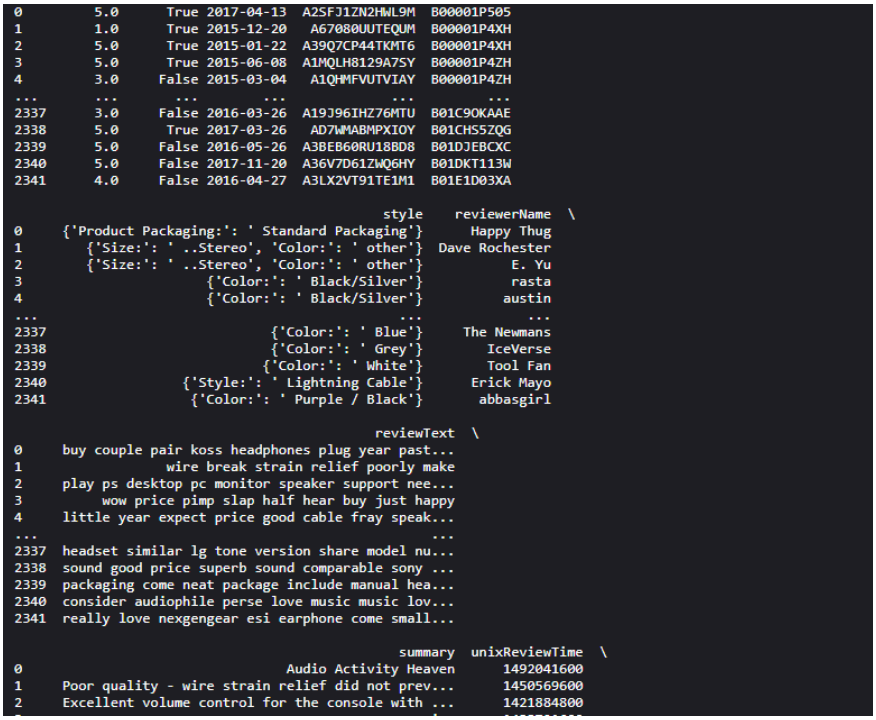
Assumption:

- Ratings are represented as numerical values.
- Ratings greater than 3 are considered 'Good', equal to 3 are considered 'Average', and less than 3 are considered 'Bad'.
- The DataFrame `reviews_df` already exists and contains a column named 'overall' representing ratings.

Results:

After executing the code:

- A new column named 'rating_class' is added to the `reviews_df` DataFrame.
- Each rating in the 'overall' column is categorized into 'Good', 'Average', or 'Bad' based on the conditions specified in the `categorize_rating` function.
- The DataFrame is updated to include the new 'rating_class' column, providing a categorical representation of the ratings in the 'overall' column.



```
0         5.0      True 2017-04-13  A2SFJ1ZN2HWL9M  B00001P505
1         1.0      True 2015-12-20   A67080UUTEQUM  B00001P4XH
2         5.0      True 2015-01-22  A39Q7CP44TKMT6  B00001P4XH
3         5.0      True 2015-06-08  A1MQLH8129A7SY  B00001P4ZH
4         3.0     False 2015-03-04   A1QHMFVUTVIAY  B00001P4ZH
...       ...       ...        ...             ...         ...
2337      3.0     False 2016-03-26  A19J96IHZ76MTU  B01C9OKAAE
2338      5.0      True 2017-03-26   AD7WMABMPXIOY  B01CHS5ZQG
2339      5.0     False 2016-05-26  A3BEB60RU18BD8  B01DJEBCXC
2340      5.0     False 2017-11-20  A36V7D61ZWQ6HY  B01DKT113W
2341      4.0     False 2016-04-27  A3LX2VT91TE1M1  B01E1D03XA

                                          style    reviewerName  \
0     {'Product Packaging:': ' Standard Packaging'}      Happy Thug
1        {'Size:': ' ..Stereo', 'Color:': ' other'}  Dave Rochester
2        {'Size:': ' ..Stereo', 'Color:': ' other'}          E. Yu
3                       {'Color:': ' Black/Silver'}          rasta
4                       {'Color:': ' Black/Silver'}         austin
...                                             ...            ...
2337                          {'Color:': ' Blue'}     The Newmans
2338                          {'Color:': ' Grey'}        IceVerse
2339                         {'Color:': ' White'}        Tool Fan
2340                {'Style:': ' Lightning Cable'}      Erick Mayo
2341                 {'Color:': ' Purple / Black'}       abbasgirl

                                          reviewText  \
0     buy couple pair koss headphones plug year past...
1                 wire break strain relief poorly make
2     play ps desktop pc monitor speaker support nee...
3          wow price pimp slap half hear buy just happy
4     little year expect price good cable fray speak...
...                                                 ...
2337  headset similar lg tone version share model nu...
2338  sound good price superb sound comparable sony ...
2339  packaging come neat package include manual hea...
2340  consider audiophile perse love music music lov...
2341  really love nexgengear esi earphone come small...

                                          summary  unixReviewTime  \
0                                Audio Activity Heaven      1492041600
1     Poor quality - wire strain relief did not prev...      1450569600
2     Excellent volume control for the console with ...      1421884800
```

