Bartomeu Ramis Tarragó
41622730K

NLP project

In this project, our objective will be to predict if that customer is satisfied with its clothing purchases based only in their written review. In particular, we will be using data science techniques, such as machine learning and data clearing and preprocessing algorithms. Our objective model will be one based in Natural Language Processing techniques, such are those found in the Quanteda package for R.
We will be using a dataset containing about 23.000 clothing reviews collected from a real e-commerce site. The data has been anonymized, removing all identifiers from the customers, and all references to the company have been replaced with “retailer”.

The first part of this project is aimed at understanding the data and clearing any invalid values that we could find.
These are the features collected for every customer’s review:

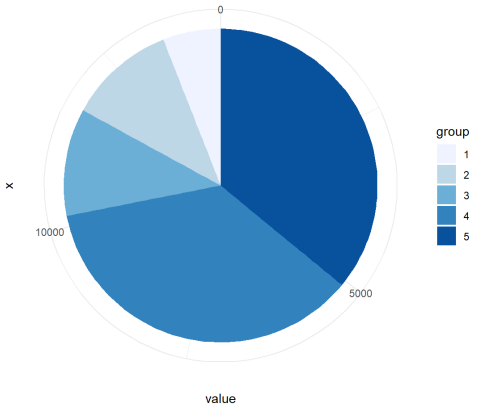
- **Clothing ID:** Integer Categorical variable that refers to the specific piece being reviewed.
- **Age:** Positive Integer variable of the reviewers age.
- **Title:** String variable for the title of the review.
- **Review Text:** String variable for the review body.
- **Rating:** Positive Ordinal Integer variable for the product score granted by the customer from 1 Worst, to 5 Best.
- **Recommended IND:** Binary variable stating where the customer recommends the product where 1 is recommended, 0 is not recommended.
- **Positive Feedback Count:** Positive Integer documenting the number of other customers who found this review positive.
- **Division Name:** Categorical name of the product high level division.
- **Department Name:** Categorical name of the product department name.
- **Class Name:** Categorical name of the product class name.

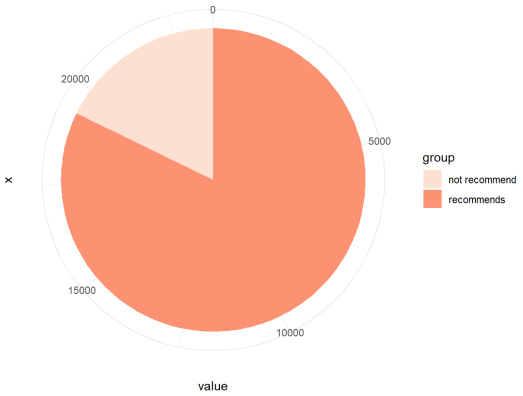
Based on our objectives, and on the techniques that will be used some of the features found will not be needed. Clothing ID could be useful if we were interested in which clothes have better opinions, but in our case, we don’t really care. Age is another piece of information that, for our purposes, doesn’t bring any useful information. Positive feedback can also be omitted, given the fact that it represents the opinion of other customers on the review, and gives no insight into the review’s expressed opinions. And finally, Division Name, Department Name and Class name won't bring any useful information for our objectives.

Rating would be extremely useful if we did not have Recommended IND. Given that we will classify a review in a binary format (recommended or not), it could be that it’s not needed, however, we will keep it, just in case.

Next, we will represent our data graphically in order to understand it better, and if it were the case, to identify possible incorrect values.

On the left blue pie chart, we see represented the rating given by the customers, while on the right red pie chart, we see the relation between the quantity of reviews that recommend and those that do not.

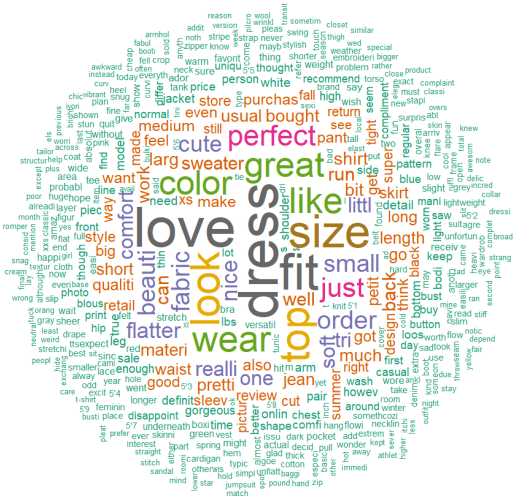




Clearly, we see that almost 3 quarters of the reviews give a positive score (equal or higher than 3) and that translates into a similar ratio of recommendation/no recommendation. This information it's quite useful, because it gives us insight into the distribution of the data. Furthermore, this clear imbalance in opinions can translate later into an overfitting problem for our model, leaning too much into assuming a review is positive when processing real opinions.

