Marissa Shand (mes7jt)
Cory Clayton (acc2ds)
Mike Pajewski (mtp9k)
Jordan Machita (jm8ux)

Amazon Review Sentiment Analysis

Abstract

We used PySpark to implement sentiment analysis on a large dataset (over 2 GB) of Amazon product reviews. We treated this as a binary supervised learning problem where reviews with a rating of three stars or below were categorized as negative and above three stars were positive. We implemented logistic regression, random forest, and gradient boosted random forest models. Comparing their performance, we determined that logistic regression was the most successful based on accuracy, recall, precision, and F1.

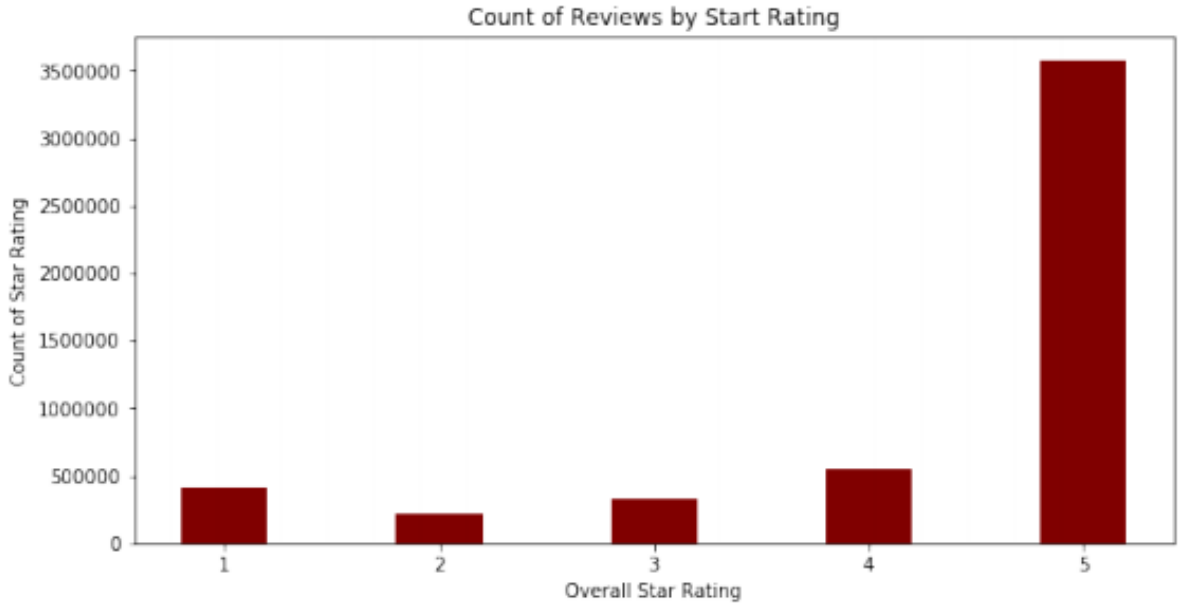
Research Question

We set out to answer the question: Can we predict whether Amazon reviews are positive or negative based on the review text? The usefulness of answering this question is that Amazon could implement our model to identify extremely negative or extremely positive reviews that they may want to have customer service follow up on. While they could do this solely by looking at the star rating, using sentiment analysis would allow them to identify reviews whose textual sentiment does not match the rating. Effectively, this could prove to be a useful backup measure for identifying extremely negative reviews that may have been given a less negative star rating.

