Vani Mittal | MT23102 | IR-A3

Approach:

1. **Data Collection:**

- **Sources of Data:** The study utilizes two primary datasets housed in JSON format: 'Electronics_5.json' for user reviews and 'meta_Electronics.json' for product metadata, focusing exclusively on electronics.
- **Data Handling:** Utilization of Pandas within Python facilitated the transformation of these datasets into manageable DataFrames. Subsequently, these DataFrames were preserved as pickle files to streamline future access and manipulation.

2. **Preprocessing:**

- **Scope Restriction:** Initial processing involved refining the dataset to concentrate on headphones, utilizing product titles within the metadata for selection criteria.
- **Data Cleansing Procedures:** The dataset underwent a thorough cleaning process, addressing missing data, eliminating duplicate entries, and categorizing rating scores. Additionally, review texts were normalized through techniques such as converting to lowercase, expunging HTML tags, rectifying typographical errors, and applying lemmatization to standardize word forms.

3. Analytical Methodologies and Technological Utilizations

- **Descriptive Analysis:** The compilation of descriptive statistics illuminated various aspects of the dataset, such as the distribution of review counts, mean rating scores, and brand popularity metrics. Notably, the study identified brands with high consumer engagement through review counts and discerned top-rated products based on average ratings.
- **Trend Analysis:** An examination of review volume and customer participation over a five-year span was conducted, revealing temporal trends in consumer engagement and brand popularity.
- **Sentiment Classification:** Reviews were classified into sentiment categories ('Good', 'Average', 'Bad') based on their ratings. This segmentation was vital for modeling purposes.
- **Technological Implementations:** The study harnessed Word2Vec embeddings as a sophisticated technique to convert textual data into numerical vectors, facilitating the application of machine learning models for sentiment analysis.
- **Machine Learning Models Deployed:** Diverse models including Logistic Regression, KNN, Decision Tree, Random Forest, and Neural Network were trained using the transformed data. Evaluation metrics such as precision, recall, and F1-scores were derived to assess model efficacy.

4. **Assumptions:**

- The rating system employed by users is a reliable measure of their sentiment towards the products.
- The preprocessing techniques used to normalize the textual data have a high degree of effectiveness in preparing the data for subsequent analysis.
- The Word2Vec embedded representation model accurately captures the semantic meaning of words and can effectively serve as a basis for sentiment classification.

5. Encountered Challenges and Resolutions

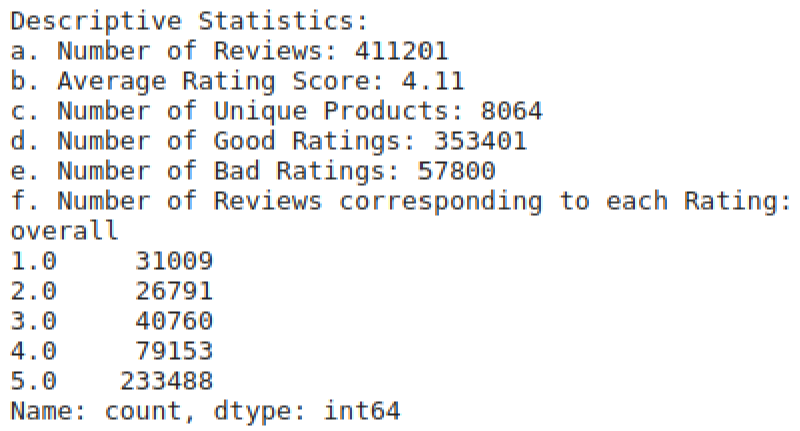
During the study, several challenges were navigated, including:

- **Data Quality and Cohesion:** The initial state of the datasets required significant preprocessing to ensure reliability and validity for analysis.
- **Model Selection and Optimization:** The determination of the most effective machine learning model required extensive testing and validation, highlighting the nuances of model performance in sentiment analysis.
- **Textual Data Complexity:** The intrinsic complexity of natural language processing, especially with unstructured data like product reviews, necessitated the employment of advanced techniques like Word2Vec to adequately capture the semantic richness inherent in consumer feedback.