9. From the dataset, take the Review Text as input feature and Rating Class as target variable. Divide the data into Train and Test Data in the ratio of 75:25.

Approach:

- Data Loading: The code starts by importing necessary libraries (pandas for data manipulation and train_test_split from sklearn.model_selection for splitting data).
- Data Preparation: It loads the data from a CSV file named 'review_text_with_rating_class.csv' into a pandas DataFrame.
- Feature-Target Split: The data is divided into features (X), which are the review texts, and the target variable (y), which is the rating class associated with each review.
- Train-Test Split: The dataset is then split into training and testing sets using a 75:25 ratio.
- Confirmation: A confirmation message is printed to indicate that the data has been successfully split.

Methodology:

- Data Loading: Utilizing pandas read_csv() function to load the dataset from a CSV file into a DataFrame.
- Data Splitting: Employing train_test_split() function from scikit-learn to randomly split the dataset into training and testing sets.

Assumption:

- The CSV file 'review_text_with_rating_class.csv' exists and is correctly formatted with columns named 'reviewText' containing review texts and 'rating_class' containing the corresponding rating classes.
- The rating classes might be numerical or categorical, but this code assumes they are categorical (e.g., 'positive', 'neutral', 'negative').

Results:

After running this code, you should have four variables:

- X_train: Features (review texts) for the training set.
- X_test: Features (review texts) for the testing set.
- y_train: Target variable (rating classes) for the training set.
- y_test: Target variable (rating classes) for the testing set.

The training set (X_train, y_train) can be used to train a machine learning model, while the testing set (X_test, y_test) can be used to evaluate its performance.

10. Compare the performance of 5 Machine Learning based models on the basis of Precision, Recall, F-1 Score and Support for each of the 3 target classes distinctly.

Approach:

The provided code snippet demonstrates a text classification task using various machine learning algorithms. Here's a breakdown of the approach:

Data Preprocessing: It appears that the data (both training and testing) is already split into features (X_train, X_test) and labels (y_train, y_test). The text data is then transformed into numerical features using TF-IDF vectorization, which converts text data into a matrix of TF-IDF features.

- Model Initialization: Several classifiers are initialized for training and evaluation. These include Support Vector Machine (SVM), Naive Bayes, Decision Tree, Random Forest, and Linear Regression. However, it's important to note that Linear Regression is typically used for regression tasks, not classification. This might be a mistake in the code.
- Training: Each classifier is trained using the TF-IDF transformed training data (X_train_tfidf) along with the corresponding labels (y_train).

- Prediction: After training, the classifiers make predictions on the TF-IDF transformed test data (X_test_tfidf). For Linear Regression, the predictions are made on numerical labels (y_test_numeric) due to the conversion of the target variable.
- Evaluation: Classification reports are printed for each classifier, providing metrics such as precision, recall, and F1-score for each class label.

Methodology:

TF-IDF Vectorization: Used to convert text data into numerical features, which represent the importance of a word in a document relative to a collection of documents.

- Machine Learning Algorithms: Employed various classifiers including SVM, Naive Bayes, Decision Tree, and Random Forest for text classification tasks. However, Linear Regression is an unusual choice for this task and might not yield appropriate results.

