Big Data Scaling Final Project - Predicting Star Ratings of Amazon Reviews

By: Logan King, Lori Chiu, Sarah Torrence
All code and documentation available at: https://github.com/kingla6/ds-5460-project

Introduction

Our project focuses on product reviews from Amazon. Our motivation for using Amazon related data is that we are a few among the many people who subscribe to Amazon Prime. We are frequent buyers on Amazon and we want to develop a better understanding of how customer reviews are structured to help us make better decisions when buying products on Amazon. More specifically, the goal of this project is to use Natural Language Processing tasks to process the product reviews and predict the user rating associated with each review.

Problem

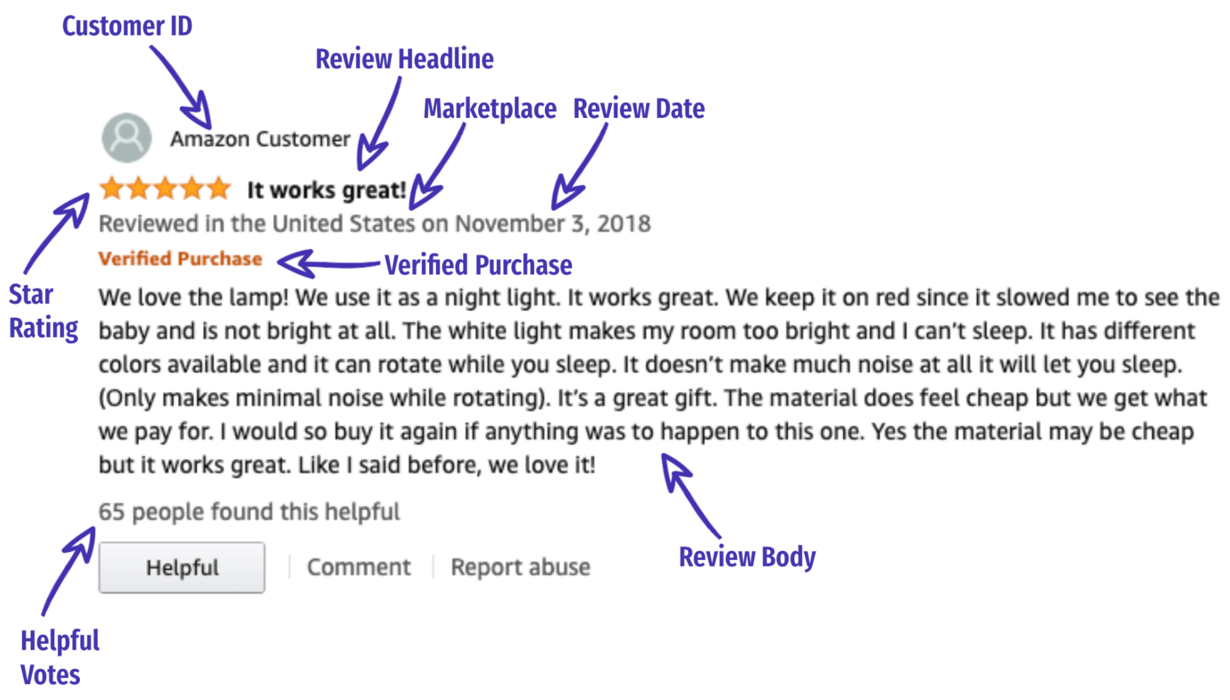
If you have ever shopped on Amazon, you may have noticed user reviews listed under each of the product pages and you may have used these reviews to understand more about those products. Many consumers are using these user reviews to decide whether or not they should make a purchase. They are more likely to purchase something if the ratings are highly rated with more than four stars. However, some users leave reviews that are highly rated without providing textual context to what the advantages of the products are, they leave reviews without providing a star rating for the products, or there are discrepancies in the star ratings and the actual text of the reviews. With this project, we want to understand the helpfulness of these reviews by predicting star ratings based on the text of the reviews. Amazon can use this model to predict star ratings for user reviews that did not have any stars associated with the review. Amazon can accurately assign a star rating based on the text rather than the sentiment of the user. For example, if a user is very angry and leaves a 1-star review but the text of the review is neutral, it should actually be assigned a 3-star rating. Users can use this model to get a more accurate star rating for the reviews to help support their purchasing decisions.

