goodreads

SENTIMENT ANALYSIS OF GOODREADS REVIEWS

Measuring the predictive power of word sentiment in estimating reviewer opinions and ratings

ABSTRACT

Sentiment analysis has shown to be a powerful tool when attempting to gauge responses to products and events based on word choice. By using the results of multiple sentiment lexicons, I attempt to make predictions using machine learning classification models on what the user's overall opinion of a book (as determined by out-of-five ratings, based solely on word choice.

Nick Kessler

Overview

User product reviews have a greater importance than ever in guiding consumer decisions. Social review sites allow everyday users to share their opinions with the world and in doing so potentially influence the purchasing decisions of countless people. The number of "stars" these everyday users bestow can have far-reaching effects in shaping consumer sentiment

Goodreads is the world's largest social book review site, home to millions of user-created books reviews & ratings. The aim of this project is to analyze the text of these user-submitted reviews to see with what degree of accuracy machine learning classification models can predict the final score a reviewer assigns a book based on the results of sentiment analysis. In my analysis, I use a collection of four sentiment lexicons to provide as much data as possible for model training.

The process begins with data collection and harvesting the reviews from Goodreads that will be used for performing this analysis. After obtaining the required data and performing the necessary cleansing, I then aim to use various classification models to determine how well sentiment analysis on each review to see if we can accurately predict the final score assigned by the user.

Data Preparation

Data Collection

Performing the sentiment analysis first required obtaining an appropriately large sample of user reviews to analyze. Full Goodreads reviews are not available through the company's web API. To collect the reviews required to perform the analysis, web scraping was necessary.

Before scraping the actual review data, it was first necessary to obtain information about the books in the Goodreads database and identify which books would be good candidates for including in the analysis. The Goodreads database includes information on tens of millions of books, however many of the books in the catalog would not have been good candidates for inclusion in this analysis. Books with few or no reviews, low ratings count, and books that were not published in English and which have no English language reviews were to be excluded from this analysis. A book would be considered eligible for inclusion only if the book was published in English and had at least 40 reviews written.

To identify eligible candidates, I needed to first collect book metadata from the Goodreads website. For this, I used the *random* feature available at http://www.goodreads.com/random to scan random book pages on the Goodreads site and extract the metadata for that book. For each book, the following data points were scraped:

Book Data Attributes Collected

Data point	Description
ID	The Goodreads book ID
Title	Book title
Original title	The Book's original title (e.g. native language title)
Author	Name of book author (first author if multiple)
Published	Publication date
Language	Language of book edition
Avg. Rating	The average user rating based on ratings of 1-5

Rating Count	The number of user ratings
Review Count	The number of user reviews
Genre 1	The #1 book genre based on user votes
Genre 2	The #2 book genre based on user votes
Genre 3	The #3 book genre based on user votes
To read	The count of users who have this book on their “to read” shelf.
Currently reading	The count of users who have this book on their “currently reading shelf.
Favorites	The count of users who have this book on their “favorites” shelves.

Initially, I had wanted to include book attributes like Genre, shelves, publication date, and others as features in this analysis which is why these additional data points for each sampled book were collected. Ultimately however, these values were not included as features in the analysis.

Book information was scraped for 95,000 items in Goodreads database using the Python BeautifulSoup module. After collecting the data and examining, it was seen that much of the information collected was redundant. Books often appeared multiple times as different editions of the same book (e.g. different edition numbers, publication dates, language editions) are included as separate items. However, all editions of the book share the same reviews, review counts, and other attributes. Thus, it was necessary to eliminate these duplicates. This was done by looking for books that shared the same “Original Title” value and eliminating duplicates, retaining only one book ID value for each distinct Original Title. Once duplicates were eliminated, books that have less than 40 reviews were also excluded. This left 7,615 book IDs for which I would collect review text.

Extracting the review text was done using a similar process as with collecting the book metadata using the Python BeautifulSoup module. For the 7,615 books that I determined were eligible for inclusion in this analysis, reviews were scraped and saved. Due to a limitation with pagination from the reviews HTML, I was restricted to collecting a maximum of 300 reviews for each book, resulting in a total 1,366,205 distinct reviews collected.

The following data points were collected for each review

Review Data Attributes Collected

Data point	Description
Review ID	The Goodreads review ID
Book ID	The Book ID associated with this review
Review Date	The date the review was submitted
Rating	The score from 1-5 that the reviewer gave the book. Not all reviews include a rating.
Review Text	The raw review text

Data Wrangling

After completing collection of the review text, it was necessary to perform data cleansing with the aim of extracting each individual word from the review for scoring. Punctuation excluding characters ' and - were stripped from the review text. Individual words were identified as being separated by spaces.

Reviews with fewer than 30 words were then excluded as it was thought such short reviews might not have had enough matches with the sentiment lexicons to provide meaningful information.