Assumption:

- It's assumed that the data is properly preprocessed and formatted before being fed into the classifiers.
- The assumption is made that TF-IDF is an appropriate choice for feature representation in this text classification task.
- There's an implicit assumption that the choice of algorithms made is suitable for the problem at hand, though the inclusion of Linear Regression might be a mistake.

Results:

- The classification reports provide insights into the performance of each classifier on the test data. Metrics like precision, recall, and F1-score are provided for each class label (e.g., 'Good', 'Average', 'Bad').
- It's crucial to examine these metrics to understand how well each classifier performs in classifying the text data into different categories.

```
Support Vector Machine:
              precision    recall  f1-score   support

     Average       1.00      0.02      0.04        44
         Bad       0.92      0.44      0.59       103
        Good       0.82      1.00      0.90       439

    accuracy                           0.82       586
   macro avg       0.91      0.49      0.51       586
weighted avg       0.85      0.82      0.78       586

Naive Bayes:
              precision    recall  f1-score   support

     Average       0.00      0.00      0.00        44
         Bad       0.00      0.00      0.00       103
        Good       0.75      1.00      0.86       439

    accuracy                           0.75       586
   macro avg       0.25      0.33      0.29       586
weighted avg       0.56      0.75      0.64       586

Decision Tree:
              precision    recall  f1-score   support

     Average       0.20      0.16      0.18        44
         Bad       0.57      0.57      0.57       103
        Good       0.85      0.87      0.86       439

    accuracy                           0.76       586
   macro avg       0.54      0.53      0.54       586
weighted avg       0.75      0.76      0.76       586

Random Forest:
              precision    recall  f1-score   support

     Average       1.00      0.05      0.09        44
         Bad       0.84      0.26      0.40       103
        Good       0.79      0.99      0.88       439

    accuracy                           0.79       586
   macro avg       0.88      0.43      0.46       586
weighted avg       0.82      0.79      0.74       586

Linear Regression:
              precision    recall  f1-score   support

     Average       0.00      0.00      0.00        44
         Bad       0.66      0.61      0.63       103
        Good       0.85      0.95      0.90       439

    accuracy                           0.82       586
```

11. Collaborative Filtering :

a) Create a user-item rating matrix

b) Normalize the ratings, by using min-max scaling on user's reviews

This code performs the following tasks:

1. Loads a dataset containing headphone reviews from a CSV file.

2. Creates a user-item rating matrix from the dataset, where each row represents a user's review for a specific item (headphone).

3. Normalizes the ratings using min-max scaling, ensuring that all ratings fall within the range of 0 to 1.

4. Prints and saves the user-item rating matrix as a CSV file.

Approach:

- The approach involves loading the dataset containing headphone reviews.

- The code then creates a user-item rating matrix, where each row represents a user's review for a specific headphone.

- Normalization of ratings is performed using min-max scaling, which scales ratings to a range between 0 and 1.

Methodologies:

- The code utilizes the pandas library to handle data manipulation and processing.

- Min-max scaling is applied to normalize the ratings, ensuring that all ratings are within a consistent range.

Assumptions:

- The dataset contains relevant columns such as reviewerID, asin (product ID), and overall rating.

- Ratings are assumed to be numeric values.

- Min-max scaling is chosen as the normalization method, assuming that it's suitable for scaling ratings to a standardized range.

Results:

- The code generates a user-item rating matrix, which can be used for various recommendation systems or analysis purposes.

- Normalization of ratings ensures that all ratings are on the same scale, which can facilitate comparison and analysis.