Data

We are using a dataset found on Kaggle that contains US Amazon customer reviews for a wide variety of products. The dataset is 54.41 GB that is broken into 37 different files. Each of the files contains reviews for a different product such as grocery, music, and apparel. The entire dataset contains reviews that span between 1995 and 2015. There are 15 different variables in the data set listed below:

- Product Title
- Star Rating
- Helpful Votes
- Total Votes
- Verified Purchase

- Review Headline
- Review Body
- Product Category
- Review ID
- Product ID
- Product Parent
- Marketplace
- Customer ID
- Vine
- Review Date


Amazon Customer

★★★★★ **It works great!**

Reviewed in the United States on November 3, 2018

Verified Purchase

We love the lamp! We use it as a night light. It works great. We keep it on red since it slowed me to see the baby and is not bright at all. The white light makes my room too bright and I can't sleep. It has different colors available and it can rotate while you sleep. It doesn't make much noise at all it will let you sleep. (Only makes minimal noise while rotating). It's a great gift. The material does feel cheap but we get what we pay for. I would so buy it again if anything was to happen to this one. Yes the material may be cheap but it works great. Like I said before, we love it!

65 people found this helpful

Figure 1. Example of a user review for an Amazon product

Star rating is our outcome variable which is a 1-5 rating provided by the reviewer. We want to focus on the text of the review and the review body in our model as well including other insightful predictors such as the product title and product category. Additionally, we believe the number of helpful votes and total number of votes will be important features as these are votes given to the review by other Amazon users. If a user found the review either helpful or unhelpful, they can vote for either category making it quicker for other users to filter through reviews and find the most helpful ones. The last feature we are using for modeling is verified purchase. This is an indicator of whether the user who wrote the review actually bought that product. This is an important variable as any Amazon user can write reviews or even bots can automate reviews, but reviews that are from verified purchases can be trusted knowing the customer has in fact purchased that product.

Methods

Data Processing

The largest challenge we faced was handling such a large amount of data. As the data set included 37 different files we needed a way of importing all of these files and aggregating the data. We came up with two different methods to approach this problem: using multiprocessing to loop through all the files in parallel and utilizing the pyspark function read.csv function to read in a list of files at once. Although both approaches worked, the pyspark approach was significantly faster and did not require us writing our own function to pool over like the multiprocessing approach. Utilizing pyspark significantly sped up the process, used less RAM and was much quicker and simpler code. With the multiprocessing approach we read the files in as pandas dataframes, did some cleaning and filtering and then created a pyspark dataframe to be used for modeling. However, the pyspark approach enabled us to start with a pyspark dataframe without having to use pandas. The filtering and cleaning could be performed on this aggregated pyspark dataframe.

There were several aspects of cleaning that needed to be performed before creating a pipeline for modeling. First, as review headline and review body were the most important variables for modeling, any observations with missing values in these columns were removed. Due to the large amount of data, we had the flexibility to remove these observations without having any concerns about the size of data for training or testing. We also found some inconsistencies in product categories including dates and long form text. For each of our two approaches we handled this in a different way. When using multiprocessing, as we read in each file, we extracted the product category from the file name and created a new product category for modeling instead of the one provided. However, for the pyspark approach, as we read in all the files at once we could not create our own column using file names. Instead we used the file names to create a list of categories and removed any observations that had values for the product category column that were not in this list to ensure we had accurate and high quality observations for this feature.

Pipeline

Due to the nature of our dataset, which contains mixed data with numeric and string variables, data transformation is required prior to modeling. Our data pipeline allows for transformation of raw data into features which are then used for predictive modeling. In this pipeline, there are several transformation steps: Tokenizers, Stop Word Removers, Word2Vec Transformers, a Vector Assembler, and a Normalizer. Our initial dataset of reviews is fit and transformed using the pipeline prior to modeling, with a specific focus on the following features: Product Headline, Review Headline, Review Body, Product Category, Helpful Votes, Total Votes, and Verified Purchase.

The pipeline first passes relevant string features through a Tokenizer. This breaks up each string into a list of words and automatically converts them to lowercase. The variables affected by this are the Product Title, Review Headline, Review Body, and Category. The resulting tokenized features are then passed into the Stop Words Remover, in order to remove any words from the list that are stop words, which are not believed to add much value to the predictive power of our model. Category is not included in the Stop Words Remover step, as each category only contains a few words. The resulting word lists are passed into the Word2Vec transformer, which extracts numeric features from each list of words. The resulting numeric features are vectors of varying lengths. Due to the nature of each set of word lists, different output lengths are specified for each numeric vector created. Product Title and Review Headline are transformed to vectors of length 5, Review Body is transformed to vectors of length 10, and Category is transformed to a vector of length 1.

Following the string feature transformation phases of the pipeline, all features are passed into the Vector Assembler. This collects all specified features into a single vector for each operation. Our integer features of Total Votes, Helpful Votes, and Verified Purchase are joined into a single feature vector with our new numeric representations of string variables provided by the previous steps of the pipeline. Finally, in order to ensure that all of our features are on the same scale, the feature set just created is normalized with the Normalizer transformation. Following this pipeline for feature extraction and engineering, our data is ready for modeling.