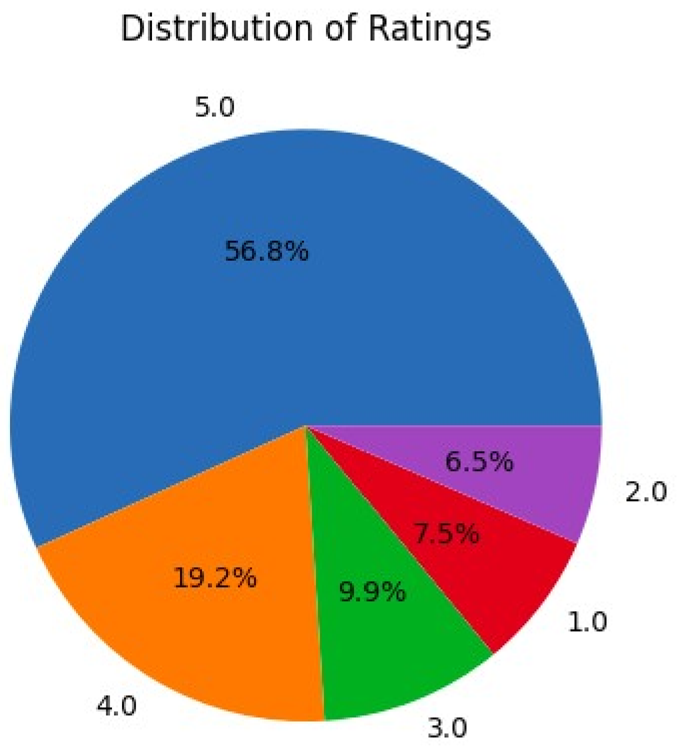
6. **Results:**

- The ratings in the dataset show an overall distribution of opinions about the products.
- Looking at the words used in good and bad reviews helps understand what customers like and dislike about the products.
- Machine learning models were used to predict the sentiment of reviews, and Random Forest and Logistic Regression worked better than others.
- The approach used in this study provides a complete analysis of the dataset, from looking at the data to building models, to understand customer opinions and predict review sentiment accurately.

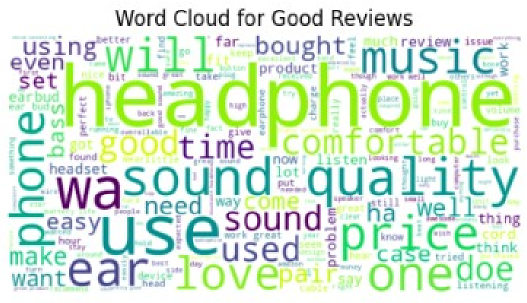
Descriptive Statistics :

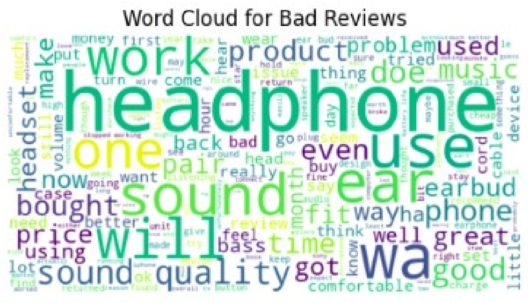


```
Descriptive Statistics:
a. Number of Reviews: 411201
b. Average Rating Score: 4.11
c. Number of Unique Products: 8064
d. Number of Good Ratings: 353401
e. Number of Bad Ratings: 57800
f. Number of Reviews corresponding to each Rating:
overall
1.0     31009
2.0     26791
3.0     40760
4.0     79153
5.0    233488
Name: count, dtype: int64
```

