Sentiment Analysis and Recommendation Prediction for Amazon Products Reviews

Team information

Team No: 10

Members: Nishi Virenkumar Patel(niship@g.clemson.edu)

Eshaa Deepak Sood(esood@g.clemson.edu)

I. Motivation

Lately the use of various e-commerce websites like Amazon and Flipkart are booming because of the trend of online shopping. Owing to the availability of several e-commerce websites a product can be bought on numerous websites at different prices. Customers usually desire to get the best quality of an item at the lowest cost and reviews given by other customers seem to be the most reliable way to take a decision on a product. Sentiment Analysis has thus proved essential in order to understand a product's popularity among buyers all over the world.

In the recent years Sentiment Analysis has become very popular in text mining and computational linguistic research. Given a bunch of text, sentiment analysis is used to computationally study people's opinions, reviews and emotion towards different entities and get a clear overview about people's attitude toward a particular entity.

However, one of the problems faced is the ratings (number of stars) assigned to a product review does not actually match the sentiment of that review. For an example a person might be very happy with a product but give a score of only two out of five stars on the other hand a person might sound very disappointed but still give five stars.

II. Objective

The objective of our project is to classify customer reviews as positive or negative over different Amazon products and develop a supervised learning model that can effectively polarize large amount of reviews. This will help customers make an effective decision about buying an item at a particular cost or not.

We would also be building a prediction model to recommend a product based on the received review text, if ratings are not available, false or missing for any reason. Prediction from text will be accurate and will help customer make a decision. This will also overcome the problem of discrepancy between the sentiment of a review and the rating provided by a user.

III. Project Environment

- Python – 3.7.0
- Python libraries - scikit-learn, numpy, pandas, matplotlib etc
- Palmetto cluster
- Jupyter Notebook

IV. Dataset

The data set is collected from Kaggle and consists of 34,660 customer reviews for Amazon products like Kindle, Fire TV Stick and more provided by Datafiniti's Product Database. The dataset majorly includes basic product information, review text, ratings and user information for each product. It includes recent reviews from the year 2017-2018.

V.

Project milestones Tasks	Date
Data analysis and cleaning	6th Feb
Fake review detection and NPS Score	13th Feb
Sentiment analysis for one product	20th Feb

Try different algorithms to achieve good accuracy	5th March
Sentiment analysis comparison with different products	12th March
Recommendation prediction	19th March
Different algorithms to achieve accuracy in prediction	2nd April
Review and Final presentation	10th April

VI. Steps:

1. Data Analysis
2. NPS Score
3. Fake review detection
4. Sentiment analysis with algorithms – Naive Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Logistic regression and LSTM.
5. Recommendation Prediction - Random Forest Classifier

VII. Implementation

1. Data Analysis

The data set consists of 34,660 customer reviews for Amazon products. Dataset has 21 columns that has information about username, product, category, review, rating, source_url, review_date etc.

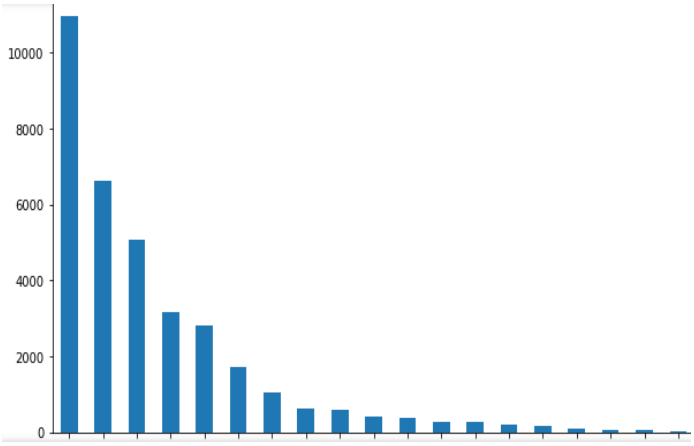
In order to implement an algorithm on data we first did analysis on the dataset to understand it. Our analysis includes following questions,

- How many different products are there?

  Dataset includes different products and their reviews so to see how many products there are we counted the number of unique product and count of reviews for each product. Result analysis is in following format, for an example: Electronics, iPad & Tables – 9000

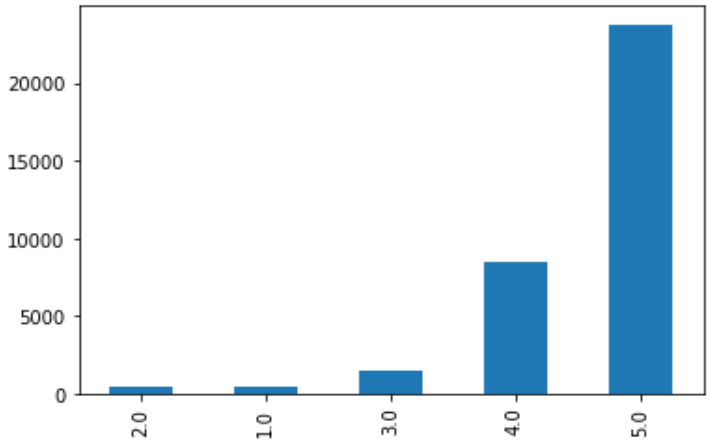
- How many reviews per each product?

