Amazon Kindle Review Data Analysis

(Group 10)
Peter Ayral
Kaden Nguyen
Nathan Ng
Yanjie Qi

03/18/2021

I. Abstract

In this project, we utilized Kindle reviews from Amazon in order to predict the sentiment of the review for the product. We incorporated 9 different features into our models: overall, review_features, summary_features, weightedRating, HasHelpful, asin_vector, reviewTime_vector. We modified the overall column to show 1 if the overall score was greater than or equal to 3 (positive review) and 0 if the overall score was less than or equal to 2 (negative review). In order to clean and preprocess the data we used Tokenizer to make words into arrays and StopWordsRemover to remove basic words. To help accurately predict the sentiment of the reviews we employed models of Linear Regression (benchmark model), Logistic Regression, Naive Bayes, and Random Forest. By evaluating each model based on its test accuracy, test recall, test precision, and F1 score we were able to find our champion model of Naive Bayes as it possessed the great results in terms of its test accuracy, test precision, and F1 score as well as its highly interpretable nature. However, all of our models were pretty accurate as our champion model was narrowed down under the criteria of processing time and interpretability.

<u>Research Question</u>

What Statistical Methods Best Predict the Overall Rating of Amazon Kindle Reviews?

II. Data and Methods

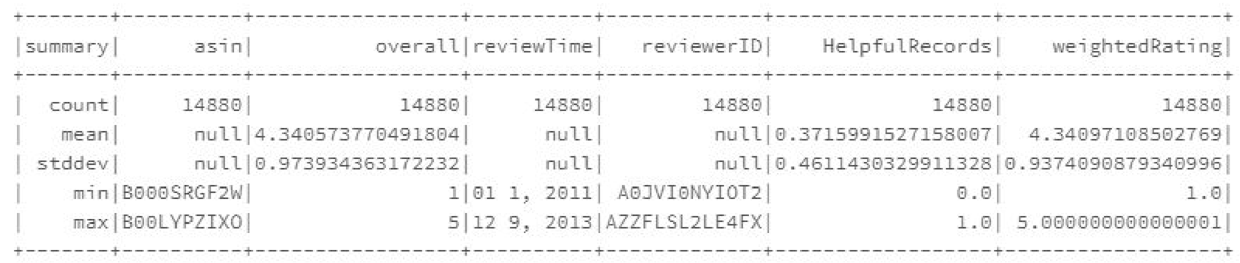
<u>Background of Data</u>

The data was obtained from Kaggle. It contains 14,880 rows of observations along with 10 columns that hold variables relating to information about the product, content of the review, and reviewer. Originally the csv file sourced from Kaggle contained columns ‘reviewTime’.
These were modified through the pipeline into the features: review_features, summary_features, WeightedRating, reviewTime_vector, and asin_vector. Asin is a crucial column of data that allows us to split the dataset into training and test without it being a confounding variable. Asin is not unique per row, and it can be shown how certain Asin’s contain even dozens of (check this) distinct reviews. This can pose issues of bias when asin is unaccounted for.

<u>Data Cleaning</u>

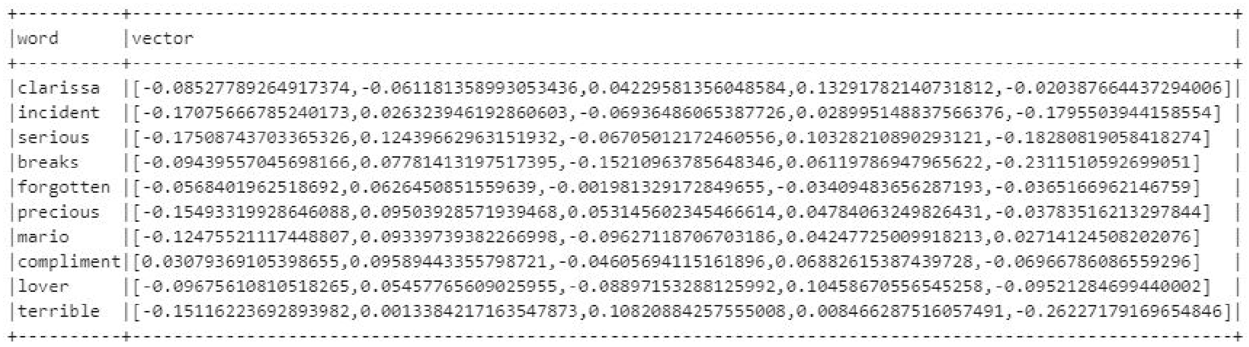
Data had missing values and quotations that needed to be removed. We were able to get rid of the null values when preprocessing the data with .dropna() to any of the columns within the dataset. The original columns are string values, which needed to be converted to variables of numeric type. We also removed stop words (of, the, and) from the reviewText and summary columns of our dataset which led to cleaner analysis. We also noticed certain observations of the data did not load into Spark correctly. For example, one row had a reviewTime that was definitely not a date. We located the problem as columns having extra quotes in the reviewText. So we removed all symbols (“&,%) from the dataset and this allowed us to properly load the data in.

