Cluster Analysis on Clothing Reviews Using NLP

By: Zack Godwin

The aim of this project is to group a collection of product reviews by whether or not the reviewer recommended the product. Understanding the relationship between the content of a review and the outcome of recommendation is the key to a business learning from it's customers. The better a business can understand it's customers, the better it can make informed decisions that may impact customer behaviour. Ultimately, a business can use this information to drive customers towards recommending a product, which generates new revenue.

The Corpus

This text corpus comes from an online retailer specializing in women's clothing.

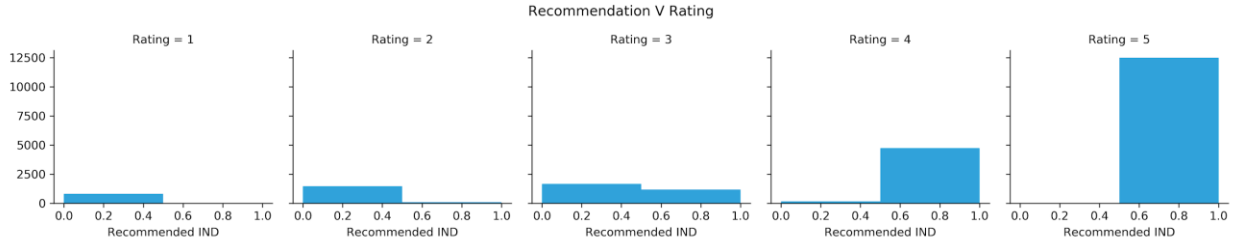
A few examples:

- "Love this dress! it's sooo pretty. i happened to find it in a store, and i'm glad i did bc i never would have ordered it online bc it's petite. i bought a petite and am 5'8"". i love the length on me- hits just a little below the knee. would definitely be a true midi on someone who is truly petite."
- "I had such high hopes for this dress and really wanted it to work for me. i initially ordered the petite small (my usual size) but i found this to be outrageously small. so small in fact that i could not zip it up! i reordered it in petite medium, which was just ok. overall, the top half was comfortable and fit nicely, but the bottom half had a very tight under layer and several somewhat cheap (net) over layers. imo, a major design flaw was the net over layer sewn directly into the zipper - it c"

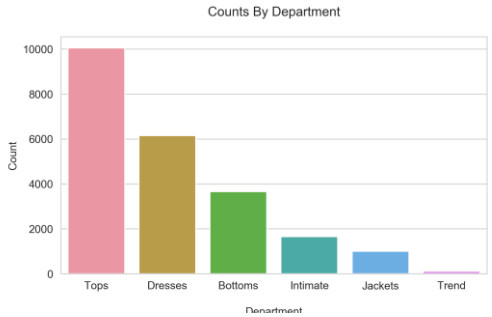
After creating some additional variables such as word and character counts, the final data frame contained 11 features, for 22628 observations. One feature contained the review text, stripped of punctuation, numbers, stopwords, special characters and whitespace. This "Filtered Review Text" served as my final corpus for cluster analysis.

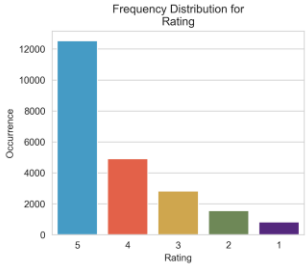
Data Visualization

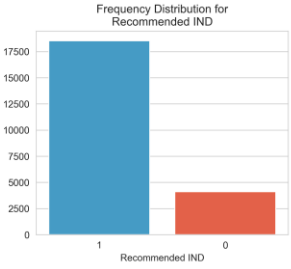
These plots investigate the relationship between the target variable "Recommended IND" and other variables.