  To find an answer we counted the reviews for each product and generated a bar graph to represent it. Here, x-axis represents product and y-axis represents number of reviews. For products like Batteries only 10 reviews are available and for products like Electronic Tablet more than 10000 reviews are present.



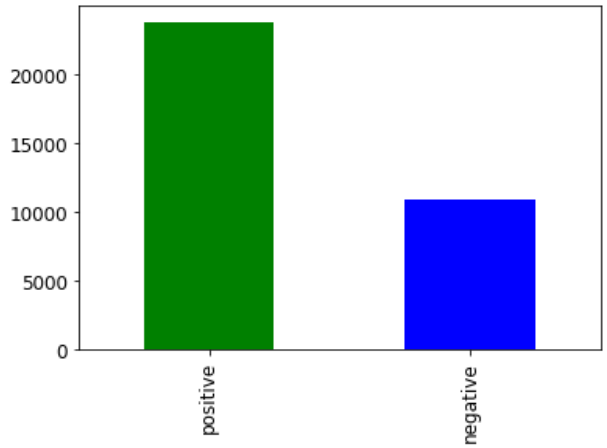
- How many users per rating?

  To find an answer we counted the number of users in each rating category (in our case 5 categories are there from 1.0 to 5.0) and drew a bar plot as a result. Here, x-axis represent ratings and y-axis represent number of users.



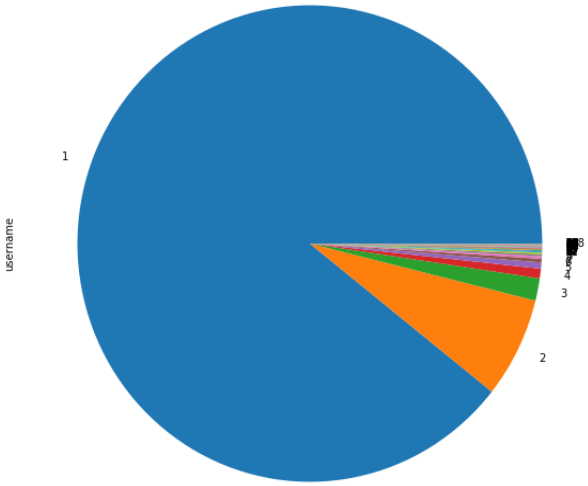
- Positive and negative ratings comparison.

  Based on the ratings given by the user, we divided it into two categories: if rating is greater than and equal to 4.0 then it is a positive review else it is negative review. Here, x-axis represents sentiment and y-axis represents number of reviews.



- How many reviews given by each user?

  Sometimes some users try to give more than one review for product promotion or detraction, these reviews are fake. To figure out if there is any fake review present in the data, we drew a pie chart as following:



From the pie chart it is visible that most of the users gave only 1 review but there are some users who gave more than 10 reviews which is not acceptable. After doing detailed analysis we decided to identify fake users and remove their reviews.

- Generated word cloud based on words in dataset.

  After cleaning the data for the algorithm implementation, we generated a word cloud as shown below

2. NPS Score

- NPS Score is Net Promoters Score, that help companies to evaluate customer satisfaction and loyalty. NPS Score can be found using below equation,

  NPS = (Promoters - Detractors)/Total ratings * 100
  Where,
  Promoter = Count of rating 4.0 and 5.0
  Detractors= Count of rating less than 4.0

- We implemented the above formula and found that the NPS score for amazon is 86.76.

3. Fake Review Detection

Sometimes some users try to give more than one review for product promotion or detraction, by analysing the data we found that there are some users who gave more than 10 reviews for a single product. This kind of misleading information can affect analysis. Therefore, we removed such users, for a single user ideally only one review should be allowed for one product. Sometimes the given space is not enough or owing to some other reason's user might write more than one review for a product. So, we allowed minimum of 3 reviews per product by a single user. We had 34660 reviews initially after removing fake reviews we were left with 28660 reviews, so approximately there was 6000 fake reviews in the dataset.

As a result, we implemented Naive Bayes algorithm on both the data with and without fake reviews, and we can see 5% of increase in accuracy.