Exploratory Data Analysis



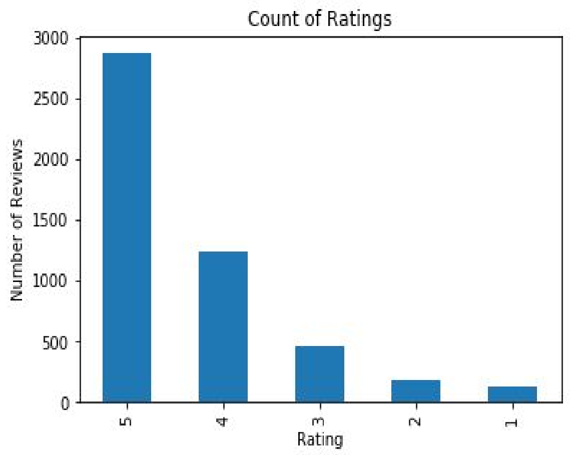
```
+-------+----------+-----------------+----------+-------------+------------------+------------------+
|summary|      asin|          overall|reviewTime|   reviewerID|    HelpfulRecords|    weightedRating|
+-------+----------+-----------------+----------+-------------+------------------+------------------+
|  count|     14880|            14880|     14880|        14880|             14880|             14880|
|   mean|      null|4.340573770491804|      null|         null|0.3715991527158007|  4.34097108502769|
| stddev|      null|0.973934363172232|      null|         null|0.4611430329911328|0.9374090879340996|
|    min|B000SRGF2W|                1|01 1, 2011| A0JVI0NYIOT2|               0.0|               1.0|
|    max|B00LYPZIXO|                5|12 9, 2013|AZZFLSL2LE4FX|               1.0| 5.000000000000001|
+-------+----------+-----------------+----------+-------------+------------------+------------------+
```

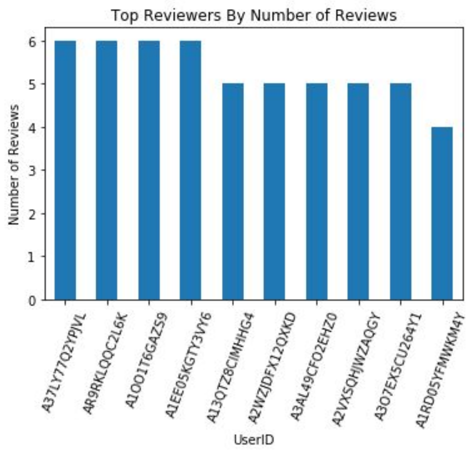
Here we used describe() to get basic statistics from the quantitative features of our dataset. Here we can see that we had 14,880 rows of data as well as having a mean rating of 4.34 and a standard deviation of 0.97. For the helpfulness column, we had a mean of 0.37, with a standard deviation of 0.46. For the weightedRating column, derived from a formula averaging the grouped reviews of each book based on their helpfulness, we also reached a mean of 4.34 and a standard deviation of 0.937.



```
+----------+----------------------------------------------------------------------------------------------------+
|word      |vector                                                                                              |
+----------+----------------------------------------------------------------------------------------------------+
|clarissa  |[-0.08527789264917374,-0.061181358993053436,0.04229581356048584,0.13291782140731812,-0.020387664437294006]|
|incident  |[-0.17075666785240173,0.026323946192860603,-0.06936486065387726,0.028995148837566376,-0.1795503944158554] |
|serious   |[-0.17508743703365326,0.12439662963151932,-0.06705012172460556,0.10328210890293121,-0.18280819058418274]  |
|breaks    |[-0.09439557045698166,0.07781413197517395,-0.15210963785648346,0.06119786947965622,-0.2311510592699051]   |
|forgotten |[-0.0568401962518692,0.0626450851559639,-0.001981329172849655,-0.03409483656287193,-0.0365166962146759]   |
|precious  |[-0.15493319928646088,0.09503928571939468,0.053145602345466614,0.04784063249826431,-0.03783516213297844]  |
|mario     |[-0.12475521117448807,0.09339739382266998,-0.09627118706703186,0.04247725009918213,0.02714124508202076]   |
|compliment|[0.03079369105398655,0.09589443355798721,-0.04605694115161896,0.06882615387439728,-0.06966786086559296]   |
|lover     |[-0.09675610810518265,0.05457765609025955,-0.08897153288125992,0.10458670556545258,-0.09521284699440002]  |
|terrible  |[-0.1511622369289398,0.0013384217163547873,0.10820884257555008,0.008466287516057491,-0.26227179169654846]|
+----------+----------------------------------------------------------------------------------------------------+
```