For each eligible review (having more than 30 words), each individual word from the review was extracted and saved to a table, along with the review ID number, so that the review words could be associated with a distinct review. Additionally, the number of words that were written in all caps and the number of exclamation points used were saved to be used as features in my prediction model.

Sentiment Scoring

Four separate sentiment lexicons were used to score each review. Multiple sentiment lexicons were included to provide as much data as possible for our prediction model.

Sentiment Lexicons Used in Analysis

Lexicon	Words	Description
AFINN	2477	Sentiment scores ranging from -5 to +5
Bing Liu	6787	Polarity scores. 0 for negative, 1 for positive
Harvard Inquirer	3629	Polarity scores. 0 for negative, 1 for positive
MPQA	6901	Polarity scores. 0 for negative, 1 for positive

The AFINN lexicon provides word scores ranging from -5 for most negative to +5 for most positive, while the others only include a binary polarity score of either Positive or Negative. So that I could work with these polarity designations numerically, I assigned a score of 0 to negative words and 1 to positive ones. The Harvard Inquirer and MPQA lexicons include additional features for each word including attributes such as emotional association. However, for this analysis I only wanted to look at positive or negative word sentiment.

These four lexicons were joined to my review words list, resulting in a data frame with each individual word along with the sentiment score for each lexicon where it matched.

Feature Selection

With each word scored, I then was able to prepare summary by grouping the words and scores by review ID and running various aggregate functions and computed columns, which were to be used as features in my prediction model. The following data points were collected for each review.

Features Used in Analysis

Feature	Description
Review ID	The Goodreads review ID
Rating	The user assigned rating from 1-5, or 0 for unscored reviews.
Word Count	The total count of words for this review
AFINN Mean	The mean score of all review words that matched with the AFINN lexicon
Bing Mean	The mean sentiment of all review words matched with the Bing Liu lexicon
MPQA Mean	The mean sentiment of all review words matched with the MPQA lexicon
Inquirer Mean	The mean sentiment of all review words matched with the Harvard lexicon
AFINN Median	The median score of all review words that matched with the AFINN lexicon
Bing Median	The median sentiment of all review words matched with the Bing Liu lexicon
MPQA Median	The median sentiment of all review words matched with the MPQA lexicon
Inquirer Median	The median sentiment of all review words matched with the Harvard lexicon

AFINN Sum	The summed score of all review words that matched with the AFINN lexicon
Bing Sum	The summed polarity values of all review words matched with the Bing Liu lexicon
MPQA Sum	The summed polarity values of all review words matched with the MPQA lexicon
Inquirer Sum	The summed polarity values of all review words matched with the Harvard lexicon
Positive AFINN Count	The count of words in the review that are positive according to the AFINN lexicon (score > 0)
Positive Bing Count	The count of words in the review that are positive according to the Bing lexicon (polarity = 1)
Positive MPQA Count	The count of words in the review that are positive according to the MPQA lexicon (polarity = 1)
Positive Inquirer Count	The count of words in the review that are positive according to the Bing lexicon (polarity = 1)
Negative AFINN Count	The count of words in the review that are negative according to the AFINN lexicon (score < 0)
Negative Bing Count	The count of words in the review that are negative according to the Bing lexicon (polarity = 0)
Negative MPQA Count	The count of words in the review that are negative according to the MPQA lexicon (polarity = 0)
Negative Inquirer Count	The count of words in the review that are negative according to the Bing lexicon (polarity = 0)
Total AFINN count	Count of words in the review that matched the AFINN lexicon. The sum of Positive and Negative AFINN count.
Total Bing count	Count of words in the review that matched the Bing lexicon. The sum of Positive and Negative Bing count.
Total MPQA count	Count of words in the review that matched the MPQA lexicon. The sum of Positive and Negative MPQA count.
Total Inquirer count	Count of words in the review that matched the Inquirer lexicon. The sum of Positive and Negative Inquirer count.
Positive AFINN ratio	The ratio of positive AFINN count divided by the total AFINN count.
Positive Bing ratio	The ratio of positive Bing count divided by the total Bing count.
Positive MPQA ratio	The ratio of positive MPQA count divided by the total MPQA count.
Positive Inquirer ratio	The ratio of positive Inquirer count over the total Inquirer count.
Negative AFINN ratio	The ratio of negative AFINN count divided by the total AFINN count.
Negative Bing ratio	The ratio of negative Bing count divided by the total Bing count.
Negative MPQA ratio	The ratio of negative MPQA count divided by the total MPQA count.
Negative Inquirer ratio	The ratio of negative Inquirer count over the total Inquirer count.
Positive AFINN density	The ratio of positive AFINN count divided by the total review word count.
Positive Bing density	The ratio of positive Bing count divided by the total review word count.
Positive MPQA density	The ratio of positive MPQA count divided by the total review word count.
Positive Inquirer density	The ratio of positive Inquirer count divided by the total review word count.
Negative AFINN density	The ratio of negative AFINN count divided by the total review word count.
Negative Bing density	The ratio of negative Bing count divided by the total review word count.
Negative MPQA density	The ratio of negative MPQA count divided by the total review word count.
Negative Inquirer density	The ratio of negative Inquirer count divided by the total review word count.
AFINN Words Ratio	The ratio of total AFINN count divided by the total word count.
Bing Words Ratio	The ratio of total Bing count divided by the total word count.
MPQA Words Ratio	The ratio of total MPQA count divided by the total word count.
Inquirer Words Ratio	The ratio of total Inquirer count divided by the total word count.
Caps Word Count	The count of review words written in all capital letters
Exclamation Count	The count of exclamation points used in the review.
All Caps Density	The ratio of Caps Word Count divided by the total review word count

