Sentiment Analysis Report

Dataset

The dataset contains 5,000 rows of review data for Amazon products such as Kindle, Fire, and Echo products - and their variants. Many of the columns in this dataset contain review metadata and can be dropped to make viewing the important data easier and faster.

Data Preprocessing

After exploring the data in our feature column of ‘reviews.text’ we can see there are no null values but that there are 615 repeated rows. We drop these now as leaving them in would result in a skewed evaluation later on.

Next, we reset the index to prevent any KeyErrors when we come to test our models performance and general quality of life.

The main bulk of preprocessing is covered as we convert our column of ‘object’ type, into ‘string’. This allows us to; strip trailing & leading whitespace, convert to lower case, and remove punctuation using the ‘translate’ method. These steps make our NLP more efficient by reducing the number of tokens in each review, the trade-off being minimal as punctuation adds little meaning.

Finally, we define our ‘preprocess’ function; this will call spaCy’s nlp function on the reviews. The function will return a string of the lemmatized tokens from the nlp doc excluding stop words for each review. Later we will experiment to see the effect stop words have on our model’s performance.

Testing revealed a slight issue in our way of dealing with punctuation causing two words to merge together due to non-standard use of punctuation, particularly ellipses. The method used to remove punctuation from the reviews caused the words to merge if there was not a whitespace following the punctuation.

```
.str.translate(str.maketrans(' ',' ', string.punctuation))
```

This is easily corrected by removing the above line and editing the return for our preprocess function to:

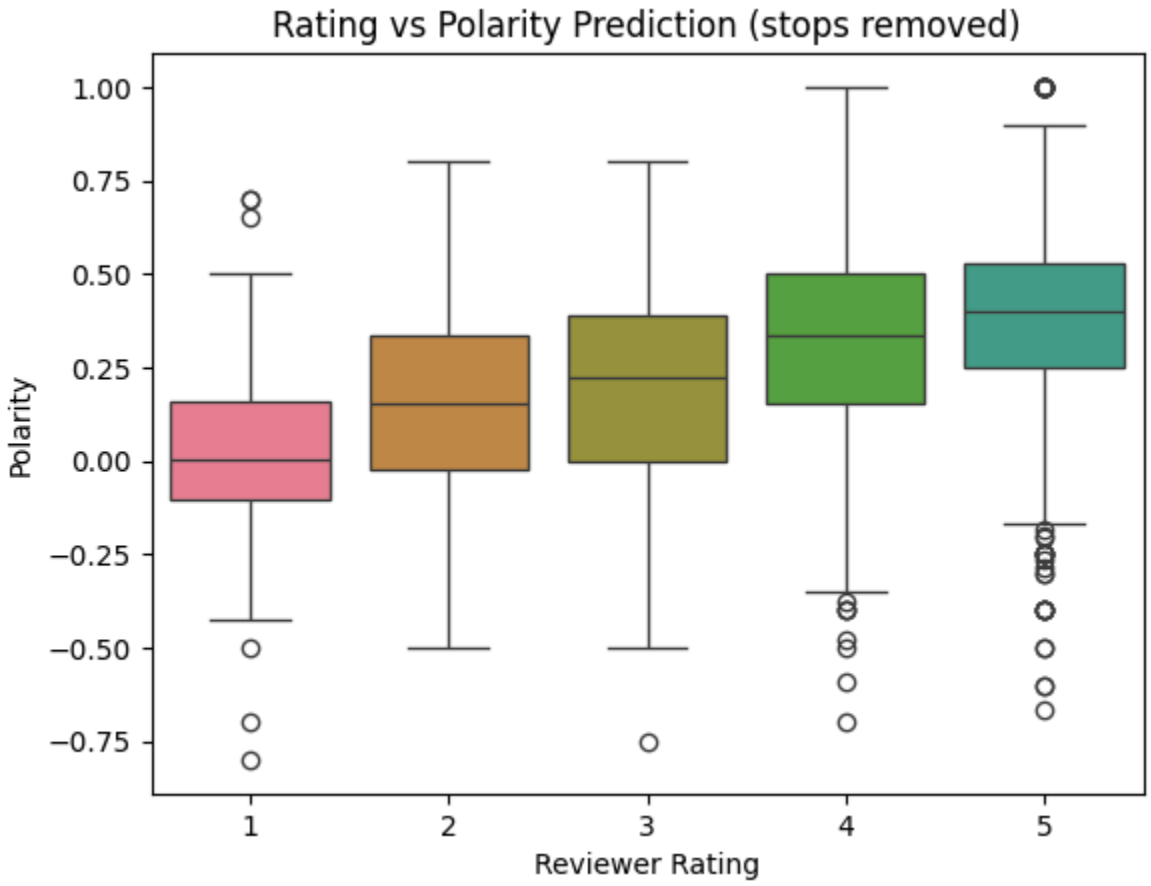
```
' '.join([token.lemma_ for token in doc if not token.is_stop and not
token.is_punct])
```

