Sentiment Analysis of Amazon Product Reviews: Using Logistic Regression

2024

MARCH 178

Neharika Joshi (NJ23110012450)

Description of Dataset

```
#   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    5000 non-null   object
 1   dateAdded             5000 non-null   object
 2   dateUpdated           5000 non-null   object
 3   name                  5000 non-null   object
 4   asins                 5000 non-null   object
 5   brand                 5000 non-null   object
 6   categories            5000 non-null   object
 7   primaryCategories     5000 non-null   object
 8   imageURLs             5000 non-null   object
 9   keys                  5000 non-null   object
 10  manufacturer          5000 non-null   object
 11  manufacturerNumber    5000 non-null   object
 12  reviews.date          5000 non-null   object
 13  reviews.dateAdded     1052 non-null   object
 14  reviews.dateSeen      5000 non-null   object
 15  reviews.doRecommend   5000 non-null   bool
 16  reviews.id            29 non-null     float64
 17  reviews.numHelpful    5000 non-null   int64
 18  reviews.rating        5000 non-null   int64
 19  reviews.sourceURLs    5000 non-null   object
 20  reviews.text          5000 non-null   object
 21  reviews.title         4987 non-null   object
 22  reviews.username      5000 non-null   object
 23  sourceURLs            5000 non-null   object
```

This dataset comprises information regarding consumer reviews of various products available on Amazon. Each row represents a single review, and the columns provide details such as the product's name, brand, manufacturer, categories, and ASINs (Amazon Standard Identification Numbers). Additionally, it includes information about the review itself, such as the rating given by the reviewer, the review text, whether the reviewer recommends the product, and the helpfulness of the review based on the number of people who found it helpful.

The dataset also contains temporal information, such as the dates when the reviews were added, last updated, and last seen. These timestamps can be useful for analyzing trends over time, identifying seasonal patterns, or assessing changes in consumer sentiment.

Moreover, the dataset includes identifiers for the reviews and the reviewers, such as unique IDs and usernames. These identifiers enable tracking and analyzing individual reviews and reviewers, facilitating tasks like sentiment analysis, user profiling, and recommendation system development.

Overall, the dataset offers a comprehensive view of consumer feedback on Amazon products, making it valuable for various analytical tasks, including sentiment analysis, trend analysis, and customer behavior analysis.

Preprocessing steps:

The preprocessing steps used in the Notebook aim to clean and normalize text data, making it suitable for further analysis or modeling tasks. Here's an explanation of each step:

1. **Lowercasing**: The clean_text function converts all text to lowercase. This step ensures that words with different cases are treated the same way and helps in standardizing the text.
2. **Removing Whitespaces**: Leading and trailing whitespaces are removed from the text using the strip() method. This step ensures consistency in text formatting and removes unnecessary spaces.
3. **Removing HTML Tags**: The function removes any HTML tags present in the text using regular expressions. This step is crucial when dealing with text data extracted from web pages to ensure that HTML markup does not interfere with the analysis.
4. **Replacing Digits**: The re.sub() method with a regular expression pattern '\d+' replaces digits with spaces. This step is often performed to remove numeric characters from text data, as they may not contribute significantly to the analysis or modeling process.
5. **Replacing Punctuation**: Similar to replacing digits, punctuation marks are replaced with spaces using a regular expression pattern. This step helps in removing punctuation that may not carry meaningful information for the analysis.
6. **Tokenization**: The word_tokenize function from NLTK is used to tokenize the text into individual words or tokens. Tokenization breaks down the text into smaller units, making it easier to process and analyze.
7. **Stopword Removal**: Stopwords, such as 'a', 'an', 'the', 'this', etc., are common words that occur frequently in text but often do not carry significant meaning. The remove_stopwords function removes these stopwords from the text to reduce noise in the data.
8. **Stemming**: Stemming is the process of reducing words to their root or base form. The SnowballStemmer from NLTK is used to perform stemming on the tokens. Stemming helps in reducing the dimensionality of the data and consolidating similar words.
9. **Lemmatization**: Lemmatization is similar to stemming but aims to convert words to their base or dictionary form (lemma). The WordNetLemmatizer from NLTK is used to lemmatize the tokens. Lemmatization often produces more readable and interpretable results compared to stemming.