Data Storage

Because the quantity of data collected ended up being so large, CSV files proved to be an ineffective storage method. Instead, I determined that a SQLite database would be better suited in storing and retrieving the data for this project.

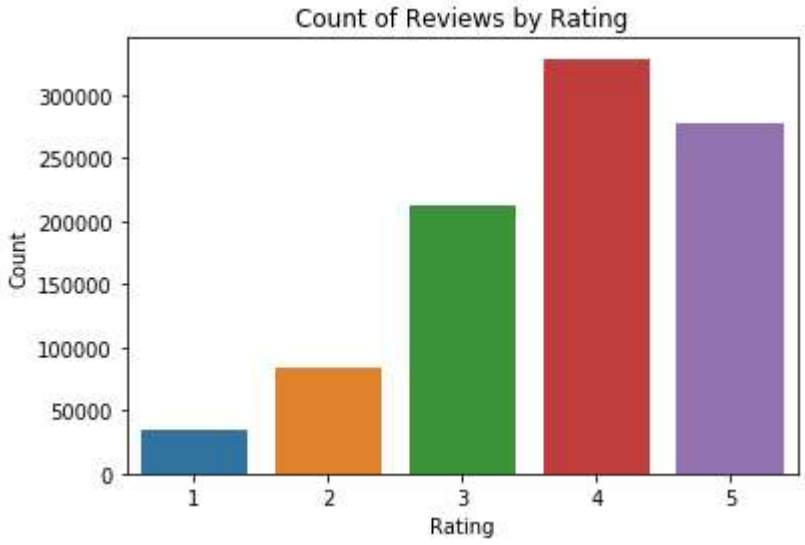
Database Tables

Table	Description
afinn_lexicon	Words and sentiment scores (-5 to 5) from the AFINN sentiment lexicon
bing_lexicon	Words and polarity scores (0 for negative, 1 for positive) from the Bing Liu sentiment lexicon.
book_info	Book metadata collected in the first part of the data collection process.
book_info_cleanred	Cleaned book info data with duplicates removed.
inquirer_lexicon	Words and polarity scores (0 for negative, 1 for positive) from the Harvard Inquirer sentiment lexicon.
mpqa_lexicon	Words and polarity scores (0 for negative, 1 for positive) from the MPQA sentiment lexicon.
review_features	Information about special features “Exclamation count” and “All Caps”
review_stats	Summarized statistics
review_words	Contains the review ID and individual words on each row
reviews	The raw review text, along with ID, date, and rating collected in the second part of the data collection process.

Exploratory Data Analysis

With the data collected and stored, and my features computed, it was time to begin exploratory data analysis.

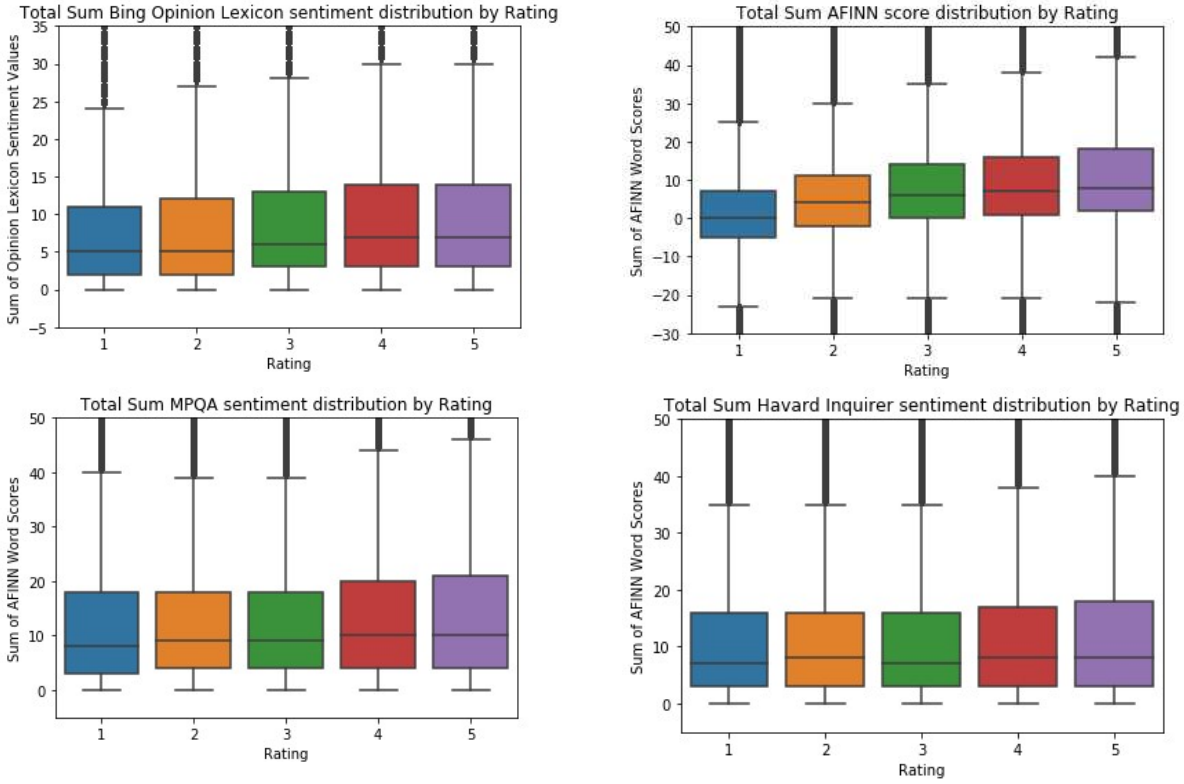
The aim of this classification analysis is to attempt to predict the rating based on the review sentiment and the associated attributes. For this reason, reviews that were unrated (have a score of 0) were dropped from my dataset.



