Daniel Ooi

Introduction

Upon examining the prompt and dataset, I considered how I might approach predicting an Amazon review rating using only the provided data fields, similar to how I might approach it in real life. Naturally, the Text and Summary fields seemed to be the most informative for understanding a review's sentiment, as these fields contain the actual feedback from users. I anticipated that the model would need more than just keyword recognition to achieve meaningful predictions. Therefore, my focus was on developing a strategy to leverage these fields effectively through feature engineering, aiming to capture nuances beyond simple keyword matching.

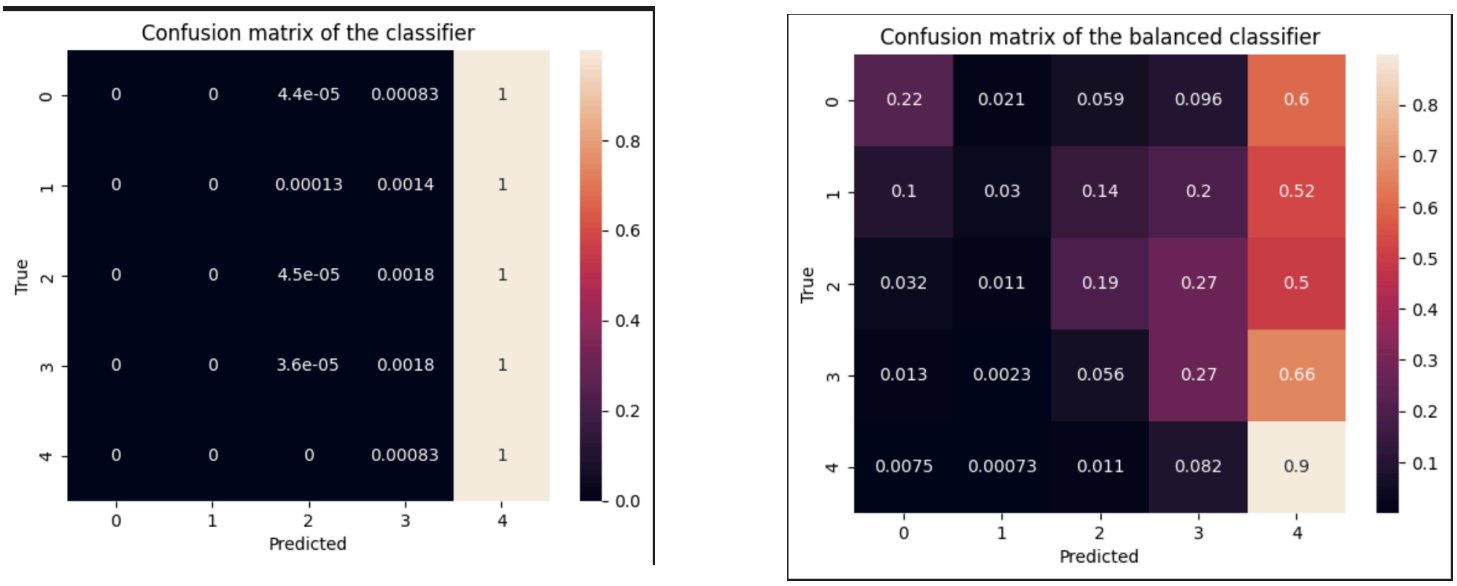
Features

For feature engineering, I wanted to use as many data fields as possible in addition to dissecting the text and summary fields. For the first part of adding features, I calculated a helpfulness ratio by dividing the helpfulness numerator with the helpfulness denominator. This feature would measure how helpful a review was to other users. For missing values, I filled them with values of zero, because I assumed that those with no helpfulness reviews had low or no engagement.
I then encoded product and user ids with numerical values as I realized it would be beneficial to train the data to see the correlation between the same users filling out different reviews as well as different products having similar review ratings or not. Finally, I needed to learn how to analyze words in the texts and summaries. I knew a little about semantics from class, but I wasn't confident that I would do the best, so I searched up other techniques. Eventually, I found the TF-IDF method. I used this method in order to convert the text and summary columns into numerical features. By focusing on the most important terms across all reviews, TF-IDF captures keywords that may indicate sentiment, product quality, or other impactful aspects of reviews.

