*Nathan Dodson, Jui Nagarkar, Rochan Peechara, Rachel Weis*

**Introduction**

Customer reviews hold a vast amount of information that organizations can leverage to better understand their product and customer base. But, this wealth of information is only as useful as organizations make it. One such useful understanding about reviews is whether or not they are considered “helpful” to other customers, such as if a review leads to a purchase or if another purchaser found the information in the review to be helpful for the purchase. Many e-commerce websites, including Amazon, use a system of ‘upvotes’ and ‘downvotes’ from customers to determine how useful a review is. With the power of text analysis using Python libraries and a variety of model building, our project aims to predict the usefulness of reviews. It generally takes multiple votes to determine whether or not a review is useful, and our models will attempt to predict the review’s usefulness based on the corpus text, all without the need of customer votes.

The selected dataset for our text analysis is from a publicly-available Amazon product dataset, specifically for the product category of “Grocery and Gourmet Food.” The raw data contains a 5-core set with 151,254 observations that we used as our starting point, instead of the full raw dataset.

**Data Preprocessing/Exploratory Analysis**

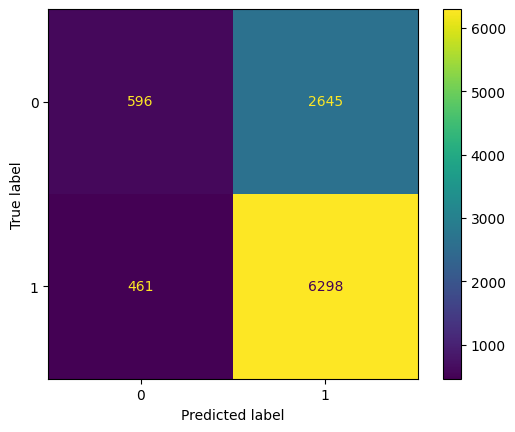
We began by creating a subset from the original dataset. This subset contains 50,000 rows and is a more suitable size for making analysis compared to the original (this subset was ~⅓ the original observation set). From here, we further subsetted the dataset to contain only the necessary columns for our analysis: review text, total votes, number of helpful votes, and proportion of helpful votes calculated as the ratio between the helpful votes and the total number of votes. This subset was much more digestible than the original 151,254 rows that the subsetted database started with. From here, we categorized the reviews based on the proportion of upvotes. The chart shown in Figure 1 shows the distribution of review helpfulness for our dataset. Using this distribution, we were able to select a cutoff of 0.5 for our review usefulness index. Each review datapoint was categorized as useful/not useful based on this proportion. 

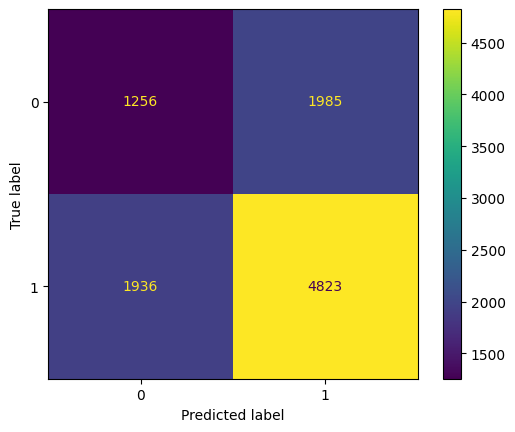
From here, we performed sentiment analysis on the entire corpus and subsetted the data based on the category useful and not useful. The mean compound polarity for the entire corpus is 0.69, the mean compound polarity for the useful subset is 0.72 and the average compound polarity for not useful is 0.60. One interesting implication for these results is that ‘useful’ reviews had a significantly higher overall sentiment than those that were considered ‘not useful.’ We also gathered emotion scores on the reviews in the subsetted data. Emotions like ‘surprise,’ ‘joy,’ ‘positivity,’ and ‘anticipation’ were common among the ‘useful’ reviews. Emotions like ‘anger,’ ‘disgust,’ ‘fear,’ and ‘negative’ were common among the ‘not useful’ reviews.

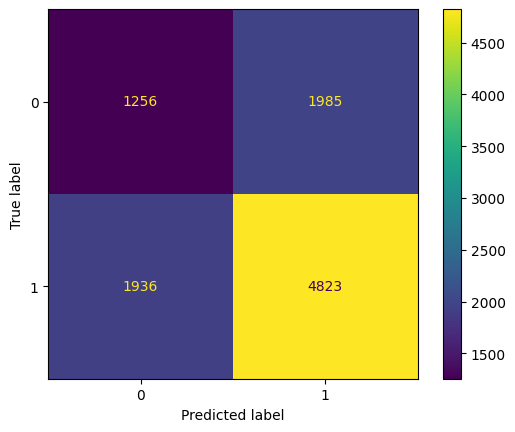
The next step in our model building process involved initial preprocessing of the data. Using python string manipulation, we looped through the corpus to make words lowercase, remove numbers, remove punctuation, remove white space, and apply stemming to the words in the corpus. In the Scikit Learn python libraries, we utilized the standard English stopwords constant list that is provided. In addition to the standard provided stopwords, we also added additional stop words that we thought helpful for our analysis. These included ‘like,’ ‘use,’ ‘nice,’ ‘don't,’ ‘i’ve,’ ‘I’m,’ ‘better,’ ‘think,’ and ‘lot,’ as they did not seem to be useful for this type of modeling and our analysis goals. The preprocessing was put into a function and called on the entire dataset as well as the subsets. According to the subsetted DTMs, there are 30,683 useful reviews, and 19,317 not useful reviews in the set. In figures 2.1 and 2.2, you can see word cloud visualizations of the corpus - world clouds are visualizations that show words in font sizes proportional to their frequency in the dataset.