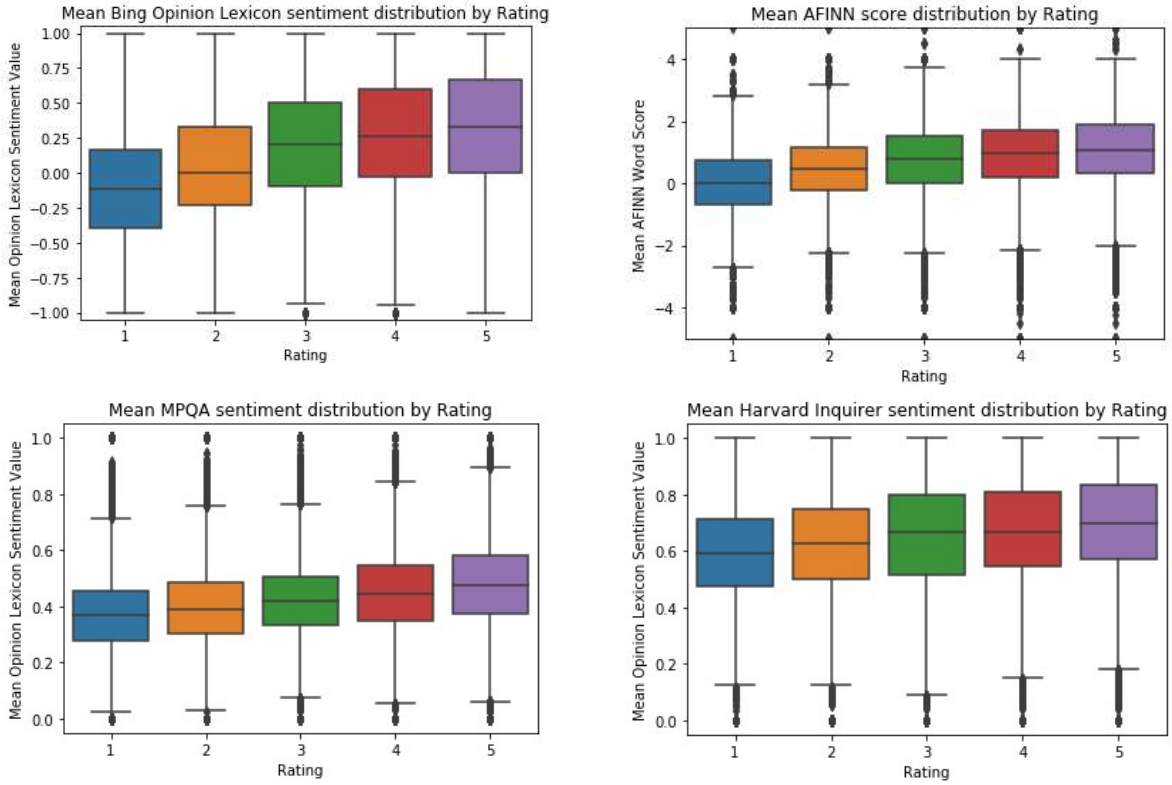
Examining the count of reviews for each rating, it reveals that there is an uneven distribution in the ratings left by reviewers. Reviews with a rating of 1 were by far the least common, while reviews with a rating of 4 the most common. Ratings of 2 were also far less frequent than other ratings. This might indicate that readers who did not enjoy a book were less likely to take the time to write a review. This skew in the distribution of ratings might also affect the results of the analysis as there are far fewer data points for low rated reviews than high rated.