Since I put the most weight on text and summary, I decided to implement other things we've learned in class to better group together findings. The TF-IDF vectors for Text and Summary were transformed using Truncated Singular Value Decomposition (SVD) to reduce the number of features. This step was essential for avoiding overfitting and reducing model complexity. The final features from this reduction (50 components each for text and summary) encapsulate the core themes without the high dimensionality of the original TF-IDF vectors. Finally, to capture patterns in the review sentiment, I utilized KMeans in order to try to group similar features. These clusters served as an additional feature, with the intuition that reviews sharing similar sentiment or thematic content may correspond to similar ratings.

Patterns I Noticed & Special Tricks

I realized very quickly that by only focusing on the features portion, I got a 53% accuracy score. I realized my confusion matrix showed big fat 0's, so I opened up my first draft submission.csv and found that almost all the scores were 5's with very little 4's and almost none of the other 3 ratings. I realized that since I was using KNN, it was often focusing on distance, and since the majority of the dataset, ~800,000, were 5's, it was predicting 5's for most submissions. So, the thing that helped me the most was to balance out the dataset. I did this, by first separating the majority class, which was 5, with the other four ratings. Then, I downsampled the majority class, reducing the number of 5-star reviews to match the minority classes, ensuring a balanced dataset. This trick really helped improve my accuracy.



Before vs After Downsampling the Majority Class

For parameter tuning, I thought back on the KNN assignment from week 5, and realized my best k was 50, which was 0.005 of the total data of 10,000. I attempted to do the same for this dataset, setting k as 10,000. However, I realized that this might lead to the same problem as above, averaging the dataset to see mostly 5's and thus underfitting the model and subsequent accuracy. Thus, I tried smaller k_values, and my hypothesis was proven to be true as I saw the accuracy start to plateau at really low n_neighbors, specifically 100 and 200. Thus, for my final submission, I set n_neighbors at 200.

One thing that I tried during my testing was omitting the helpfulness ratios. I found that by including the helpfulness ratios, my predictions in the confusion matrix for 1's and 5's actually increased. This thus also increased my accuracy in the long run. This pattern makes sense, as highly engaging or divisive reviews tend to evoke strong opinions, either very positive or negative which helps with distinguishing the polar reviews.

A special trick that proved effective was performing SVD on the TF-IDF vectors after splitting the data into training and testing sets. This approach allowed for consistent feature reduction across both sets and prevented the need to re-run SVD on the entire dataset each time. This saved computational resources and ensured consistency in the reduced feature space.

As KNN and feature engineering were computationally intensive, one practical improvement was limiting the sample size to 100,000, which significantly reduced training time while maintaining accuracy. This adjustment balanced model performance with computational efficiency, allowing for faster experimentation and iteration.

Things I Tried & Next Steps

Something that I tried using was applying sentiment analysis on the Text and Summary fields in addition to TF-IDF which I thought would help as it groups based on polarity. One possible reason is that sentiment alone may not capture the nuances of star ratings, especially in cases where reviewers use mixed sentiment or where polarity does not align directly with the star rating.

Additionally, I chose not to look at the Time field as I did not expect that the time when the review was posted would have anything to do with the review score. Using this field might have proved to be a little helpful as during our class where we discussed the midterm afterwards, one of the top performing students noted his surprise when he graphed it out and saw that there was a very present increase in scores as much more time passed.

As we discussed the midterm in class on October 28, something that I want to try would be to combine KNN with other algorithms like Decision Trees or Naive Bayes to create an ensemble model. This hybrid approach might leverage strengths from different algorithms, potentially improving predictive power.

Conclusion

While KNN is interpretable, it is computationally intensive with large datasets. An alternative classifier, such as a decision tree or ensemble method, could provide better scalability. Additionally, the model sometimes struggled with predicting intermediate ratings. Further improvements in feature engineering, perhaps by including additional metadata or more articulated sentiment analysis, could help.

Citations

"Understanding TF-IDF (Term Frequency-Inverse Document Frequency)." *GeeksforGeeks*, GeeksforGeeks, 19 Jan. 2023 www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/