This does result in a longer process time for the data but properly separates tokens.

Evaluation

To understand the performance of the model we can look individually at raw reviews and their polarity score after we have processed them; we see in most cases that the meaning has been preserved and that our program has divined an accurate sentiment. However, by excluding stop words from our doc's tokens we have removed common negations like 'no', 'not', and 'never'. In some short reviews, by removing these words, the meaning is flipped entirely, or in longer reviews, consecutive phrases appear to have contradictory sentiment. Also among the stop words are the common adverbs 'very' and 'really', which are used to emphasise the adjective in normal speech, without which the reviews can seem dulled, turning a "very positive statement" into just a "positive statement".

In evaluating the performance of the model I decided to compare polarity to the 'reviews.rating' columns, assuming that the sentiment of the review is accurately reflected in the text, this would be a good comparison. Below is a boxplot of this 'reviews.rating' column against our model's polarity assessment.



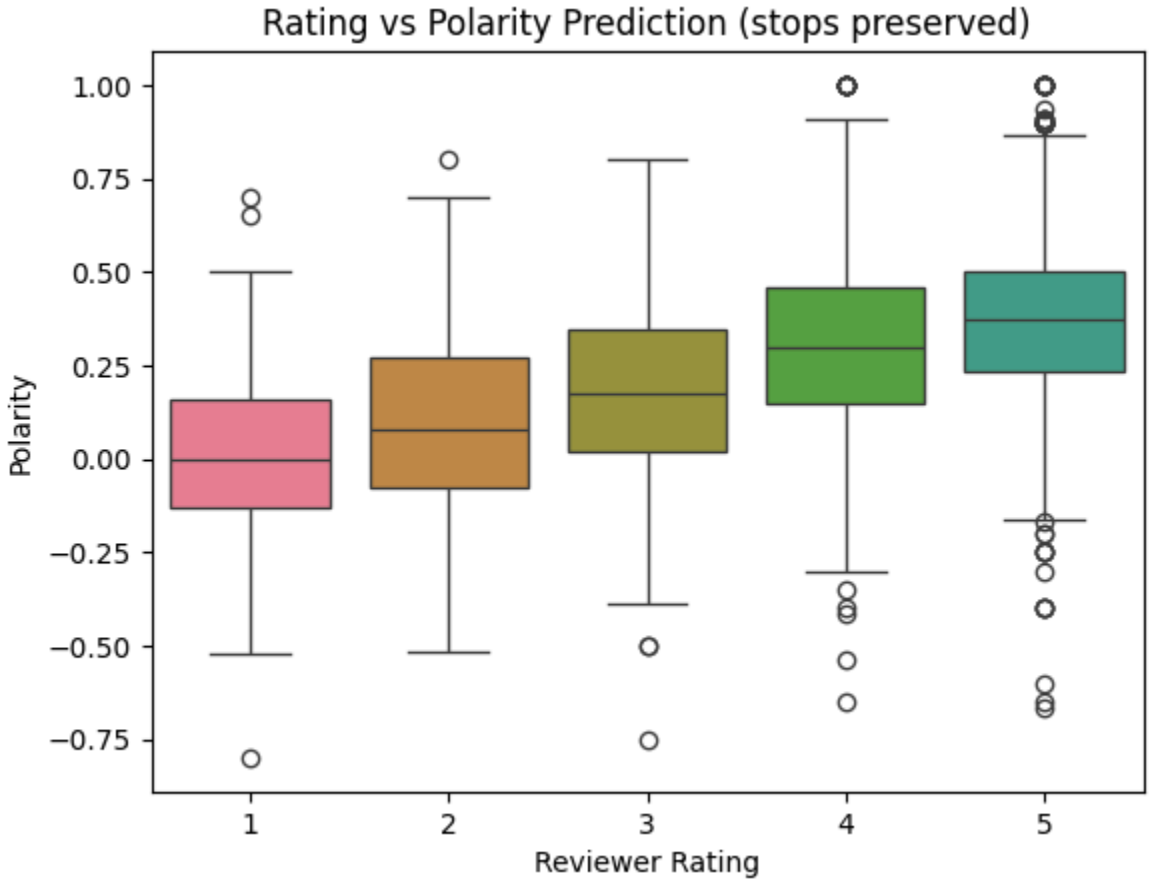
We can see that all review rating categories have a mean above 0 polarity. There is clearly a positive correlation between reviewer rating and polarity but not as strong as I had