Creating box plots for the distribution of summary statistics features, such as the mean, median, and sum of scores for each lexicon, reveal that there is a clear directional trend in the data, such that higher rated reviews are associated with higher IQRs. This observation can be seen for each summary statistic across all the lexicons. However, also seen in these boxplots are the very long tails indicating that much of the data is outside the interquartile range and is widely spread out.

Distributions of Sentiment Score Sums by Rating

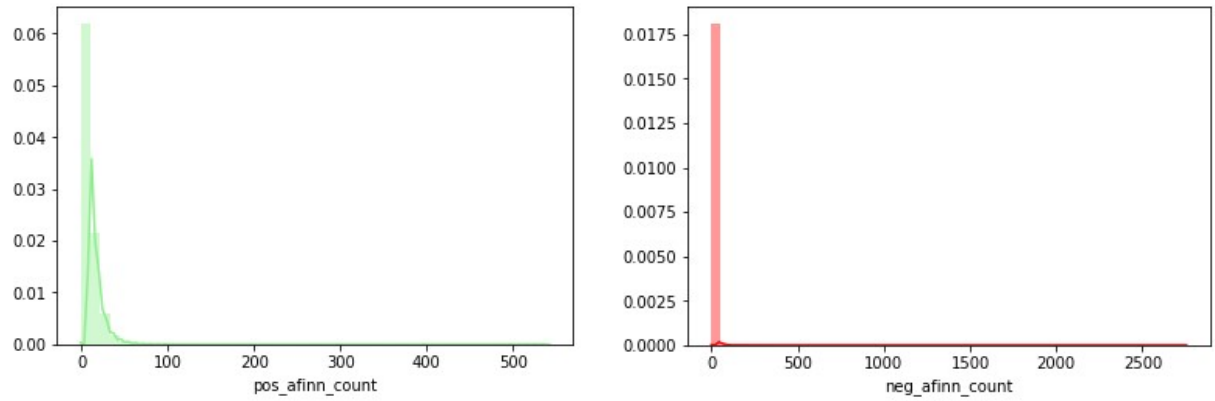


Distributions of Sentiment Score Means by Rating



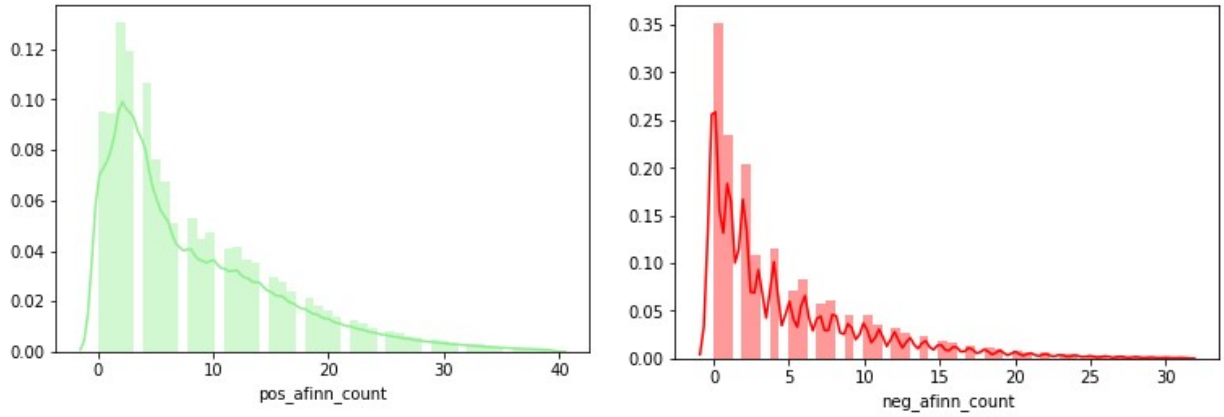
Extreme outliers can also be seen in other features, such as the count of positive or negative lexicon words in each review. The kernel density plots below show the distribution of positive and negative AFINN word counts per review, we can see that most of the values are grouped together, but with several extreme values far away from the mean.