Modeling

In the modeling phase, the resulting features from the pipeline are used to make predictions of Star Rating of each review. To make such predictions, the MLlib modeling functions of Logistic Regression, Decision Tree, and Random Forest are used. These functions are chosen in particular, because they are optimal for multiclass classification problems included in the MLlib library. For each model, Star Rating is specified as our label, while the features are Normalized Features resulting from our pipeline, which are developed from the feature set [Product Title, Review Headline, Review Body, Category, Total Votes, Helpful Votes, Verified Purchase].

The modeling dataset is split using a 70/30 training and testing ratio prior to making predictions. Following data splitting, predictive modeling is conducted and each model's performance is evaluated on the test set in order to determine the best model. Results of this process are discussed in the following section.

Scaling

We were able to import the entire data set of 54GB and process and clean it, however, due to limitations in our pipeline we were unable to fit our pipeline to the entire data set. We ended up modeling over a sample of the data that included 4 of the categories: Baby, Appliance, Digital and Video. We utilized both Google Cloud Platform (GCP) and ACCRE to perform our analysis enabling us to utilize larger data sizes at quicker speeds, however we believe the Word2Vec transformer, used for 3 different variables, was slowing down fitting the pipeline. After running the code for just the pipeline for an entire day, we were not able to get any output and therefore were not able to model on the entire data set. ACCRE was used with 8 CPU and 100GB of

RAM to run the entire process on the subsample utilizing the multiprocessing method. This took a total of only about 15 minutes to run. GCP was utilized to upload the data into a bucket, transfer it into the master node and then copy the file to hdfs. Using hdfs and pyspark we were able to read in and clean and process the entire data set in the matter of minutes.

Results

In comparing the accuracy of the three different models, all three performed relatively equally with an accuracy of approximately 63%. The exact accuracies of the models ordered best performing to least is Decision Tree (63.2%), Logistic Regression (63.1%), and Random Forest (62.8%). As we've mentioned above, we had limitations with scaling the project to our entire large dataset, so the models were tested on just a sample of the data and did not undergo any tuning and instead relied on default parameters. To dig deeper into the results of our predictions, we evaluated where our model correctly and incorrectly made predictions. As shown in Fig. 2, it appears that the 5-star reviews were most correctly predicted at nearly 100%, with 1-star reviews following at just below 50%. The model has a hard time correctly identifying the neutral 3-star ratings. When the model is incorrect, it is most often predicting 5-star reviews by a wide margin. For the incorrect 5-star reviews, a 1-star rating is most often predicted. This is likely due to the fact that more 1-star and 5-star ratings are observed more often than 2, 3, or 4-star ratings. Perhaps our model is just guessing the most commonly occurring types of reviews instead of making reliable predictions.

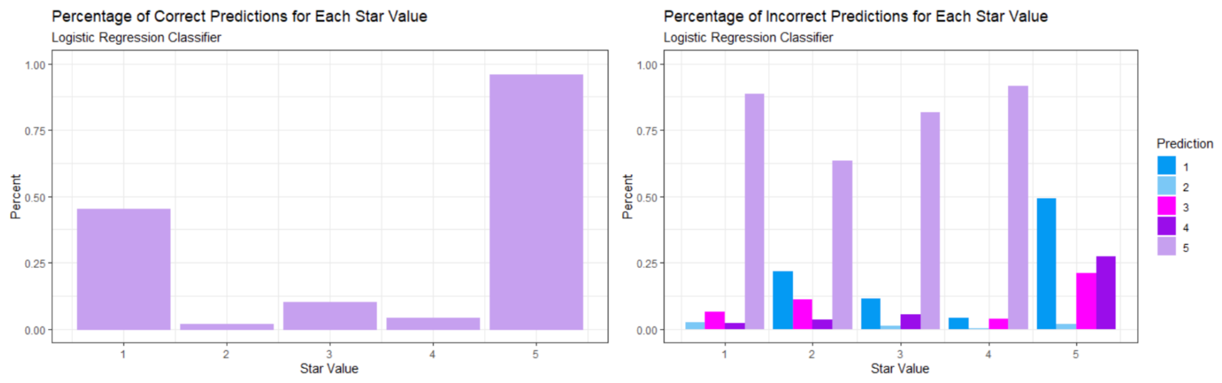


Figure 2. Performance of the Logistic Regression Classifier

As shown in Fig. 3, we see similar results for the decision tree classifier, but with greater concentration at the correct 5-star reviews. One-star reviews have dropped to only 25% correctly predicted. Similar to the Logistics Regression Classifier, this model is almost always predicting 5-star when the predictions are incorrect.