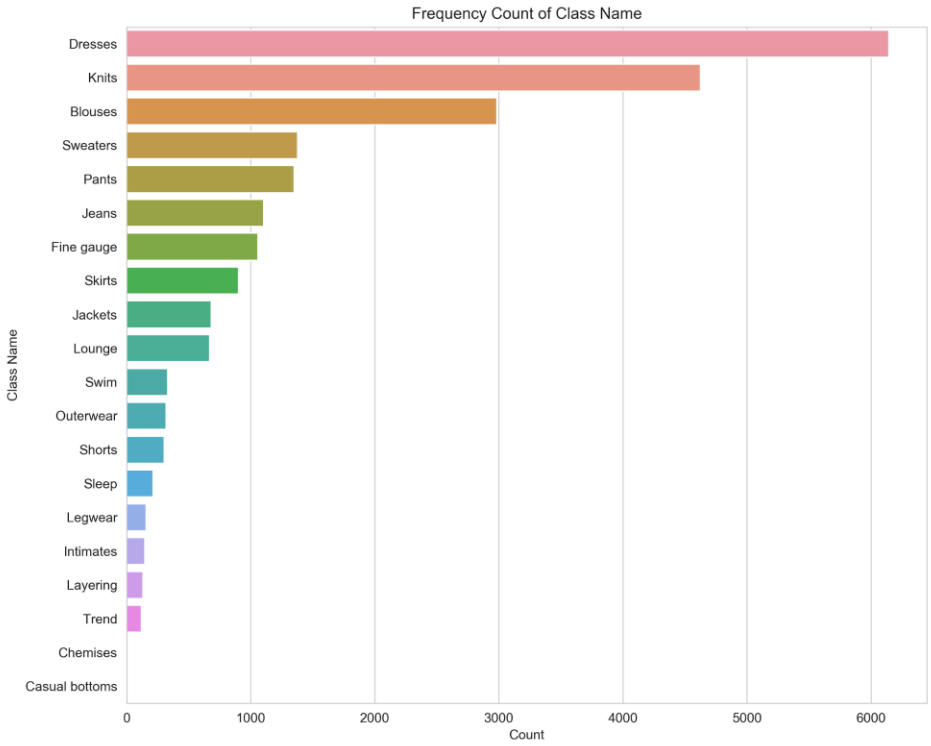


We can see that the data is imbalanced, particularly when it comes to our target variable. This is something that will be revisited later.







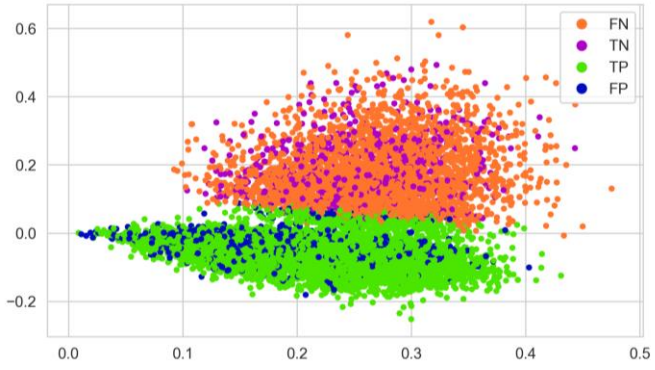


Methodology

I began with the Filtered Text column, and ran the strings through a tf-idf vectorization. This gave me weighted vectors to use for data clustering, however before doing so I had to reduce the dimensionality of the sparse array tf-idf returns. To do this I leveraged latent semantic analysis (LSA), in which I applied a truncated singular value decomposition (SVD) transformer. This performs linear dimensionality reduction similar to principal component analysis (PCA) but contrary to PCA, this estimator does not center the data before computing the singular value decomposition. As such, I included a normalizer from sklearn in my LSA pipeline. **The output was an array of 600 features that explained 68% of the variance in the data.**

K-Means

The first cluster analysis I performed was K-Means. We can see on the plot to the right that while there are 2 somewhat distinct clusters, **there was a high degree in error when using the training data to predict whether the product was recommended or not.** In fact, 32% of the data was misclassified.The f1-scores were also underwhelming at 0.21 and 0.80 for not recommended



and recommended respectively. I also investigated mini-batch k-means, Birch, Mean-Shift, and Spectral clustering algorithms. They all performed considerably worse.

Error Coding:

The colors were hard coded to values using binary logic and the following equation:

mini_error = (predict_mini*2)-Y_train

This takes the predicted value (0 or 1) , multiplies it by 2 and subtracts the actual value. Therefore, “mini_error “ can only ever be -1, 0, 1 or 2.

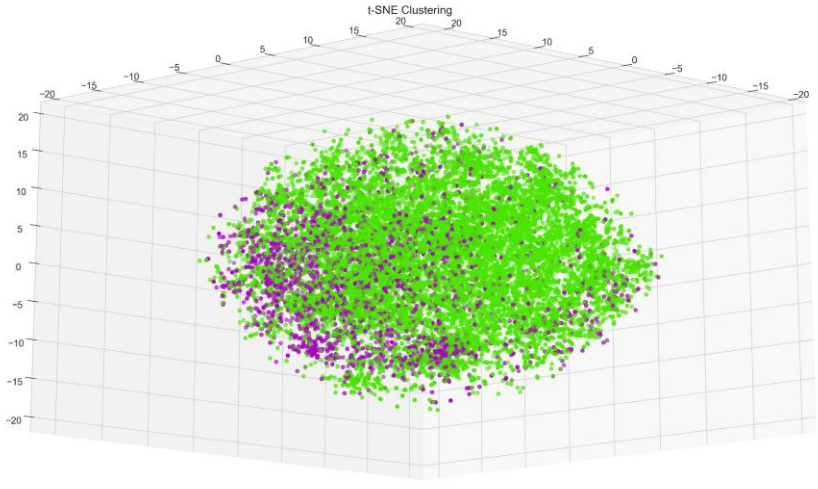
```
LABEL_COLOR_MAP = {-1: '#FF752E',
                    0:  '#ad00c8',
                    1: '#4ae500',
                    2: '#000dbf'
                    }
label_color = [LABEL_COLOR_MAP[i] for i in mini_error]
label_convert = {-1: 'FN',
                 0: 'TN',
                 1:'TP',
                 2: 'FP'
                 }
```

Using dictionaries and a list comprehension I was able to code those to each error type.