Now that we have some information on the data, we will need to process and modify it in order to process it later on.
The first step required is normalisation, this refers to the process of converting all of our data into a common scale. In our case, the data refers to the reviews text, so we will need to convert all characters into lowercase letters and also, remove any special non-letter characters (numbers, punctuation or at signs), due to not bringing new useful information to the model.
Another aspect to consider are stop words. Those are a type of word that are very common in normal speech but give very little information such as: a, is or at. These words are easily identifiable and normally, to remove unnecessary overhead, are eliminated from the original text.
Lastly, we will look into stemming. The process of stemming focuses on transforming each word into its most basic form, called root form. This will allow our model to group similar words written in different tenses, reducing complexity and, in a way, normalising all of those words that present multiple forms.

Once our text has been processed, it's split into tokens, each one containing only one normalised and stemmed word. And those tokens are organised in a document-feature matrix (also called DFM), associating each reviewer with the occurrences to each tokens.



Once we have the DFM, we can use it to generate word clouds with our data. Intuitively, words like "dress", "top" or "size" are very common due to the nature of our data. Other words like "love" or "perfect" indicate, again, that most of those reviews are positive; meanwhile, words like "return", indicating a

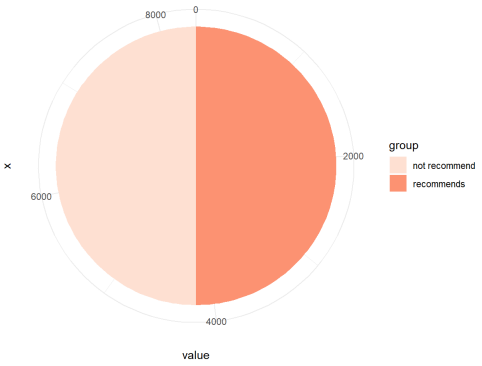
desire to return the product, are much less common.

Now, it's time to use our data to train models.
Our first attempt will use the Naive Bayes based model from our Quanteda library. One of the parameters of that model is prior. This argument refers to the prior distribution of the classes inside the text, which we really don't know. So we will test all 3 possible options for that parameter to get the best possible result.

option	termreq	uniform	docfrec
accuracy	0.8918459	0.8520332	8920588

With our accuracy score we can say that the "docfrec" option for the prior parameter works best, even if it is barely.

Before commiting to our final model, we will try to correct our biggest problem with the data: the unbalance created by the huge disparity in the amount of positive/negative reviews. Our model has too many positive reviews (about 19.000) while having too few negative ones (4.000). In order to correct this, and obtain a more balanced training set, we will sample the data set and select, randomly, as many positive reviews as available negative reviews. And then, we will merge the original 4.000 negative reviews with the 4.000 selected positive reviews to obtain a perfectly balanced set of data.



With this new data, we will repeat the preprocessing process and obtain a balanced DFM which will be used to train our final model.

With our model trained, we will apply the predict function to our test set and check the evaluation metrics of that model.

The accuracy decreased to 0.8569712. This is due to the fact that now that the data is balanced, it's quite harder to predict correctly values. However, this allows for a more "reliable" model, less "overfitted" to our unbalanced data, and more fit to work in the real world.

Reference / Prediction	Real positive values	Real negative values
Predicted posities	694	117
Predicted negatives	121	732

Our confusion matrix gives a precision of 0.875 and a recall of 0.831. While the precision refers to the quality of a positive predicted value, the recall refers to the ability of the model to correctly classify the positive inputs.

In conclusion, our model still holds an accuracy of 0.857, which is still quite high. Also, the precision and recall measures, together with the confusion matrix indicate that the model is quite balanced, meaning that it doesn't fail more at False Positives or at False Negatives, instead, it fails, more or less, the same amount in both cases.
If we wanted to go even deeper, we could test other, more advanced, machine learning models such as BERT o roBERTa, but with them, there would also come a huge increase in power and memory requirements. That would require other developing tools than those used in this project, because we already experienced some problems with the speeds of our software while training a simple Naive Bayes model.
Also, another aspect to aim for would be more data, in particular negative data. Even though the data set contains 23.000 observations, more than half of those were discarded to balance the data. So it would be a great idea to collect more negative reviews to increase the model's learning capabilities.