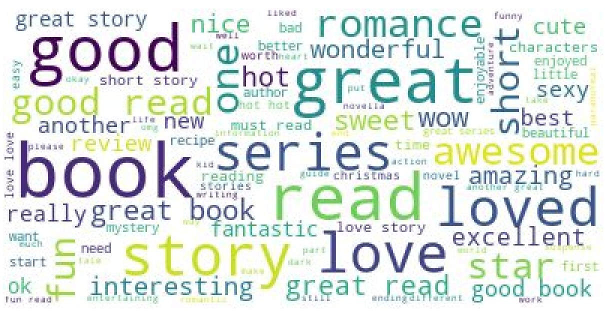
Here is a sample of 10 word vectors from our word2vec. Each word has 5 numbers describing their location in a 5-dimensional space. If we were doing an NLP analysis, this would be very useful.

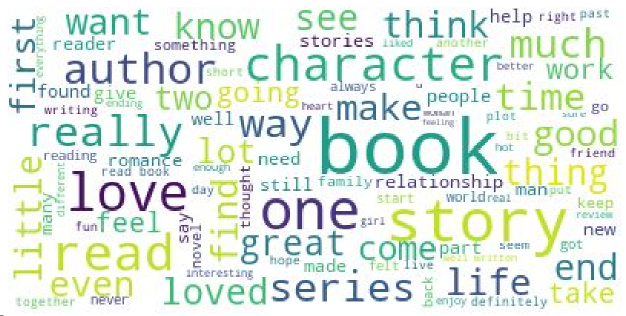




In the first plot, we created a visualization summing the total count of all ratings given within the reviews for the product. The ratings are based on a number scale out of 5 with 5 being the highest and 1 being the lowest. The majority of reviewers gave this product a rating of 5/5 and there was an overwhelming majority that possessed a positive sentiment rating towards the product as a large percentage of the reviewers rated it 3 or above. There was only a small fraction of reviewers who were unsatisfied with the product giving a 1 or 2 rating that conveyed their negative sentiment. In the plot on the right, we are provided with more insights to the behavior of some of the top reviewers. There are many reviewers that write more than one review, with a highest of 6.

Summary **Reviews**





We also created word clouds to represent the summary and review text features of our data set. We removed stop words from the dataset (a, the, I), removed unwanted symbols (&%”), and also tokenized the words. Other than being pretty, they also have an application in understanding the dataset. For a word cloud, the bigger the word, the more time that word appeared in our observations. As one can see, these are interesting because they highlight the reactionary basis of the summary inputs. In the summary category, one sees more reactionary words, such as “good”, “great”, and “love”. However, in reviews, these words still appear, but they are less frequent. In the review text, you get more information about the book, having words like “character” and “author”.

Data Preprocessing and Pipelines

Feature Engineering: Weighted Rating

To show an overall assessment of each product, we did feature engineering to weight each record of rating and develop a new column "WeightedRating". We considered two cases based on the "useful" column: it has records of users thinking the reviewText is useful or it has no records at all.

For the first case, if the helpful records of a product are mixed with rows that have records and rows that have no records. Define ratings of rows that have records as $r_1, r_2, \ldots, r_n$, where their weights are proportion obtained from "Helpful" column, denoted as $P_1, \ldots, P_n$. In the meantime, define ratings of rows that have no records as $NR_1, NR_2, \ldots, NR_m$, with the weights of each of them being 1, then we have:

$$Weighted\ Rating = \frac{(P_1+1)\, r_1 + \ldots + (Pn+1)\, r_n + NR_1 + \ldots + NR_m}{(P_1+1) + \ldots + (P_n+1) + m}$$

Therefore, if the helpful records of a product shows that all of reviews have records of people thinking it is helpful or not, then:

$$Weighted\ Rating = \frac{(P_1+1)\, r_1 + \ldots + (Pn+1)\, r_n}{(P_1+1) + \ldots + (P_n+1)}$$

For the second case, if all of "useful" column of a product shows no users considering any reviews of this product is useful or not, then:

$$Weighted\ Rating = \frac{NR_1 + \ldots + NR_m}{m}$$

Pipeline 1

Pipeline1 is organized so that the data is preprocessed. The final features are suitable for our candidate model chosen from Logistic Regression, Naive Bayes and Random Forest. This pipeline involved Tokenizer, StringIndexer, OneHotEncoder, Vector Assembler, and HashingTF. Tokenizer and HashingTF were the essential components that performed lexical analysis and mapping respectively.

III. Results

Models and Results

We opted to split our data by assigning 80% towards our training set and 20% towards our test set. The models we selected to run consists of Linear Regression, Logistic Regression, Naive Bayes, and Random Forest.