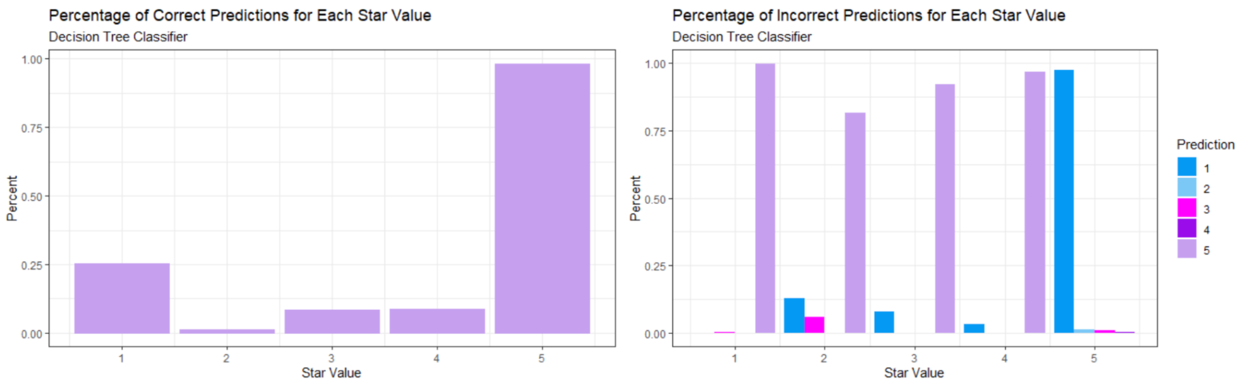


Figure 3. Performance of the Decision Tree Classifier

As shown in Fig. 4, again, we see similar patterns of what is correctly and incorrectly predicted. It is almost exclusively predicting 5-star reviews when the model is incorrect.

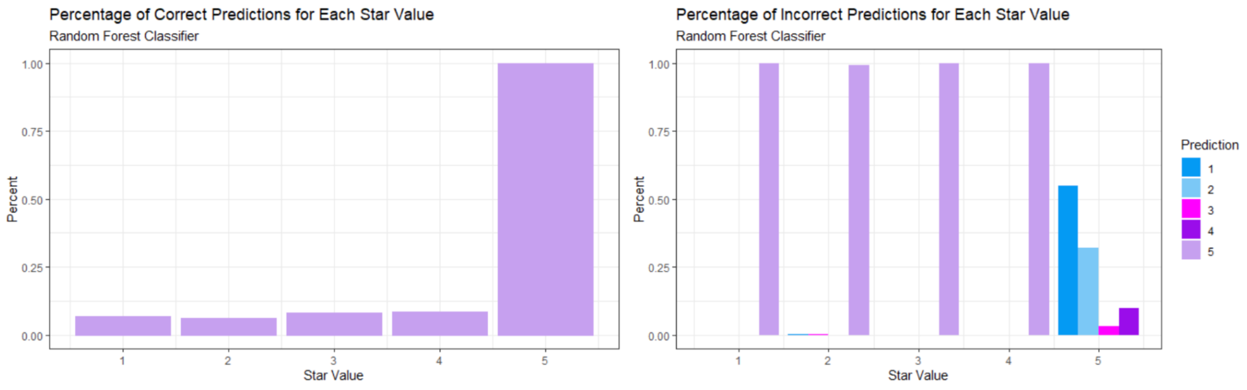


Figure 4. Performance of the Random Forest Classifier

As the accuracy and evaluation of the predictions have shown, the Logistic Regression Classifier is the best model. Again, to reiterate, the model used default parameters and has not been tuned. The results of the models could change if we were able to scale and tune each model.

Conclusion

Due to lack of knowledge in scaling pipelines and modeling to large datasets, we were unable to achieve one of our goals of training a model on the entire 54GB of data. Time permitting, this would be the immediate next steps to enhance this project. Additionally, due to imbalance in the data caused by a heavy proportion of 5 star ratings, it would be beneficial to create a more balanced data set for training to hopefully improve accuracy of model predictions. Our best model had about 63% accuracy, however, predicting a star of 5 when the true rating is 1 is much worse than if the true rating is 4. Handling star rating as an ordinal outcome rather than just a multi-class problem would correctly penalize these differences in errors. Further, rather than just being able to predict star ratings, sentiment analysis or topic modeling to understand the content of reviews and how content might affect star ratings would enhance the understanding of Amazon reviews and their role in how reviewers choose a star rating.