hoped. The above chart also reveals a large range of predicted polarity at all ratings and many negative outliers on our 5* review predictions.

Mean polarity grouped by review rating (stop words removed).

1	0.015169
2	0.135094
3	0.204473
4	0.333829
5	0.390349

Below we apply the model again but retain stop words.



The performance is overall worse, with all means falling slightly and the MAE increasing. This demonstrates the noise that stop words add to processing language.

Mean polarity grouped by review rating (stops preserved).

1	0.005935
2	0.098409
3	0.187736

4 0.313305
5 0.373442

Another way of looking at the performance of the model is through the Mean Absolute Error (MAE) of the whole set of polarity values. To do this we will have to assign each of the 1-5 ratings a polarity value on our -1, 1 interval, we map the transformation by thinking of each rating as the midpoint of an even-sized interval of our polarity range:

```
reviews_data['rating_map'] = reviews_data['reviews.rating'].map({1: -0.8,
2: 0.4,                              3: 0, 4: 0.4, 5: 0.8})
```

Then we calculate the absolute error for each row:

```
reviews_data['abs_polarity_error'] = abs(reviews_data['rating_map']
                                         - reviews_data['polarity'])
```

This allows us to easily calculate the MAE as: 0.364

Insights

Let's briefly look at some examples to understand the limitations of the model. I have sorted the data by abs error to find the reviews we have performed worst on. (References are for re-indexed dataset with duplicates removed.)

Row	Original review	Processed review	Review Rating	Polarity
4364	Very cheap and was not impressed at all never again	cheap impressed	1	0.7
1479	Would not take a charge, good thing i tried before wrapping it for Christmas.	charge good thing try wrap christmas	1	0.7
2188	I got this because it is almost impossible to break	get impossible break	5	-0.67
2026	did not power up...at all. I am so happy that my grand-niece wasn't here and doesn't know about this yet, she would have been crying.	power happy grand niece know cry	1	0.65
2166	Goog tablet the only bad part is that you have to be creating a profile and thats where it gets complicated because when they sell u the tablet they saw its for kids	goog tablet bad create profile s get complicated sell u tablet see kid	5	-0.6

4195	Gave it to my granddaughter and she loves it..I want to buy one for myself....	give granddaughter love iti want buy	1	0.5
2803	Was very difficult to set up. With the 3 hours help of Geek Squad and my granddaughters I-Phone we got it done.	difficult set 3 hour help geek squad granddaughter iphone get	5	-0.5

[sic]

We can see here the limitations removing stop words brings to some examples. Particularly those with very simple vocabulary like row 4364. We also glimpse an issue with assessing the polarity scores en masse on row 4195: A review that is positive with no qualifiers or mitigations has been entered as 1*. There are a few such reviews in the dataset where even a human processing of the text cannot reconcile the review and the rating. The same could be said for row 2803; we might infer as humans that after the difficult set-up they were very happy with the product, but a literal reading of the text would agree with the polarity assessment of the model as negative.

Another big drawback of the data is the relative paucity of negative review data, with only 95 negative reviews (1 or 2 *) vs 4115 positive we have a limited understanding of how well we are processing text with negative sentiment.

On the side of strengths we see our model accurately assessing sentiment on the majority of reviews, ~25% of reviews have an abs error of less than 0.2. The model performs best on reviews with clear and non-colloquial english, efficiently boiling down multi-sentence text to a few key words. Our overall MAE of 0.364 is a good result and with some refinement, and more negative review data to balance the dataset, can be even better.