**Experiment Methodology**

We chose three model types for comparison to predict the usefulness of a review. These included a logistic regression model, k-nearest neighbors model, and decision tree classifier. These three model types were chosen because they are the easiest classifier models to understand, especially when in a business context by a lay audience. We created a testing and training split of 20% to 80% respectively of our processed data so we could then train and test our models.

**Logistic regression**

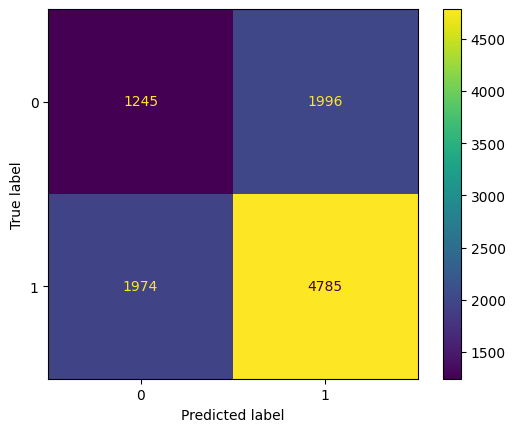
Logistic regression is a regression technique used for predicting categorical classification data. For this reason, we used logistic regression as one of our three model types. The response variable, Usefulness, was converted to binary, with 0 being category Not Useful and 1 being Useful. This resulted in a model with 70% accuracy, .56 AUC, .93 sensitivity, and .18 specificity. From the confusion matrix on the right, we can conclude that the model does better at predicting useful reviews in comparison to its predictive power for not useful reviews.

**KNN Classifier** 

KNN is one of the simplest classification algorithms due to its easy implementation and intuitive understanding. Our implementation uses a k value of 3, which we found to be optimal in our tests for achieving the best diagnostic balance.

As shown in the confusion matrix here, the model predicted a significant number of true positives and true negatives but struggled with false positives and negatives. KNN resulted in a 61% accuracy, 0.55 AUC index, 0.71 sensitivity, and 0.39 specificity.

**Decision Tree Classifier**

Decision Trees are highly favored for their transparency and ease of interpretation. The model’s structure, resembling a tree with branches and leaves, allows us to trace decisions back to their root causes, making it particularly useful for understanding the logic behind each classification.

In our testing, the Decision Tree yielded an accuracy of 60%, an AUC of .54, sensitivity at .70, and specificity at .38. These figures place it closely with the KNN in terms of performance, but with slightly less effectiveness in distinguishing between the classes as indicated by its lower specificity and AUC.

Looking at the confusion matrix displayed here, you can see the distribution of predictions. The model has identified 1,245 true positives and 4,785 true negatives. However, it also misclassified 1,974 instances as false negatives and 1,996 as false positives. This pattern demonstrates the model’s comparable capability in identifying true classes but also highlights its challenge with a considerable number of misclassifications.

**Results + Discussion**

| **Model** | **Accuracy** | **AUC** | **Sensitivity** | **Specificity** |
| --- | --- | --- | --- | --- |
| *Logistic Regression* | 70% | .56 | .93 | .18 |
| *KNN* | 61% | .55 | .71 | .39 |
| *Decision Trees* | 60% | .54 | .70 | .38 |

As we begin our discussion on the performance of these three models, let’s focus on the key metrics presented in this table: Accuracy, AUC, Sensitivity, and Specificity. These metrics help us evaluate how effectively each model predicts outcomes and distinguishes between the classes.

Firstly, looking at Logistic Regression, it stands out with the highest accuracy at 70%. This indicates that it correctly predicts outcomes 70% of the time, making it the most accurate among the models we tested. However, its Area Under the Curve, or AUC, is relatively low at .56, which suggests that its ability to differentiate between the classes is moderate. Its sensitivity is very high at .93, indicating it's excellent at identifying true positives, but it has a low specificity of .18, which means it struggles to correctly identify true negatives, leading to a higher rate of false positives.

Next, the KNN model, which shows an accuracy of 61%. This is slightly lower than Logistic Regression, but still respectable. The AUC here is .55, very close to Logistic Regression, implying similar difficulties in class differentiation. Sensitivity at .71 shows that KNN is reasonably good at identifying true positives, but not as effective as Logistic Regression. Its specificity is higher at .39, which means it’s better at avoiding false positives compared to Logistic Regression but still not ideal.

Lastly, we have the Decision Trees model. This model shows the lowest accuracy at 60% and the lowest AUC at .54, indicating it has the most considerable difficulty distinguishing between the classes effectively. Its sensitivity is close to KNN at .70, and its specificity is similar at .38, making its overall performance comparable to KNN in terms of correctly identifying true negatives and positives but slightly less effective overall.

**Conclusion**

Overall, our analysis aimed to predict the usefulness of customer reviews on e-commerce platforms like Amazon without relying on customer votes, instead utilizing text analysis techniques and the development of three distinct classification models: logistic regression, k-nearest neighbors (KNN), and a decision tree. Logistic regression was demonstrably the most accurate at 70%, excelling particularly in the prediction of “useful reviews” due to its high sensitivity index of 0.91. KNN, with an accuracy of 61%, showed a balanced performance in sensitivity and specificity, while decision trees, though similar in performance, exhibited the lowest accuracy at 60% and struggled with class distinction. Our results overall effectively underscore the trade-offs of these different specific models, while also highlighting the benefits of incorporating a variety of appropriate potential models when exploring a hypothesis. Modeling e-commerce reviews in general has the potential to be very applicable to consumers, especially with the continued maturity of the e-commerce industry (specifically, firms like Amazon), as people seek to find quality products online.