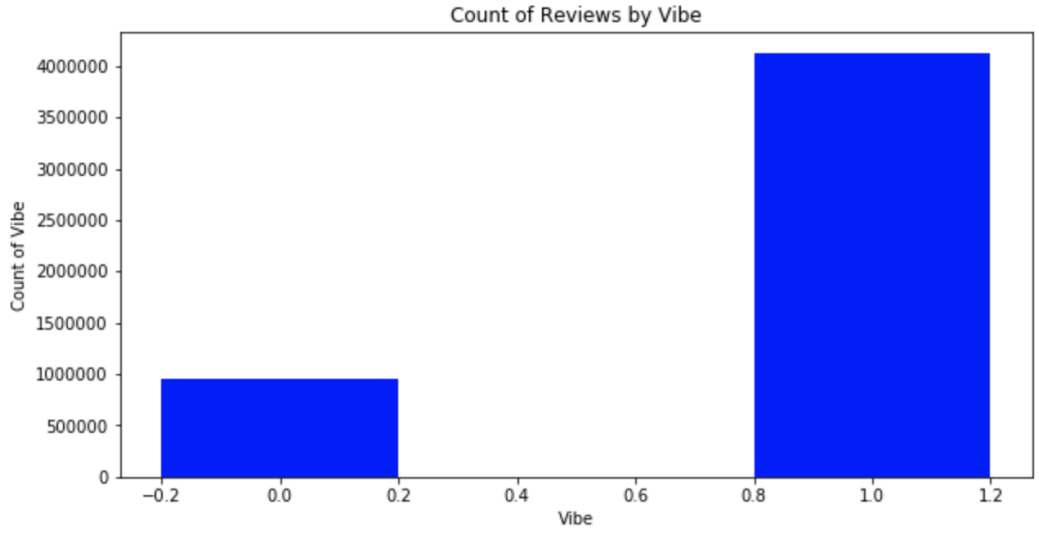
Data

The full dataset from which we extracted a subset contains 142.8 million Amazon product reviews spanning from May 1996 to July 2014 and was sourced by Julian McAuley at UCSD. We chose to focus solely on reviews for products in the Grocery and Gourmet Food category, which narrowed our dataset down to 5,074,160 reviews and 2.05 GB.

For the purpose of answering our question of predicting sentiment based on review text, we needed to create a sentiment label before implementing a supervised learning approach. We used the star rating of the review to gauge sentiment. Figure 1 shows the distribution of star ratings. We can see that most product reviews are given five stars.

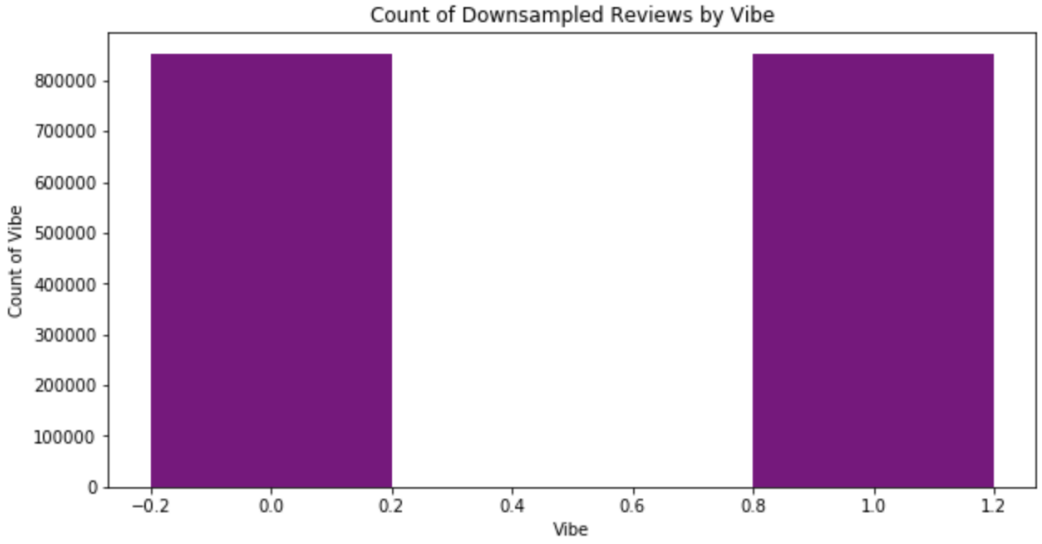


Since we have framed this problem as binary classification, we need to categorize the reviews as positive or negative. To do this, we set the threshold of positivity at three stars, meaning that any reviews with above three stars will be categorized as positive and any with three stars or below will be categorized as negative. We created a new variable called vibe to represent the positive or negative sentiment. The positive reviews are given a vibe of 1 and the negative reviews are given a vibe of 0. We can now see the distribution of reviews by vibe in Figure 2.