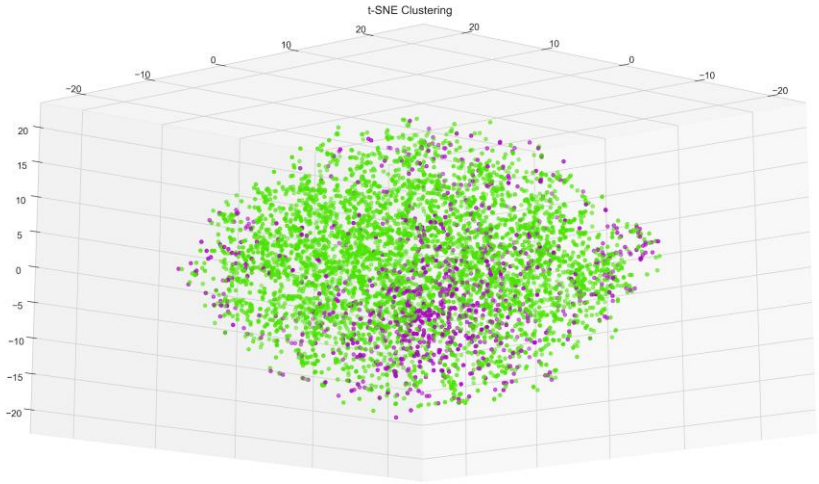
t-SNE:

K-Means tells us there are 2 clusters, but it’s unable to properly classify the true positives and true negatives all of the time. To confirm the data can cluster by recommendation we can use t-Distributed Stochastic Neighbor Embedding (t-SNE). t-SNE is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets, such as our 600 feature LSA array. I input the LSA transformed array into the t-SNE algorithm, which models each high-dimensional object by a three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. I’m also able to color code the objects by the true outcome variable, which gives us an error free cluster visualization.

We can see relative grouping of the purple and green data points. **This means that the tf-idf vectors for these reviews are similar, with a few that are still considered dissimilar.** These may explain positive reviews that do not result in a product recommendation and vice a versa.



We can cluster the testing holdout group and see similar grouping patterns. This confirms that what we observe is not limited to either subset of data. **The positioning of clusters is different, but that is due to relative nature of the plotting space, which is determined by the relative similarity of objects.**



NLP:

I chose to further preprocess my corpus, given the high error rate of the LSA output. I took the "Filtered Review Text" data and created a new feature that contained the tokenized lemmas for the list of words, and named it "Filtered Review Text Lemmas". From here I leveraged the sklearn CountVectorizer to transform my list of lemmas into count vector n-grams. Applying tf-idf, I'm able to generate weighted n-grams. These have the advantage of simplicity and scalability, which will be handy for machine learning and implementing models into production.

Supervised Learning:

For this part of the the project I trained Gaussian Naive Bayes, Random Forest, Logistic Regression, and Linear Support Vector classification models and compared performance using a 5 fold cross validation. Model performance was not ideal, likely due to the aforementioned class imbalance. I could have normalized the distribution and rerun everything, but I wanted to preserve as much of the original corpus as possible. As such, I could either downsample the majority class or upsample the minority class, and I chose the latter. To do this I leveraged the Synthetic Minority Oversampling Technique (SMOTE) from the imbalanced-learn library. **Retraining the models with this data resulted in a Random Forest Classifier with an accuracy of 0.89 and f1-scores of 0.89 and 0.90 for not recommended and recommended respectively.** Tuning hyperparameters did not meaningfully improve the model.

Summary:

We can summarize the findings from this study as follows:

- It is possible to model and group a corpus of reviews using tf-idf and the K-Means clustering, albeit with significant error.

- t-SNE is an alternative, and in this case more informative way to view the clustering of such high dimensional data.
- Through several preprocessing steps and transformations we can use customer reviews to predict whether or not that review lead to a recommendation of the product.

Limitations:

The main limitations of this study were:

- The text corpus only contained product reviews for women's clothing.
- The text corpus was skewed towards positive product ratings and recommendations.