Benchmark Model

For our benchmark model we selected to run a Linear Regression model by implementing the LinearRegression() from the ml.classification package. We were successfully able to fit our model on the training data in order to obtain our coefficients and intercept. Lastly, we calculated our root mean squared error of the training data which was 0.245. For the test data the root mean squared error was a little higher, at 0.255.

Model Comparison

After our Benchmark Model we were able to run the models of Logistic Regression, Naive Bayes, and Random Forest and calculate their test accuracy, test recall, test precision, and F1 scores to evaluate the effectiveness of our models.

For our Logistic Regression model, we applied LogisticRegression() from the ml.classification package on our label column “overall” and fit it on our training data to receive a test accuracy rating of 0.927. In addition, we also calculated the amount of true positives, true negatives, false positives, and false negatives in order to calculate the test recall of 1.0, test precision of 0.933, and F1 score of 0.966.

With our Naive Bayes model, we were able to create the trainer and set its parameters with NaiveBayes(). We then fit it on our training dataset to be able to make predictions on your test set. We also employed the MulticlassClassificationEvaluator() in order to find our test set accuracy of 0.930. Like our previous model we were able to calculate the confusion matrix in order to get our test recall of 1.0, test precision of 0.930, and F1 score of 0.964.

For our last model, Random Forest, we were able to set its parameters with 20 total trees with RandomForestClassifier() and fit it to our training data. To find our predictions we then transformed our model on the test data set where we then used MulticlassClassificationEvaluator() to evaluate our test accuracy of 0.930. The results of our confusion matrix for our Random Forest model consisted of a test recall of 1.0, test precision of 0.930, and a F1 score of 0.964.

Champion Model

When selecting our "champion" model we analyzed each model's test accuracy, confusion matrix metrics, as well as their runtime to determine which model best suited our research topic.

Test accuracy measures the number of correct predictions over the total amount of predictions made. The Naive Bayes and Random Forest model possessed the highest test accuracy barely beating out Logistic Regression by 0.003. For our non-Benchmark models we were able to get a value of 1.0 for all of our test recalls meaning that all of our actual positives were identified correctly for these models. Test precision aims to measure what proportion of positive identifications was accurately correct and we can see that Logistic Regression carried the highest rate of 0.933 only 0.003 better than Naive Bayes and Random Forest. Lastly, the F1 score is utilized in order to compute a model's accuracy as it conveys how robust and precise our classifier is by utilizing the test recall and precision of the model. We see that Logistic regression once again maintains the highest ratio of a F1 score of 0.966 compared to F1 score of 0.964 for Naive Bayes and Random Forest. These are all in fact great F1 scores since the higher the ratio, the better the performance of the model is.

Because Naive Bayes and Random Forest both have low test errors, the criteria for determining the group's Champion Model is focussed on the time taken to process the training as well as interpretability. The random forest model is the fastest model to load, but the Naive Bayes is a highly interpretable model due to the independence assumption [1]. The independence assumption will allow the group to calculate conditional probability and thus clearly describe the effect each feature has to a class prediction. The champion model should be the Naive Bayes. The figure below expresses a conditional probability model under Naive Bayes where C_k is a class. For the data, k: k={0,1}. C_0=Negative overall rating; C_1=Positive overall Rating. x=data matrix.

$$P(C_k|x) = \frac{1}{Z} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

Model	**Test Accuracy**	**Test Recall**	**Test Precision**	**F1 Score**
Logistic Regression	**0.927**	**1.0**	**0.933**	**0.966**
Naive Bayes	**0.930**	**1.0**	**0.930**	**0.964**
Random Forest	**0.930**	**1.0**	**0.930**	**0.964**

IV. Conclusion

Regarding: 'What Statistical Methods Best Predict the Overall Rating of Amazon Kindle Reviews?' The group's research provided statistical evidence that the Naive Bayes model is the 'Champion Model' for predicting the overall rating of Amazon Kindle reviews.

During the research, using the large dataset caused periods of lagging through the cluster. Databricks was utilized as an effective IDE. While case1 has 14880 cleaned rows, a potential case2 could utilize all rows of the 600MB of data.

In the future, we would want to utilize all data from the original csv data. This would require a reliable algorithm that can separate the data so that each asin is fully contained in either the training or test data. The benchmark model and the 3 models made to find the champion model are all supervised learning methods where a response variable was predetermined before any training. Unsupervised learning is another area to explore for research. In addition our models possessed very similar metrics in their model evaluation as this could be due to the measurement of positive sentiments vs negative sentiments. It would be interesting to examine how changing our positive sentiment rating from greater than or equal to 3 to greater than or equal to 4. Thus, making all negative sentiments any rating less than or equal to 3.

V. Bibliography

1. https://christophm.github.io/interpretable-ml-book/other-interpretable.html