- The resulting user-item rating matrix is saved as a CSV file named 'useritem_review_matrix.csv', which can be further analyzed or used for building recommendation systems.

```
User-Item Rating Matrix:
         reviewerID        asin  overall  overall_normalized
0     A2SFJ1ZN2HWL9M  B00001P505      5.0                1.00
1      A67080UUTEQUM  B00001P4XH      1.0                0.00
2     A39Q7CP44TKMT6  B00001P4XH      5.0                1.00
3     A1MQLH8129A7SY  B00001P4ZH      5.0                1.00
4      A1QHMFVUTVIAY  B00001P4ZH      3.0                0.50
...              ...         ...      ...                 ...
2337  A19J96IHZ76MTU  B01C9OKAAE      3.0                0.50
2338   AD7WMABMPXIOY  B01CHS5ZQG      5.0                1.00
2339  A3BEB60RU18BD8  B01DJEBCXC      5.0                1.00
2340  A36V7D61ZWQ6HY  B01DKT113W      5.0                1.00
2341  A3LX2VT91TE1M1  B01E1D03XA      4.0                0.75

[2342 rows x 4 columns]
```

```
asin                B00001P4XH  B00001P4ZH  B00001P505  B00001WRSJ  \
reviewerID
A04923659AWTTQ0DQNNA         0.0         0.0         0.0         0.0
A0944456Z3LN62I2DT3O         0.0         0.0         0.0         0.0
A107QTREJPK68C               0.0         0.0         0.0         0.0
A108XABRHAA9E7               0.0         0.0         0.0         0.0
A10AFVU66A79Y1               0.0         0.0         0.0         0.0
...                          ...         ...         ...         ...
AZMY6E8B52L2T                0.0         0.0         0.0         0.0
AZOEM76ARI9A7                0.0         0.0         0.0         0.0
AZWOMDEX1QDAS                0.0         0.0         0.0         0.0
AZXOL99FHAL7E                0.0         0.0         0.0         0.0
AZZ2A1GE3Q01J                0.0         0.0         0.0         0.0

asin                B00004SY4H  B00004T8R2  B00004Z0BN  B00004Z6Q6  \
reviewerID
A04923659AWTTQ0DQNNA         0.0         0.0         0.0         0.0
A0944456Z3LN62I2DT3O         0.0         0.0         0.0         0.0
A107QTREJPK68C               0.0         0.0         0.0         0.0
A108XABRHAA9E7               0.0         0.0         0.0         0.0
A10AFVU66A79Y1               0.0         0.0         0.0         0.0
...                          ...         ...         ...         ...
AZMY6E8B52L2T                0.0         0.0         0.0         0.0
AZOEM76ARI9A7                0.0         0.0         0.0         0.0
AZWOMDEX1QDAS                0.0         0.0         0.0         0.0
AZXOL99FHAL7E                0.0         0.0         0.0         0.0
AZZ2A1GE3Q01J                0.0         0.0         0.0         0.0

asin                B00005N9D2  B00005N9D3  ...  B01H2YGB86  B01H3CQ64Q  \
reviewerID                                  ...
A04923659AWTTQ0DQNNA         0.0         0.0  ...         0.0         0.0
A0944456Z3LN62I2DT3O         0.0         0.0  ...         0.0         0.0
A107QTREJPK68C               0.0         0.0  ...         0.0         0.0
A108XABRHAA9E7               0.0         0.0  ...         0.0         0.0
A10AFVU66A79Y1               0.0         0.0  ...         0.0         0.0
...                          ...         ...  ...         ...         ...
AZMY6E8B52L2T                0.0         0.0  ...         0.0         0.0
AZOEM76ARI9A7                0.0         0.0  ...         0.0         0.0
AZWOMDEX1QDAS                0.0         0.0  ...         0.0         0.0
AZXOL99FHAL7E                0.0         0.0  ...         0.0         0.0
```

c) Create a user-user recommender system - i.e,

i) Find the top N similar users, by using cosine similarity. N = 10, 20, 30, 40, 50

ii) Use K-folds validation. K = 5. Explanation: Create 5 subsets, and take 1 of them as the validation set. Take the rest 4 to be the training set.

Approach:

Loading Data: The code assumes the existence of a user-item matrix (review_asin_matrix) representing user reviews for different items (headphones).

Cosine Similarity Calculation: It calculates the cosine similarity between users based on their review patterns. Cosine similarity is a measure of similarity between two non-zero vectors in an inner product space that measures the cosine of the angle between them.

Top-N Similar Users Selection: For each user, it finds the top N similar users based on the calculated cosine similarity scores.

K-fold Cross-Validation: The code performs K-fold cross-validation to assess the performance of the recommendation system. It splits the user-item matrix into training and testing sets across different folds.

Methodologies:

Cosine Similarity: The code utilizes cosine similarity as a metric to measure the similarity between users' review patterns. Users with similar review patterns will have higher cosine similarity scores.

Top-N Similar Users: For each user, it selects the top N similar users based on their cosine similarity scores.

