Names: Aya Hajjeh and Bilan Aden
Kaggle usernames: ayahajjeh and BilanAdenn
Kaggle team name: Aya & Bilan
Binary Classification of Amazon Product Ratings Project

Data

- Describe the dataset you worked with, including explanations of interesting feature variables and the target variable.
- Description of the dataset
    - The dataset contains various information about amazon reviews across a variety of products and product categories. It includes various features such as overall, verified, review time, reviewer id, asin, reviewer name, review text, summary, unix review time, vote, image, style, category, and id of the sample in the data.
- Some of the interesting features were
    - Verified
        - This denotes if a review has been verified by Amazon as being legitimate. Verified reviews should have a higher weight than non-verified reviews when processing the data.
    - Review text
        - The textual content of the body of the review. This astronomically influences the overall rating of the product. Positive reviews indicate a higher-rated product, and vice versa.
    - Summary
        - A high-level summary of the review. This also astronomically influences the overall rating of the product. Positive review summaries indicate a higher-rated product, and vice versa.
    - Category
        - High-level product category (e.g. Toys, Automotive). It can be relevant to distinguish between different categories of products since reviewers might leave reviews of makeup products in a different tone than they might describe automotive products.
- The target variable
    - The target variable is the overall rating of the products in the data set. We are performing a binary classification, where the goal is to identify if the text represents a high-star review or a low-star review. For the purposes of this project, we are defining a high-star review as one with a score > 3 (i.e. a 4 or 5 star review), with low-star reviews being <= 3 (i.e. 1,2,3 star reviews).

Methods

- Detail the steps you took to process your data
    - First I dropped all the columns that I won't be working with. In the training data set, that would be all the columns except for the overall, text review, and summary columns. For the testing data set, that would be all the columns except for the review text and the summary columns. After dropping the columns I won't be working with, I combined both

of the review text and summary columns into one column by joining the strings in each row into one single column instead of two. In the training data set, I converted the overall column into 1s and 0s instead of values varying from 0 to 5. Essentially, any rating less than or equal to 3 would be 0, and any rating more than or equal to 4 would be 1.
    - After having the two text columns merged into one single text column, I played with some models where I deleted the stop words in each string, while in other models, I kept the stop words. Eventually, I decided to build my SVC model after removing the stop words, whereas I kept the stop words in my other three models: Logistic Regression, Perceptron, and Random Forest Classifier.
    - After that, I used the TfidfVectorizer function to vectorize the text data so that it's ready to be used by the different classification models such as Logistic Regression, Perceptron, SVC, and Random Forest Classifier.
- Describe the steps you took to build your model
    - After having the text data and the target label ready to be used by the different classifiers, first I chose what classifier I wanted to experiment with. I experimented with four different classifiers: Logistic Regression, Perceptron, SVC, and Random Forest Classifier. Then, for each classifier, I played with some hyperparameters to see how the same classifier can produce different results using different hyperparameters such as max iterations, regularizer, and C. After playing with such hyperparameters, I decided on initial values that produced the best F1 macro score for each classification model. Then, I passed the model along with the initial hyperparameters that I chose to do a grid search using 5-fold cross validation models to pick the best hypermaters for each model.
- Describe the hyperparameters you used in your best model. Describe how you did your hyperparameter search
    - The best hypermaters I used in my best model were C = 7.0 and max_iter = 1000
    - I found these hyperparameters using the Grid Search CV function using
        - a 5-fold cross validation process
        - and F1 macro score as the evaluation metric
        - And by using all the available processors, ie, n jobs = -1

Results

- Show the confusion matrix, ROC, AUC, macro F1 score, and accuracy for the best set of hyperparameters using 5-fold cross-validation for each of your three models.
    - Make sure to show the best model for both Part 1 and Part 2. Thus, you will report each set of metrics twice.