Density of Positive and Negative AFINN Words by Review – Outliers



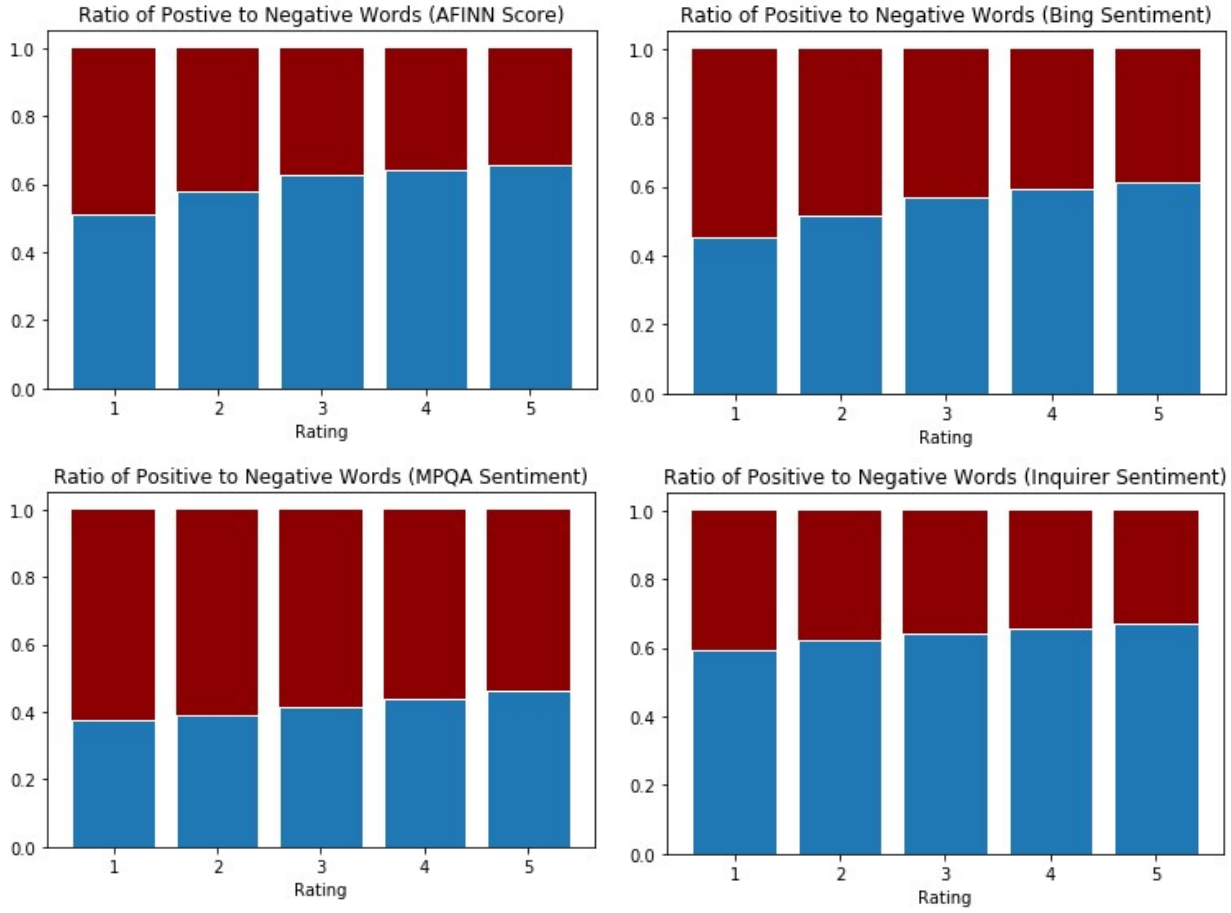
To eliminate these outliers and prevent them from adversely impacting the results of the classification analysis, I chose to remove observations where the value for a feature was more than 3 standard deviations away from the mean. Discarding these extreme outliers resulted in more normal looking distributions.

Density of Positive and Negative AFINN Words by Review – No Outliers



The ratio of positive to negative words in each lexicon (outliers removed), grouped by rating, also revealed a pattern indicating an upward trend between the rating and the positive to negative ratio. This trend is consistent across lexicons.

Ratio of Positive to Negative Words by Rating



My exploratory data analysis indicates that there is a clear directional trend for each feature I examined. This was an encouraging finding as it indicated that higher ratings are associated with attributes such as higher sentiment values and greater counts positive word counts.

Classification Analysis

The aim of this analysis is to attempt to predict the user rating based on features derived from the sentiment analysis. Though the values are numeric, the ratings scores are discrete variables making this a problem better suited for classification analysis rather than regression.

Feature Selection

For this analysis, I began by importing data from my *review_stats* table into a pandas Data Frame. Review ID is not a feature for this analysis and was only used as a join key, so the ID column was dropped. With the remaining columns, rating becomes the target variable. All remaining columns are used as features.

Because scikit-learn classifiers cannot use variables which have missing values, I needed to decide what to do regarding reviews that had fields missing. For each observation, all the features are based on summary statistics of the sentiment lexicon values for the words in that review. If none of the words in a review matched with any word in a sentiment lexicon, this would result in a missing value for the feature columns related to that lexicon. For instance, if a review had words that matched the AFINN, MPQA, and Bing lexicons, but not the Inquirer lexicon, then all features related to the Inquire lexicon (mean, sum, count, ratios) would have meaningless or missing data. Rather than attempting to impute missing values, I instead decided to simply drop any rows that had missing, indicating none of the words in that review matched with at least one word in all four lexicons used. This left a total of 877,941 observations to be included in the analysis.