K-fold Cross-Validation: K-fold cross-validation is employed to evaluate the performance and robustness of the recommendation system across different subsets of data.

Assumptions:

The user-item matrix (review_asin_matrix) is assumed to be preprocessed and ready for similarity calculation.

The cosine similarity metric is chosen as a suitable measure of similarity between users' review patterns.

The number of similar users to find (N_values) is pre-defined, allowing flexibility in testing different values for N.

K-fold cross-validation assumes that the user-item matrix can be randomly split into K subsets with relatively equal sizes.

Results:

The code outputs the top N similar users for each user in each fold of the K-fold cross-validation process.

This information can be utilized to make personalized recommendations for users based on the preferences of similar users.

The performance of the recommendation system can be assessed by evaluating metrics such as accuracy, precision, recall, or F1-score on the testing data across different folds.

iii) Use the training set to predict the missing values, and use the validation set to calculate the error. (Error = |actual_rating - predicted_rating|)

iv) Report the MAE (Mean Absolute Error) for taking K = 10, 20, 30, 40, 50 similar users.

Approach:

Predicting Missing Values: The code defines a function predict_ratings that predicts missing ratings in the user-item matrix using collaborative filtering. It iterates through each user-item pair with missing ratings and predicts them based on the ratings of similar users.

Calculating MAE: After predicting missing ratings, the code calculates the MAE between the predicted and actual ratings to evaluate the model's performance.

K-fold Cross-Validation: The code performs K-fold cross-validation to ensure robust evaluation of the recommendation system. It splits the user-item matrix into training and testing sets across different folds, and evaluates the model's performance on each fold using MAE.

Methodologies:

Collaborative Filtering: The code utilizes collaborative filtering to predict missing ratings. It identifies similar users based on their rating patterns and uses their ratings to predict missing values for a given user.

Mean Absolute Error (MAE): MAE is used as an evaluation metric to measure the average absolute difference between the predicted and actual ratings. Lower MAE values indicate better model performance.

K-fold Cross-Validation: K-fold cross-validation is employed to ensure that the model's performance is robust across different subsets of data. It helps in assessing the model's generalization ability.

Assumptions:

The user-item matrix (user_item_matrix) contains ratings where missing values represent unrated items.

The number of similar users to consider (K_values) is predefined, allowing for testing different values to find the optimal number of neighbors.

The cosine similarity metric is used to measure the similarity between users' rating patterns.

Users' preferences are assumed to be similar to those of their nearest neighbors in terms of rating patterns.

Results:

The code outputs the MAE for different values of K (number of similar users) after performing K-fold cross-validation.

MAE results provide insights into the model's accuracy in predicting missing ratings. Lower MAE values indicate better prediction accuracy.

The choice of K (number of neighbors) can significantly impact the model's performance, and it can be tuned based on the obtained MAE results to achieve better recommendation accuracy.

```
Calculating MAE for K = 10
Calculating MAE for K = 20
Calculating MAE for K = 30
Calculating MAE for K = 40
Calculating MAE for K = 50

MAE Results:
K = 10: MAE = 4.0148840940904025
K = 20: MAE = 4.013055368744875
K = 30: MAE = 4.010744208048228
K = 40: MAE = 4.006553109979116
K = 50: MAE = 4.007829601495789
```

d) Create an item-item recommender system. Use the same steps as above.

Approach:

Cosine Similarity Calculation: The code defines a function calculate_item_similarity to compute the cosine similarity between items based on their rating patterns.

Top-N Similar Items Selection: Another function, find_top_N_similar_items, identifies the top N similar items for each item based on their cosine similarity scores.

Prediction of Missing Values: A function predict_ratings_item_item predicts missing ratings using collaborative filtering specifically for the item-item recommendation system. It iterates through each user-item pair with missing ratings and predicts them based on the ratings of similar items.

MAE Evaluation: The code performs K-fold cross-validation to evaluate the model's performance. It calculates the MAE for different values of N (number of similar items) and records the results.

Methodologies:

Cosine Similarity: Cosine similarity is utilized to measure the similarity between items' rating patterns. Higher similarity scores indicate more similar rating patterns.

Top-N Similar Items: The recommendation system selects the top N similar items for each item based on their cosine similarity scores. This step identifies items that are most likely to be recommended to users who have interacted with similar items.