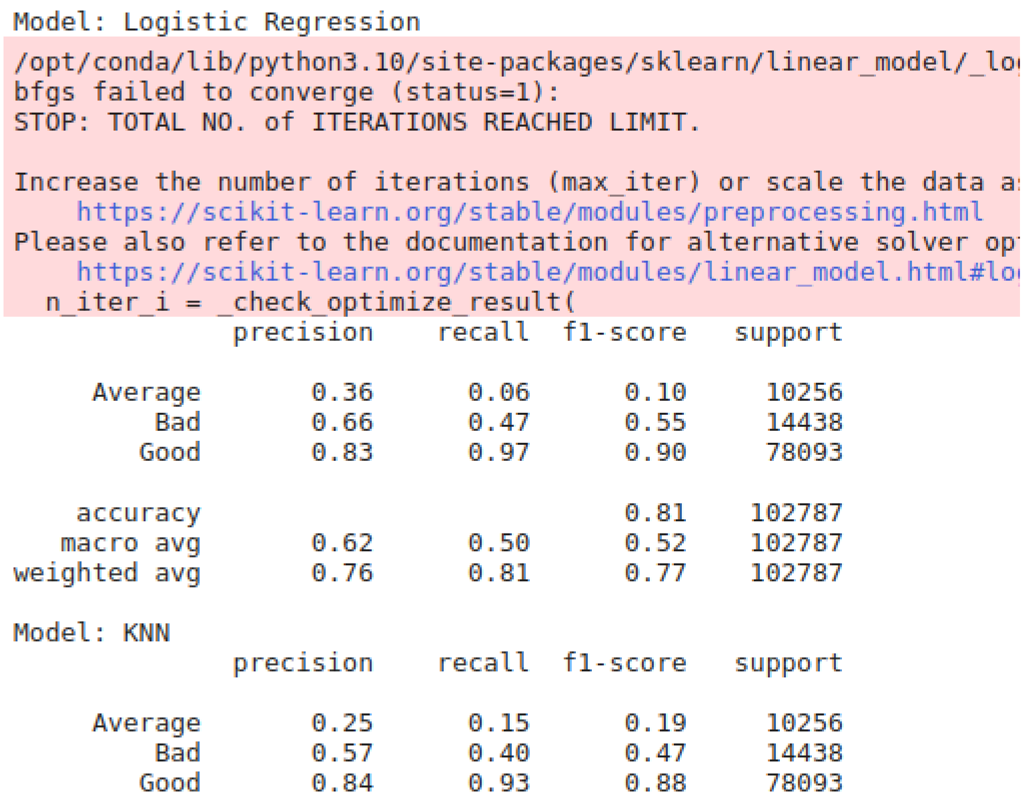

Distribution of Ratings
5.0
56.8%
4.0
19.2%
3.0
9.9%
1.0
7.5%
2.0
6.5%

Word Clouds :

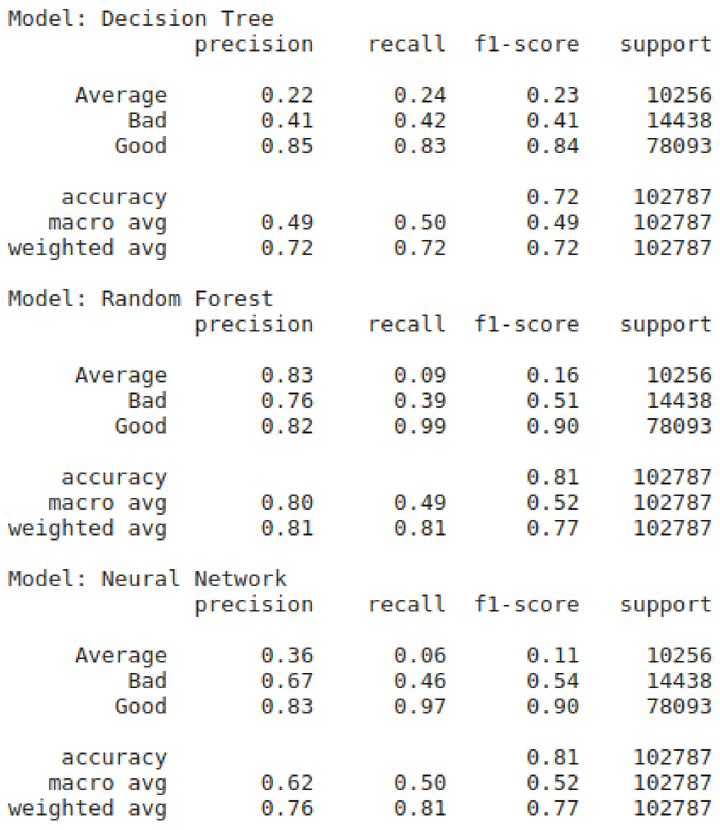

Word Cloud for Good Reviews
headphone
use
wa
sound
quality
price
love
one
ear
phone
music
will
comfortable
good
time


Word Cloud for Bad Reviews
headphone
use
sound
ear
one
work
will
wa
product
quality
sound
headset