- Achieved accuracy (Fake reviews not removed): 58.97%
- Achieved accuracy (Fake reviews removed): 64.91%

4. Sentiment Analysis

Naïve Bayes:

It is an effective algorithm that can be used for classification. In our project we are using Naïve Bayes to classify the reviews as positive or negative. Naïve Bayes is an extension of the Bayes theorem.

$$P(y \mid x_1, \cdots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \cdots, x_n)}$$

Source: Scikit-learn.org [1]

The above Naïve Bayes formula denotes "the probability that classification y is correct given the features, x1, x2 and so on equals the probability of y times the product of each x feature given y, divided by the probability of the x features". To find the "right" classification, we just find out which classification has the highest probability with the formula. Naïve Bayes algorithm assumes that all features are independent.

In our implementation we are first cleaning the reviews by converting all words to lower case, remove all spaces and generate words by stripping the sentences based on whitespaces in the sentences. Each review is initially assigned a sentiment based on the review ratings (>=4 "positive", else "negative"). We calculate the probability of each word occurring in a negative review and the probability of each word occurring in the positive review. This will allow us to eventually compute the probabilities of a new review belonging to each class. If maximum words in the review occur mostly in positive reviews the probability of review being positive would be greater and vice versa.

- Achieved accuracy (Fake reviews not removed): 58.97%
- Achieved accuracy (Fake reviews removed): 64.88%

However, when words that were not in training data occurs while using the model for predicting the sentiment of the review, the probability for both sentiments would result in being 0, commonly called "Zero Probability Problem".To address this problem Laplace smoothening is used. We add 1 to every count so it's never zero. To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1.

Also, when considering each word individually it could result in different sentiment than taking the words grouped together, this can be solved by using n-gram. In our implementation we used 2-gram that is using two words grouped together as 3-gram and higher gave similar accuracy as 2-gram:

- Achieved accuracy (Fake reviews not removed): 53.27%
- Achieved accuracy (Fake reviews removed): 55.4%

Multinomial Naïve Bayes:

It estimates the conditional probability of a word given a class as the relative frequency of term x in documents belonging to class c

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

Source: Wikipedia [2]

This variation thus considers the number of occurrences of term x in training documents from class c, including multiple occurrences.

In our model we are using Count Vectorizer to tokenize a collection of reviews and TfidfTransformer to transform the count matrix to a normalized tf or tf-idf representation. Multinomial Naïve Bayes algorithm is then applied to these tf-idf weights.

- Achieved accuracy (Fake reviews not removed): 93.3%
- Achieved accuracy (Fake reviews removed): 92.95%

Sklearn library used to implement Multinomial Naïve Bayes takes alpha value as 1 (i.e. Laplace smoothening adds 1 to the count of each word). Using ngram_range= (1,3) gives the following accuracy:

- Achieved accuracy (Fake reviews not removed): 93.29%
- Achieved accuracy (Fake reviews removed): 93%

Bernoulli Naïve Bayes:

It has a Boolean indicator about each term of the vocabulary equal to 1 if the term belongs to the examining document and 0 if it does not. This model differs significantly from Multinomial not only because it does not take into consideration the number of occurrences of each word, but also because it considers the non-occurring terms within the document. While in Multinomial model the non-occurring terms are completely ignored, in Bernoulli model they are factored when computing the conditional probabilities and thus the absence of

terms is considered. The equation is given by:

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

Source: Wikipedia [2]

In our model, as done while implementing multinomial naïve Bayes we are using Count Vectorizer to tokenize a collection of reviews and TfidfTransformer to transform the count matrix to a normalized tf or tf-idf representation. Bernoulli Naïve Bayes algorithm is then applied to these tf-idf weights.

- Achieved accuracy (Fake reviews not removed): 92.04%
- Achieved accuracy (Fake reviews removed): 91.78%

Using ngram_range= (1,3) gives the following accuracy:

- Achieved accuracy (Fake reviews not removed): 92.25%
- Achieved accuracy (Fake reviews removed): 92.3%

Logistic Regression:

It is used to model a binary dependent variable (in our case positive and negative review). The log-odds (logarithm of the odds) is calculated. The logistic function converts this log-odds to probability. For a given review the class that gets the highest probability is denoted as the sentiment of that review.

In our model, as done while implementing Multinomial and Bernoulli naïve Bayes we are using Count Vectorizer to tokenize a collection of reviews and TfidfTransformer to transform the count matrix to a normalized tf or tf-idf representation. Logistic regression algorithm is then applied to these tf-idf weights.