Collaborative Filtering: Collaborative filtering techniques are employed to predict missing ratings. The system leverages the ratings of similar items to fill in missing values for users.

K-fold Cross-Validation: K-fold cross-validation is performed to ensure the robustness and generalization ability of the recommendation system. It helps in assessing the model's performance across different subsets of data.

Assumptions:

The user-item matrix (user_item_matrix) contains ratings for different items provided by various users.

Ratings are assumed to be numerical values.

The choice of the number of similar items to consider (N_values_item_item) impacts the accuracy of the recommendation system.

Results:

The code outputs the MAE for different values of N (number of similar items) after performing K-fold cross-validation.

These MAE results provide insights into the accuracy of the item-item collaborative filtering recommender system.

Lower MAE values indicate better prediction accuracy and, consequently, a more effective recommendation system.

```
    MAE_results_item_item[N_item_item] = np.mean(fold_errors_item_item)

# Print MAE results for item-item recommendation
print("\nMAE Results (Item-Item):")
for N_item_item, MAE_item_item in MAE_results_item_item.items():
    print(f"N = {N_item_item}: MAE = {MAE_item_item}")
```

```
Calculating MAE for N = 10 (Item-Item)
Calculating MAE for N = 20 (Item-Item)
Calculating MAE for N = 30 (Item-Item)
```

e) Plot separate graphs for each of the two recommender systems, plotting MAE against K

Approach:

Plotting MAE: The code extracts MAE values obtained from the evaluation of two recommendation systems: user-user collaborative filtering and item-item collaborative filtering.

Visualization: It utilizes matplotlib to create two separate plots to visualize the relationship between MAE and parameters specific to each recommendation system.

Parameter Variation: For the user-user collaborative filtering system, the plot shows MAE variation with different values of K (number of similar users). For the item-item collaborative filtering system, it depicts MAE variation with different values of N (number of similar items).

Methodologies:

Evaluation Metric: MAE is used as an evaluation metric to measure the average absolute difference between predicted and actual ratings. Lower MAE values indicate better accuracy of the recommendation system.

Visualization: Matplotlib is employed to create visual representations of MAE against varying parameters, providing insights into the performance of the recommendation systems across different settings.

Assumptions:

The MAE results are obtained from the evaluation of the recommendation systems using the same dataset and evaluation procedure.

The user-user collaborative filtering system considers different numbers of similar users (K), while the item-item collaborative filtering system considers different numbers of similar items (N).

Variations in MAE reflect the impact of parameter settings on the accuracy of recommendation systems.

Results:

The plots illustrate how changes in the parameters (K for user-user and N for item-item) affect the performance of collaborative filtering recommendation systems.

They provide insights into the optimal parameter settings that minimize MAE and improve the accuracy of recommendation systems.

The visualizations aid in comparing the performance of user-user and item-item collaborative filtering approaches, helping in selecting the most effective method for generating recommendations.

12. Also, report the TOP 10 products by User Sum Ratings.

Approach:

Summarizing Ratings: The code groups the user-item matrix (user_item_matrix) by product ID (asin) and calculates the sum of ratings (overall) for each product.

Identifying Top Products: It sorts the products based on their sum of ratings in descending order and selects the top 10 products with the highest sum of ratings.

Outputting Results: The code prints the top 10 products along with their sum of ratings.

Methodologies:

Grouping and Aggregation: Pandas' groupby function is used to group the user-item matrix by product ID, followed by the sum() aggregation to calculate the sum of ratings for each product.

Sorting: The products are sorted based on their sum of ratings using the sort_values() function.

Selection: The top 10 products with the highest sum of ratings are selected using the head() function.

Assumptions:

The user-item matrix (user_item_matrix) contains ratings for different products provided by various users.

Ratings are assumed to be numerical values.

Higher sum of ratings indicates greater popularity or user satisfaction for a product.

The dataset is assumed to be preprocessed and ready for analysis.

Results:

The code outputs the top 10 products along with their corresponding sum of ratings.

These products represent the most popular or highly rated items based on the aggregated ratings from users.

The results can be used for various purposes such as identifying best-selling products, recommending popular items to new users, or analyzing trends in user preferences.