10. **POS Tagging**: Part-of-speech (POS) tagging is performed using NLTK's pos_tag function to identify the grammatical parts of each token (e.g., noun, verb, adjective). This information is used in lemmatization to determine the correct lemma for each word based on its POS.

Overall, these preprocessing steps help in standardizing, cleaning, and normalizing text data, making it more suitable for various natural language processing (NLP) tasks such as sentiment analysis, text classification, and information retrieval.

Evaluation of results

Classification Report:

The classification report provides a summary of key performance metrics for a binary classification model. Here's an explanation of each metric:

1. **Precision**: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). In the context of the report, precision for class 0 (negative class) is 0.72, and for class 1 (positive class) is 0.86. This means that 72% of the instances predicted as negative are actually negative, and 86% of the instances predicted as positive are actually positive.
2. **Recall (Sensitivity)**: Recall measures the ability of the model to correctly identify positive instances out of all actual positive instances. It is calculated as the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). In the report, recall for class 0 is 0.86, and for class 1 is 0.73. This means that 86% of actual negative instances are correctly classified as negative, and 73% of actual positive instances are correctly classified as positive.
3. **F1-score**: The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is calculated as 2 * (precision * recall) / (precision + recall). In the report, the F1-score for class 0 is 0.78, and for class 1 is 0.79. Higher F1-scores indicate better model performance in terms of both precision and recall.
4. **Support**: Support refers to the number of actual occurrences of each class in the dataset. In the report, the support for class 0 is 56, and for class 1 is 70.
5. **Accuracy**: Accuracy measures the overall correctness of the model across all classes. It is calculated as the ratio of correctly predicted instances to the total number of instances. In the

report, the accuracy is 0.79, indicating that the model correctly predicts the class label for 79% of the instances.

6. **Macro Avg**: Macro average calculates the average of precision, recall, and F1-score across all classes, giving equal weight to each class. In the report, the macro average for precision, recall, and F1-score is 0.79.
7. **Weighted Avg**: Weighted average calculates the average of precision, recall, and F1-score across all classes, weighted by the support of each class. In the report, the weighted average for precision, recall, and F1-score is also 0.79.

Overall, the classification report provides a comprehensive evaluation of the model's performance, highlighting its ability to correctly classify instances into the respective classes.

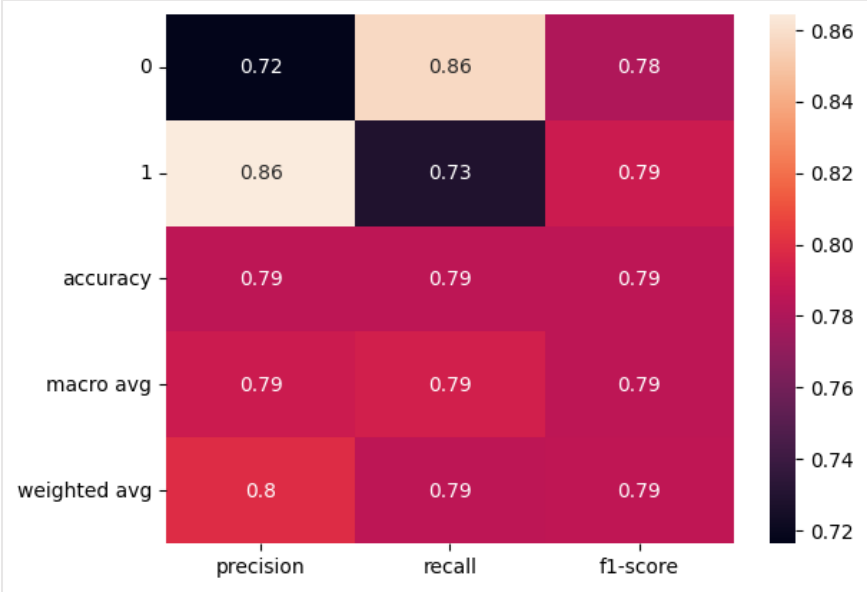

Figure 1: Classification Report of the model

Confusion Matrix:

The confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm and provides insights into the types of errors it is making.