Model	ROC_AUC	Macro F1	Accuracy	Confusion matrix
Model 1 (Best Logistic Regression model)	0.8408391635364558	0.8058076225045373	0.8533744433025009	[[3206 370] [486 1776]]

Model 2 (Best SVC model)	0.8304356630409628	0.793199813693526	0.8478931140801644	[[3247 329] [559 1703]]
Model 3 (Best Perceptron model)	0.7563557373352559	0.6988779482482254	0.7747516272696129	[[2997 579] [736 1526]]

Results from Part 1

The best model for Part 1 would be Model 1 as described in the Methods section above

Model	ROC_AUC	Macro F1	Accuracy	Confusion matrix
Model 1	0.6696355455467929	0.6525079078174424	0.6048304213771884	[[1365 2211] [96 2166]]

Results from Part 2 (Sentiment Only)

Model	ROC_AUC	Macro F1	Accuracy	Confusion matrix
Model 1	0.8512069276972406	0.8121621621621623	0.8571428571428571	[[3201 459] [375 1803]]

Results from Part 2 (Text + Sentiment)

Please note that the F1 score we got for this model on the validation data set was below the Kaggle baseline, but when we submitted the actual test predictions for this model on Kaggle, the F1 score we got was above the baseline (see the F1 scores we got on Kaggle below)

- Discuss why these metrics are appropriate for this task. (note, for this question I consulted ChatGPT 3.5 for some help)
    - Accuray is an appropriate metric for this task because we care about the percentage of total items classified correctly, ie, misclassification of both classes has equal importance to us for this task
    - ROC_AUC is an appropriate metric for this task because it is less sensitive to class imbalance and provides a more balanced assessment of model performance. It measures the model's ability to discriminate between high-rated products and low-rated products regardless of the class distribution.
    - Macro-F1 is an appropriate metric for this task because this metric provides an equal weight to both classes (high-rated products and low-rated products). It calculates the F1 score for each class separately and then takes the average, which means it treats both classes as equally important. This can be valuable to us if we want to give equal consideration to correctly identifying high-star and low-star product reviews.
    - Confusion matrix is an appropriate metric for this task because this matrix provides a comprehensive and detailed breakdown of the model's performance, making it valuable for various reasons such as class-specific performance and misclassification analysis.

- List the score you achieved using Kaggle
    - Text-Only F1 macro score: 0.84811
    - Text + Sentiment F1 macro score: 0.85053

Error Analysis - Sentiment Analysis Model

- Recall from the project instructions “include analysis as to why sentiment is or is not enough to perform amazon review predictions”.
    - Describe why using the sentiment analysis model to predict binary review scores (i.e. using the Vader model by itself with no other information) is a good or bad choice. Support your explanation with at least two examples. That is, look at the correct predictions / errors made by this model and select a few texts to help you in writing your description.
    - Sentiment analysis is not enough (alone) to perform amazon review predictions because we have noticed this by looking at the Review_text section. Sentiment analysis does provide general insights but it can’t be applicable to every detail in a comment. For example, a binary classification would say a rating of a ⅔ is in the middle as neutral but that alone can have different sentiments that aren’t necessarily neutral. Therefore that prediction alone can be misleading. It is also important to mention that the category column provided some context for the review text because each review comes from a product that is in a category and that can have a range of sentiment meanings.
    - In this section we combined review text and summary and this was all in the train data and that was combined into train review. Then we used the sentiment analysis function to encode the train reviews into binary labels that used vader sentiment analysis.
    - Then we built a logistic regression model and it was used for binary classification based on the printed sentiment results. The specific target here was the overall column which is used to create a label for the classifier. If the prediction is greater than or equal to the threshold then it is assigned a 1 and 0 if it is less than 4 which is the threshold respectively.
    - The sentiment alone was not enough as shown on the table above but when combined with text the scores were much higher. And we chose the function we did with the parameters because it performed best at part 1.
    - 'my grandson loved this. it is a great toy'
    - {'compound': 0.8402, 'neg': 0.0, 'neu': 0.467, 'pos': 0.533}
    - **********
    - 'wrong part. our fault'
    - {'compound': -0.7003, 'neg': 0.744, 'neu': 0.256, 'pos': 0.0}
    - **********
    - 'this wire set it really sucks!!!'
    - {'compound': -0.5674, 'neg': 0.423, 'neu': 0.577, 'pos': 0.0}
    - **********

      The examples here show wrong part our fault is more of a negative text than this wire set really sucks and that only speaks for the error that is bound to happen.

Future Work

- Describe potential future work: what would you do differently next time?
    - For this project, we focused primarily on text features of the data set. As helpful and relevant text features proved to be, we might in the future try to include other features that can improve our classification that are not necessarily textual features such as the verified, category, and review time features especially because customer opinion changes over time and things like product quality can only be determined over time especially when it's a product people want to keep for a long time. It would also be helpful to add a range of prices for products when the review was given and that could lead to a better classification.