With around 4 million positive reviews and less than 1 million negative reviews, we are clearly working with an imbalanced dataset. Thus, in order to account for this and to avoid achieving 80% accuracy by simply predicting a positive label, we downsample our positive reviews in our training data after breaking out a test set. This allows our models to more evenly be able to

predict positive and negative labels. We show the updated distribution of reviews by vibe after downsampling in Figure 3.



Methods

Getting text in a format that machine learning algorithms can process takes some effort. First cleaning the review text by setting all words to lowercase and removing punctuation ensures that the same word is treated consistently. Stopwords were also removed as they do more syntactical work dealing with grammatical structure, leaving the semantic words that provide meaning to the reviews.

Sentences are too complex for computers to work with initially, so the reviews are split into a list of words called tokens. To have a more space efficient as well as model efficient representation, the review tokens are converted into a term frequency vector using the Hashing Term Frequency function. Finally the term frequency vectors are scaled using their inverse document frequency (IDF) which down weights features that appear more frequently and offer less information, and scales up terms that are used less frequently and provide more information. The weighted term frequency vectors are used as the input feature in the machine learning models using the binary sentiment as the label.

Models

To model this data, we used three different models: a logistic regression model, random forest classifier, and gradient boosted tree classifier. We decided to use three different methods to compare the performance of each of the models on our data.

For the logistic regression, we first fit a logistic regression with the max iterations set to 100. We then used the CrossValidator from the pyspark's ml.tuning package to tune hyperparameters using a parameter grid. While setting the max iterations of 100, we tuned different values of the regularizer parameter and threshold for the classification and found the best model to have a regularizer parameter of 0.1 and a threshold value of 0.4.

For the random forest classifier model we initially fit the model with a number of 100 trees, max depth of None, and max bins of 32. We then used CrossValidator from the pyspark's ml.tuning package to tune hyperparameters using a parameter grid. Cross validated a max depth of 5, 8, 10 and number of trees of 10, 20, and 30. The best model from the CrossValidator used 30 trees and a max depth of 10.

After using the CrossValidator to tune the random forest classifier we wanted to use what we had learned from this to choose the hyperparameters for the gradient boosted trees model. However, the gradient boosted tree classifier created a number of issues when trying to train the model. Using a max depth of 10 and num iterations (number of trees in the ensemble) of 30 always produced a memory error. Our final Gradient Boosted Tree model was set to a max depth of 5 and a max iterations of 20. This model took significantly longer to train than the logistic regression or the random forest models.

Results

Changing the values of the hyperparameters of the logistic regression model caused the accuracy of the model to increase by 3% from 86.9% to 89.9%. This halved the number of false positives while doubling the number of false negatives, creating a more equal balance between the two. Consequently, the precision and recall are now about the same for this model, while increasing both the accuracy and F1 measure.

Changing the values of the hyperparameters of the random forest model caused the accuracy of the model to increase by about 5% from 75.2% to 80.0%. Similar to the logistic regression model, changing the hyperparameters of this model caused the number of false negatives to increase slightly while decreasing the number of false positives. This resulted in the increase of all the metrics, accuracy, recall, precision, and F1 measure.

The gradient boosted tree model has the worst performance out of the three models. It was much more likely to predict the review was positive than negative, which resulted in the precision to be very high while the accuracy remained low.

Summary of Metrics

	Accuracy	Recall	Precision	F1
Logistic Regression	0.899	0.943	0.933	0.938
Random Forest	0.800	0.841	0.906	0.872
Gradient Boosted Trees	0.619	0.561	0.949	0.705

Conclusions

As a conclusion we consider this experiment a success as our logistic regression model is able to correctly predict the sentiment of a review from only the text at nearly 90% accuracy using the logistic model. For the data we used the logistic model outperformed both tree based models. Tree based models typically work better with tabular data as it makes for easier decision boundaries. The model that was produced has potential value and use in other areas to help classify food reviews where a star rating label might not be available.

Further research can be done by changing the sentiment distinction to 2 or 1 stars to try and flag very negative reviews which might be useful for companies to see what is making their customers upset. Also some work could be done to chuck the reviews into sections and try and identify which sections are positive and which sections are negative to help sellers get a more holistic understanding of their reviews.