In the provided confusion matrix:

- Δ True Positive (TP): The model correctly predicted 48 instances as positive (class 1).
- Δ False Positive (FP): The model incorrectly predicted 8 instances as positive (class 1) when they were actually negative (class 0).
- Δ True Negative (TN): The model correctly predicted 51 instances as negative (class 0).

- Δ False Negative (FN): The model incorrectly predicted 19 instances as negative (class 0) when they were actually positive (class 1).

This confusion matrix indicates that the model has:

1. Successfully identified 48 instances as negative and 51 instances as positive.
2. Misclassified 8 instances as negative when they were actually positive.
3. Misclassified 19 instances as positive when they were actually negative.

Overall, the confusion matrix provides a clear picture of the model's performance in terms of correctly and incorrectly classifying instances into their respective classes.

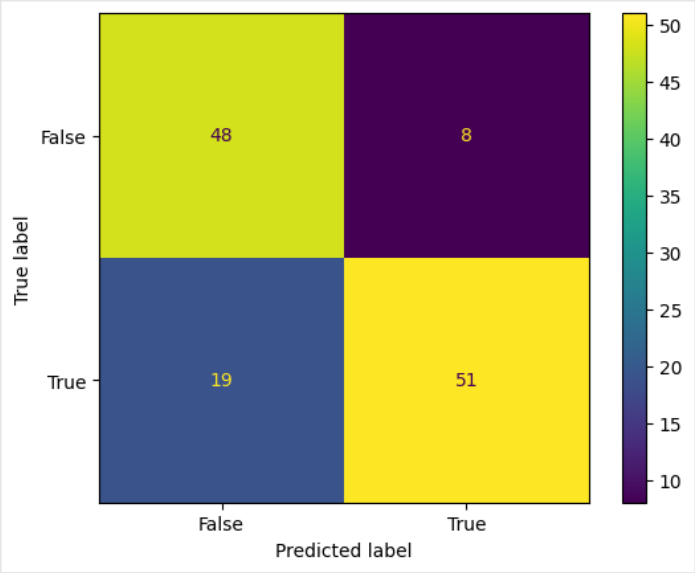

Figure 2: Confusion Matrix

Insights

Strengths:

- Δ **Decent Performance with Limited Data**: Despite the small sample size, the model achieves respectable performance metrics such as precision, recall, and F1-score. This indicates that the model can still learn meaningful patterns from the available data and make reasonably accurate predictions.
- Δ **Efficient Training and Inference**: With a smaller dataset, the training time for the model is likely to be shorter, making it more efficient to train and deploy in production environments. Additionally, inference time would also be faster, allowing for quick predictions on new data.

- Δ **Simplicity and Interpretability:** With fewer samples, simpler models may be preferred, which tend to be more interpretable. This can be advantageous for understanding the underlying factors driving the predictions and gaining insights into the problem domain.

Limitations:

- Δ **Limited Generalization**: The model's ability to generalize to unseen data may be compromised due to the small dataset size. It may not capture the full variability and complexity of the underlying data distribution, leading to suboptimal performance on new, unseen instances.
- Δ **Vulnerability to Overfitting**: With a small dataset, there is an increased risk of overfitting, where the model learns to memorize the training data rather than generalize from it. This can result in poor performance on unseen data and reduced reliability in real-world scenarios.
- Δ **Difficulty in Capturing Rare Events**: If the dataset is small and imbalanced, as mentioned, capturing rare events or minority classes becomes challenging. The model may not have enough examples to learn meaningful patterns for these classes, leading to biased predictions and reduced accuracy.
- Δ **Limited Exploration of Feature Space**: With a small dataset, there may be fewer opportunities to explore the full feature space adequately. This can result in overlooking potentially relevant features or interactions among features, limiting the model's capacity to capture the underlying data distribution accurately.

Given these considerations, it's essential to be cautious when interpreting the model's performance and to consider strategies such as cross-validation, regularization, and feature selection to mitigate the limitations associated with the small dataset size. Additionally, collecting more data or augmenting the existing dataset through techniques like data synthesis or transfer learning could help address some of these limitations and improve model performance.