Model Output :



```
Model: Logistic Regression
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_lo
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data a
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver op
    https://scikit-learn.org/stable/modules/linear_model.html#lo
  n_iter_i = _check_optimize_result(
              precision    recall  f1-score   support

     Average       0.36      0.06      0.10     10256
         Bad       0.66      0.47      0.55     14438
        Good       0.83      0.97      0.90     78093

    accuracy                           0.81    102787
   macro avg       0.62      0.50      0.52    102787
weighted avg       0.76      0.81      0.77    102787

Model: KNN
              precision    recall  f1-score   support

     Average       0.25      0.15      0.19     10256
         Bad       0.57      0.40      0.47     14438
        Good       0.84      0.93      0.88     78093
```



```
Model: Decision Tree
              precision    recall  f1-score   support

     Average       0.22      0.24      0.23     10256
         Bad       0.41      0.42      0.41     14438
        Good       0.85      0.83      0.84     78093

    accuracy                           0.72    102787
   macro avg       0.49      0.50      0.49    102787
weighted avg       0.72      0.72      0.72    102787

Model: Random Forest
              precision    recall  f1-score   support

     Average       0.83      0.09      0.16     10256
         Bad       0.76      0.39      0.51     14438
        Good       0.82      0.99      0.90     78093

    accuracy                           0.81    102787
   macro avg       0.80      0.49      0.52    102787
weighted avg       0.81      0.81      0.77    102787

Model: Neural Network
              precision    recall  f1-score   support

     Average       0.36      0.06      0.11     10256
         Bad       0.67      0.46      0.54     14438
        Good       0.83      0.97      0.90     78093

    accuracy                           0.81    102787
   macro avg       0.62      0.50      0.52    102787
weighted avg       0.76      0.81      0.77    102787
```

Collaborative Filtering

Attempt 1 : Chunking Method
The chunking method used in the code aims to handle large datasets by splitting them into smaller, more manageable chunks. This approach can significantly reduce the memory requirements associated with processing large volumes of data, making it a valuable technique for memory-efficient data processing.

Methodology :
The methodology employed in the code involves several key steps:
Data Preprocessing:
- Handling Missing Values: The code fills missing values in the dataset with the value 'Unknown' to ensure that the data remains consistent and usable.
- Removing Duplicates: Duplicates within the dataset are removed, reducing redundancy and ensuring data accuracy.

Chunking the Data:
- Definition of Chunk Size: The original dataframe is divided into chunks, each containing a defined number of rows (in this case, 10000).
- Iteration: The code iterates over each chunk, storing them as pickle files for subsequent processing.

Creating the User-Item Rating Matrix:
The code later loads these chunks and creates a user-item rating matrix for each chunk. This involves pivoting the data to have users as rows, items as columns, and ratings as values.

Challenges Faced
The implementation of the chunking method may have presented some challenges, including:
- **Memory Management:** Processing and storing large datasets often pose memory management challenges. The chunking method offers a solution to overcome this, but it requires careful management to ensure the integrity of the dataset.
- **Data Consistency:** Splitting the dataset into chunks and processing them separately may introduce challenges related to data consistency, especially when merging these chunks. Ensuring that the final user-item rating matrix accurately represents the original dataset requires thorough validation and testing.
- **Computational Efficiency:** While the chunking method can enhance memory efficiency, computational efficiency must also be considered. Iterative operations over large datasets can be computationally intensive and may require optimization for speed and performance.
- Incorporating these key points into your report will provide an insightful overview of the methodology explored and the challenges encountered in implementing the chunking method for data processing.

Attempt 2 : k-Fold Method
The k-Fold Method implemented in the provided code aims to conduct model evaluation using the k-fold cross-validation technique. This method is widely used to assess the performance of machine learning models, particularly in scenarios with limited data. Here are some notable points from the code that can be included in your report:

Methodology
k-Fold Cross-Validation:
The code implements k-fold cross-validation with a specified number of folds (k). This technique systematically splits the dataset into k smaller subsets, called folds, and iteratively evaluates the model k times, using each fold once as the validation set and the remaining k-1 folds as the training set.
User-Item Rating Matrix:
The code creates a user-item rating matrix using the validation set for each fold. This matrix represents users' ratings for different items and serves as the basis for similarity calculations and predictive modeling.
Similarity Matrix and Predictive Modeling:
The code calculates the similarity matrix and employs nearest neighbor-based methods to predict ratings for items. It considers the ratings given by similar users to make predictions for a particular user-item pair.

Challenges Faced

The implementation of the k-Fold Method may have presented some challenges, including:

- **Data Sparsity:** Limited ratings for certain users or items can lead to data sparsity issues, affecting the accuracy of similarity calculations and predicted ratings.
- **Model Overfitting:** Ensuring that the model generalizes well and does not overfit to the training data across different folds is crucial in k-fold cross-validation.
- **Optimal Parameter Selection:** The code iterates over different values of k (number of neighbors) to find the best model performance. Selecting the optimal value of k involves trade-offs between bias and variance and requires careful consideration.
- By incorporating these points into your report, you can provide a comprehensive overview of the methodology deployed and the challenges encountered in implementing the k-Fold Method for model evaluation and user-item rating predictions.