Model Selection

Using a test set size of 20%, I ran my data though several of the classifiers available in scikit-learn. I ran my data through all the following classifier models, performing cross-validation with RandomSearchCV to optimize parameters. The table below shows the results:

Five Category Classification Model Results Summary

Classifier	Score	Avg. Precision	Avg. Recall
K-Nearest Neighbors	0.32	0.30	0.32
Decision Trees	0.27	0.26	0.37
Naïve Bayes	0.28	0.36	0.28
MLP	0.37	0.37	0.38
Random Forest	0.38	0.38	0.38

These results were less accurate than I had anticipated, with accuracy of only around one third. These results are after I had tuned the hyperparameters using RandomSearchCV.

The classification report for the MLP model reveals that there is higher accuracy for higher ratings, with 5 achieving a precision of 50%. This might be the result of the skew in the rating score distribution where the bulk of the data is composed of high rated reviews. Notice also that the 4s have a recall of 0.83, indicating that the model frequently predicted 4 incorrectly. This result might also be expected from the

distribution of review scores, as 4 is by far the most frequently chosen rating. 1s and 2s are the least frequently chosen and have the poorest accuracy in the prediction models. That might be due to lower available number of samples in these categories with which to train the model.

MLP Model Classification Report

Rating	Precision	Recall	F1 Score	Support
1	0.29	0.03	0.06	6602
2	0.23	0.00	0.01	15786
3	0.30	0.08	0.12	39794
4	0.36	0.83	0.50	61421
5	0.50	0.22	0.31	51986

Model Improvement

To improve the accuracy of my models, I tried several different methods. I attempted to use scikit-learn scalers such as MinMaxScaler and StandardScaler to standardize the values of the features. However, using scalers did not result in any meaningful improvement in the model accuracy.

I also attempted down-sampling the data to get an equal number of observations from each rating category. As seen above, the distribution of ratings is heavily skewed toward higher ratings. To adjust for this, I created a new data frame containing an equal 30,000 observations from each category. Applying the MinMaxScaler transformation to the down-sampled dataset and using the Random Forest classifier, the results were even less accurate than before, with a resulting score of 0.27.

Following up on this approach, I also attempted to balance the data using the Random Forest Classifier parameter *class_weight* set to '*balanced',* which adjusts the weights for each category based on frequency. The MinMaxScaler was also applied to this class weight balanced model However, this attempt produced similar results as my attempt to manually down sample the data into equal sized sets, resulting in an average precision of 0.31.

Another approach I tried was to reduce the number of features. Using the feature importance attribute of the Random Forest classifier, I removed several features that were deemed by the classifier to have low importance. Unfortunately, this also did not improve the model accuracy and instead decreased it. Reducing the features to only the 10 most important, the Random Forest classifier was able to achieve a score of 0.32, less than when all available features were included.

Instead of trying to have the model predict five different rating categories, I attempted to separate into five separate binary classification problems. For each of the five rating, I created a model that attempted to predict whether the review was in that rating group or not.

Using the MLP Classifier, I created the five separate models. The below table shows the results:

MLP Two Category Classification Model Results Per Score

Rating	Score	Precision – 0	Recall – 0	Precision – 1	Recall – 1
5	0.69	0.72	0.91	0.45	0.17
4	0.64	0.65	1.00	0.50	0.00
3	0.77	0.77	1.00	0.00	0.00

2	0.91	0.91	1.00	0.00	0.00
1	0.96	0.96	1.00	0.00	0.00

In the above table header, 1 indicates that an observation was predicted to be in the specified rating category, while 0 means the model predicted the observation was not in that rating category. While the accuracy is improved over the five-category classification model, we also see that the model achieves the accuracy it does largely by predicting negative for each observation (for ratings of 4, 3, 2, and 1). This indicates the finding is meaningless, as the model simply guessed negative for all observations.

Lastly, instead of trying to classify into five different categories, I attempted to turn this into a binary classification problem to see if I could predict positive or negative score. Reviews rated 4 or 5 are designated positive, while scores of 1, 2, or 3 are designated negative.

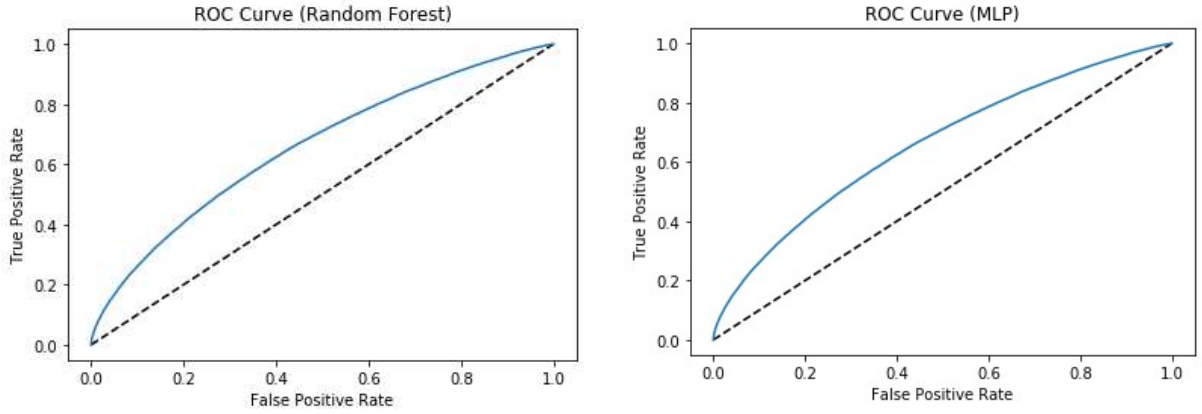
With this configuration for the data, I applied the Random Forest and MLP classifiers which have shown the best accuracy.