- Achieved accuracy (Fake reviews not removed): 93.68%
- Achieved accuracy (Fake reviews removed): 93.39%

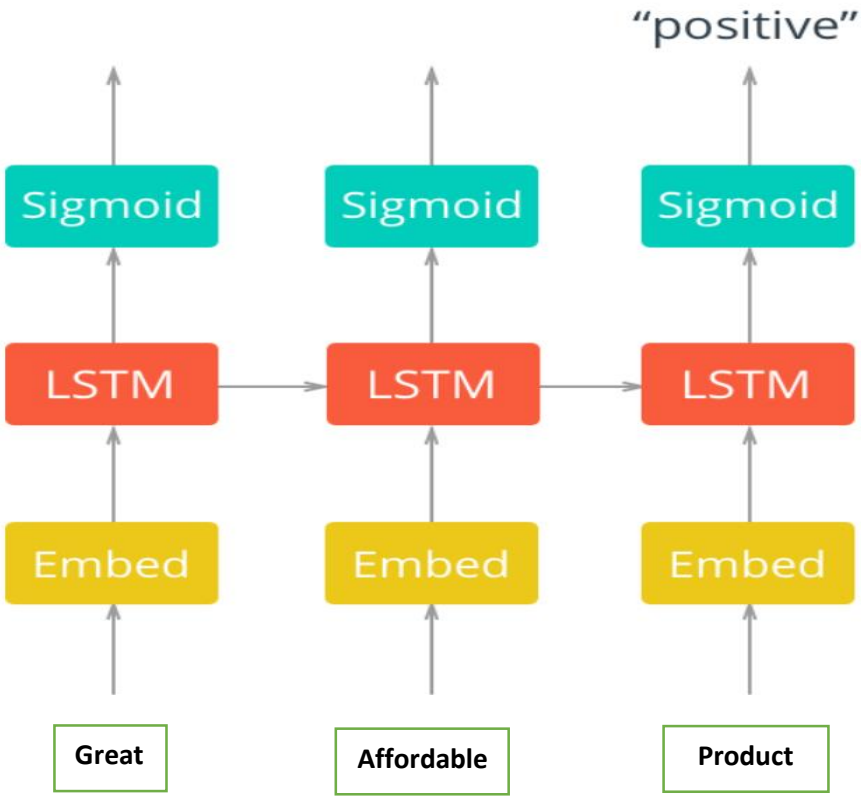
Using ngram_range= (1,3) gives the following accuracy:

- Achieved accuracy (Fake reviews not removed): 94.12%
- Achieved accuracy (Fake reviews removed): 93.8%

Neural Network (LSTM):

Tokenizer is used that turns each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf. These tokens are then passed to the embedding layer. From the embedding layer, the new representations will be passed to LSTM cells. The LSTM cells add recurrent connections to the network so we can include information about the sequence of words in the data. Finally, the LSTM cells will go to a sigmoid output layer here.

The output layer will just be a single unit then, with a sigmoid activation function that gives the prediction as positive or negative. The figure below gives the overview of the complete pipeline.



Source: TowardsDataScience [3]

The convolutional layers look like:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 1850, 150)         4500000
_________________________________________________________________
lstm (LSTM)                  (None, 200)               280800
_________________________________________________________________
dense (Dense)                (None, 5)                 1005
=================================================================
Total params: 4,781,805
Trainable params: 4,781,805
Non-trainable params: 0
_________________________________________________________________
```

After 30 epoch's we were able to achieve an accuracy of 95%.

5. Recommendation Prediction:

Sometimes a user might be very satisfied with a given product and still give a low rating and vice versa. Therefore, at times the ratings given by the user would not be an

efficient way of deciding whether a product should be bought by a user or not. Thus, we have trained a model that can predict if a product is recommended or not based on the reviews text.

We first used term normalization processes like Porter stemming and Lemmatization. The Porter stemming algorithm is a process used for removing the commoner morphological and inflexional endings from words in English. Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

We then calculated the tfid vector values for each of the terms in the reviews, by using the library vader_lexicon we calculated the sentiment score of each of the review and also the length of each of the review text was calculated. The do recommend column was used as the label, true was given a value of 1 and false 0. Thus sentiment score and length of the reviews were the features and do recommend the label.

We used random forest classifier to train the model and achieved a precision value of 0.963, recall value of 0.985 and accuracy of 0.971.

VIII. Results:

a) Sentiment Analysis:

Algorithm	Achieved Accuracy
Naïve Bayes	55.4%
Multinomial Naïve Bayes	93%
Bernoulli Naïve Bayes	92.3%
Logistic Regression	93.8%
Neural Networks (LSTM)	95%

b) Recommendation Prediction:

Evaluation Parameter	Value
Precision	0.963
Recall	0.985
Accuracy	0.971

IX. Conclusion:

We successfully analyzed the data, removed the fake reviews and implemented a prediction model to effectively predict the sentiment of a given review. Also devised a recommendation model that would either recommend or not recommend a product based solely on the review text given by the customers. However, in the future to achieve a higher accuracy more data can be used to train the models. Also, the neural network can be run for more than 30 epochs to improve it's accuracy.

X. References:

1. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
2. https://scikit-learn.org/stable/modules/naive_bayes.html#
3. https://towardsdatascience.com/sentiment-analysis-using-rnns-lstm-60871fa6aeba
4. https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products