**Amazon Product Reviews - NLP Analysis**

**Table of Contents**

**1. Project Overview**

**2. Problem Statement**

**3. Data Source**

**4. Methodology**

**1. Data Loading and Preprocessing**

**2. Exploratory Data Analysis (EDA) and Data Cleaning**

**3. Feature Engineering**

**4. Model Building and Evaluation**

**5. Results and Conclusion**

**Project Overview**

This project performs Natural Language Processing (NLP) analysis on Amazon product reviews to predict sentiment (+ve or -ve) based on the review text and metadata. The dataset used is a subset of the Amazon product reviews dataset, focusing on the "Appliances", "All\_Beauty", and "AMAZON\_FASHION" categories.

**Problem Statement**

The goal is to build a classification model that can accurately predict whether a given product review is positive or negative. This is achieved by:

**1.Preprocessing the text data:** Cleaning and preparing the review text for analysis.

**2.Feature Engineering:** Transforming the text data into numerical features that can be used by a machine learning model.

**3.Model Building:** Training and evaluating a classification model to predict sentiment.

**Data Source**

The data is sourced from the [Amazon product reviews dataset] (https://nijianmo.github.io/amazon/). The following JSON files were used:

`Appliances\_5.json.gz`

`All\_Beauty\_5.json.gz`

`AMAZON\_FASHION\_5.json.gz`

**Methodology**

**1. Data Loading and Preprocessing**

What we did: Loaded the data from the `.json.gz` files into a Pandas DataFrame. We then combined the data from the three categories into a single DataFrame.

Why we did it: To create a unified dataset for our analysis.

Results: A single DataFrame was created with 10,722 reviews and 12 columns.

What we did:Created a `Score` column based on the `overall` rating:

`1` for ratings of 4 or 5 (positive)

`0` for ratings of 1 or 2 (negative)

Why we did it: To convert the multi-class rating into a binary sentiment classification problem.

What we did: Dropped reviews with a neutral score of 3.

Why we did it: To focus on clearly positive and negative reviews and simplify the classification task.

Results: The dataset was reduced to 9,855 reviews.

What we did: Converted the `unixReviewTime` to a datetime object.

Why we did it: To enable time-based analysis and sorting.

**2. Exploratory Data Analysis (EDA) and Data Cleaning**

**What we did:** Removed duplicate reviews based on "reviewerName", "UnixTime", "summary", and "reviewText".

**Why we did it:** To remove redundant information and prevent bias in the model.

**Results:** The number of unique reviews was reduced to 1,874, which is approximately 19% of the original data.

**What we did:** Cleaned the text data by:

Removing HTML tags and URLs.

Expanding contractions (e.g., "don't" to "do not").

Removing punctuation and special characters.

Converting text to lowercase.

Removing stopwords.

**Why we did it:** To standardize the text and remove noise that could negatively impact the performance of the NLP models.

**Results:** The `reviewText` and `summary` columns were processed and stored in new columns named `processedText` and `processedSummary`, respectively. A few rows with null text were dropped.

**3. Feature Engineering**

**Bag of Words (BoW):**

**What we did:** Created a document-term matrix using `CountVectorizer`.

**Why we did it:** To represent the text data as a matrix of token counts.

**Results:** A sparse matrix of shape (1871, 5928) was created, representing 1871 reviews and 5928 unique words.

**TF-IDF:**

**What we did:** Transformed the text data into TF-IDF features using `TfidfVectorizer`.

**Why we did it:** To give more weight to words that are important to a specific review and less weight to words that are common across all reviews.

**Results:** A TF-IDF matrix of shape (1871, 869) was created using unigrams and bigrams.

**Word2Vec:**

**What we did:** Trained a Word2Vec model on the processed review text.

**Why we did it:** To create dense vector representations of words that capture their semantic relationships.

**Results:** A Word2Vec model was trained with a vocabulary of 1,540 words.

**What we did:** Created average Word2Vec vectors for each review.

**Why we did it:** To represent each review as a single vector.

**What we did:** Created TF-IDF weighted Word2Vec vectors for each review.

**Why we did it:** To combine the semantic information from Word2Vec with the importance of words from TF-IDF.

**4. Model Building and Evaluation**

**Logistic Regression:**

**What we did:** Split the data into training and testing sets. Trained a Logistic Regression model on the TF-IDF weighted Word2Vec features.

**Why we did it:** To build a baseline classification model.

**Results:** The model achieved an accuracy of 92% on the test set. However, the confusion matrix and classification report revealed that the model was only predicting the majority class (positive reviews) and failed to identify any negative reviews. This is a classic example of a model that performs well on accuracy but is not useful in practice due to severe class imbalance.

**Support Vector Machine (SVM) with SMOTE:**

**What we did:** Oversampled the training data using SMOTE to address class imbalance. We then trained an SVM model on the resampled data.

**Why we did it:** To create a more balanced dataset and improve the model's ability to identify the minority class (negative reviews).

**Results:** The SVM model achieved a lower accuracy of 53% but was able to identify 66% of the negative reviews. This demonstrates that by addressing the class imbalance, the model was able to learn the characteristics of the minority class, even at the cost of overall accuracy.

**Results and Conclusion**

The project successfully demonstrated the end-to-end process of NLP analysis on Amazon product reviews. The key findings are:

**Class Imbalance is a Major Challenge:** The dataset is highly imbalanced, with a vast majority of positive reviews. This significantly impacts the performance of the classification models.

**Logistic Regression:** While achieving high accuracy (92%), the Logistic Regression model was not a practical solution as it failed to identify any negative reviews. This highlights the importance of looking beyond accuracy as an evaluation metric, especially in imbalanced datasets.

**SVM with SMOTE:** The SVM model, when combined with SMOTE for oversampling, was able to identify a significant portion of the negative reviews (66% recall for the negative class). This came at the cost of a lower overall accuracy (53%).

**Final Conclusion:**

The choice between the two models depends on the specific business objective. If the goal is to identify as many positive reviews as possible, the Logistic Regression model might be preferred. However, if the business needs to identify negative reviews to address customer complaints and improve products, the SVM model with SMOTE is the more valuable tool, despite its lower overall accuracy.

This project serves as a good starting point for sentiment analysis on product reviews. For future improvements, one could explore more advanced NLP techniques like using pre-trained language models (e.g., BERT, RoBERTa), experimenting with different oversampling/undersampling techniques, and performing hyperparameter tuning to optimize the models further.