Two Category Classification Model Results Summary

Classifier	Score	Precision – Neg.	Recall – Neg.	Precision – Pos.	Recall – Pos.
MLP	0.66	0.55	0.22	0.68	0.90
Random Forest	0.62	0.66	0.78	0.53	0.37

When configured as a binary classification problem, the accuracy of the classifiers is greatly improved, though still not nearly accurate enough to make meaningful predictions. The ROC Curves below plot reveals that the accuracy of the model is lacking at all thresholds between 0 and 1, with a high proportion of false positives.

ROC Curve Plots for Two Category Classification



Future Improvements

In retrospect, storing each individual word as a separate row and then joining to the sentiment lexicon tables was likely the wrong approach for the sentiment scoring. In addition to resulting in a very sizeable table using a large amount of disk space, the joining operation also performed very slowly. A "bag of words" approach for each review would have likely been much more efficient. Instead of joining the individual words to a table of sentiment scores for those words, each review could have been run through a function that would return the sentiment score and other features per review. This would have likely not only been faster, but also saved considerable storage used for my review words table.

Certain additions to the dataset may also improve the model's accuracy. When building my models, it became clear that including additional features resulted in better model accuracy, while dropping features, even ones deemed low "importance" according the model's coefficients, always returned more accurate predictions even if only slightly so. By identifying and including additional features it's possible that the model might return more accurate predictions.

Features used in this model were primarily derived from the sentiment scores of the words in each review. However, two additional features that I included were the count of all caps word, the density of all caps words to total words, and number of exclamation points in each review. Though these additional features did not boost the accuracy of the model to any significant degree, excluding them did reduce the model's accuracy. Thus, inclusion of additional features based on traits of the review not associated with sentiment may produce better results.

Inclusion of additional sentiment lexicons might also boost the accuracy of the model. When I first assembled this data, I began with only two lexicons: AFINN and Bing. By adding the additional two lexicons and the resulting word scores, the model was able to boost its accuracy, though only by a small degree. The inclusion of additional word sentiment sources may improve the model's accuracy by providing better indications of the overall review sentiment.

Lastly, the inclusion of additional text analysis beyond positive or negative word sentiment might aid in improving the predictive abilities of the model. While certain sentiment lexicons like AFINN and Bing provide only positive or negative score values, the Harvard Inquirer lexicon goes much deeper and provides data for emotional sentiment and the strength of the sentiment for each word in the lexicon. By adding a dimension for emotion or other word attributes beyond positive or negative, the model may be able to make more accurate predictions.

Conclusion

From my analysis, it appears that accurately predicting a user's final score based on sentiment analysis is not possible using my model. Even after increasing the number of sentiment lexicons used, and thereby the number of derived features used in the analysis, the increase in the model score was only marginal. The average score (overall accuracy) of the five classification models I attempted was only 0.314, with the best being 0.38. In other words, my models are only accurate in their predictions about a third of the time. Breaking the analysis down into individual ratings, we see that the model's predictions are even less useful. While the accuracy scores saw a boost after converting the problem into two category classification attempting to predict whether a review falls into a particular score bucket, these results are achieved only by the model making uniform predictions, guessing 0 (review not in that rating category) for nearly every

observation in the data. My last attempt at converting the classification problem into two categories for positive (review score of 4 or above) and negative (review score of 3 or below) did result in a model that could classify reviews with a reasonable degree of accuracy, but the score of 0.66 from the MLP model and 0.62 for Random Forest indicates that there's still significant inaccuracies in the model's predictions.

My exploratory data analysis reveals that there's a clear trend between rating and the distribution of the sentiment features. This, along with the findings of my modeling reveal that although my model can't predict with high accuracy, there is an association between sentiment and review score, and text sentiment analysis does have some predictive ability in garnering that association. Sentiment analysis might be one leg used in building a model. By adding additional features based on the characteristics of the text, predicting final score with high accuracy may be possible. Sentiment analysis is a powerful tool, but this outcome reveals that there are limitations and that by itself sentiment analysis can't pinpoint the exact feeling, as expressed by score, that a reader holds.