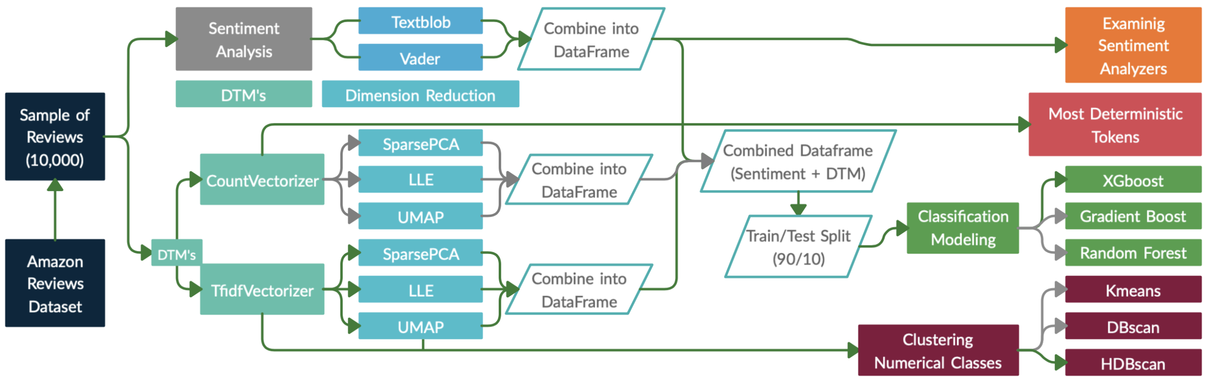
Amazon Reviews
Final Case Report
Professor Mittal
Jacob Mannix — December 17th, 2019

Method and Process



The data set given was three million Amazon reviews with their respective rating, title and description spanning over eighteen years. The flow chart above outlines the process that I went through for this analysis. To begin on the left side, I created a sample of the full data set due to computing time. I chose to take a sample of 10,000 rows. Each color indicates a different aspect of the process and green arrows indicates the preferred method over others. On the far right there are four colors indicating the four main sections which will be explaining more in detail below.

Examining Sentiment Analyzers

To examine the quality of the sentiment analyzers, *TextBlob* and *Vader* I graphed the polarity scores along with the actual ratings of the reviews in order to see how accurate they were.

Figure 1 to the right showcases rating versus polarity score for *TextBlob*. As shown in the graph it appears that on average when a review has a higher rating it has a more positive polarity score and when a review has a lower rating the polarity score is lower. That being said, the majority of the polarity scores seem to be between 0 and 0.5 with a few reaching the opposing peaks. As a whole *TextBlob* seem to be fairly conservative in the ratings and chooses to score many of the reviews slightly above or below zero.

Figure 1: TextBlob

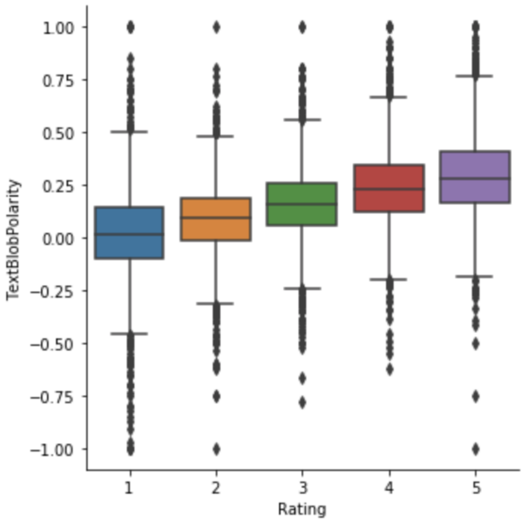
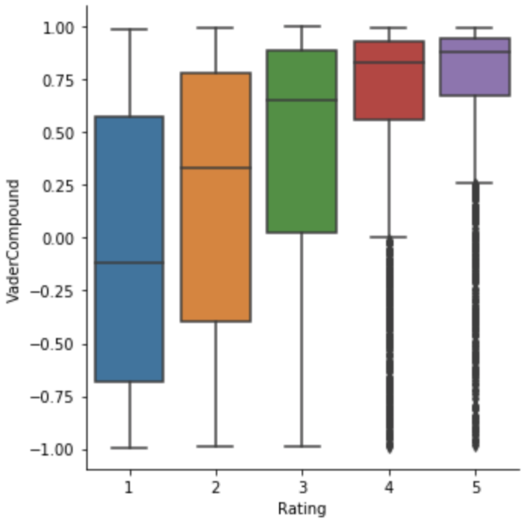


Figure 2 shown to the right displays the rating versus polarity scores for *Vader*. As shown in the graph there is a noticeable distinction between polarity scores for each rating. The majority fifty percent of the box-plot spans a large range for a rating of one but becomes much smaller as the ratings increase. I think this could be due to *Vader* not being as harsh on negative words/reviews that do not contain many negative words. This causes *Vader* to do a pretty good job of separating reviews by polarity scores with higher ratings but struggles a bit with lower ratings.

Figure 2: Vader



Overall, the two sentiment analyzers were able to partly distinguish ratings based off of their polarity scores, but they were by no means excellent. I think that between the two, *Vader* did a better job than *TextBlob* by being able to separate the difference in the ratings more being able to separate the majority of the four and five star ratings from the others.

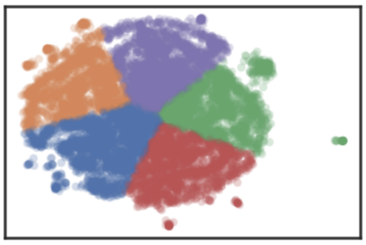
Classification Modeling

To begin modeling I created a document term matrix using *Tfidf Vectorizer* L2 parameters. I then ran for *SparsePCA*, *LLE* and *UMAP* on the document term matrix to reduce the dimensions and combined these dimension reduced columns from *SparsePCA*, *LLE* and *UMAP* together into one data frame. Next I added the polarity scores from *Textblob* and *Vader* to that data frame. After creating this combined data frame, I created a train/test split of 90/10. I chose this split because it was only a sample of 10,000 and I wanted to make sure I had a large enough set to train on. I chose to model using *XGboost*, *Gradient Boosting* and *Random Forest*. I used *GridSearchCV* to test a variety of hyper-parameters for each algorithm. After running many models, I ended with a best model *Gradient Boosting* that predicted the numerical class with a *38.9%* accuracy score. For that best accuracy score I used a *learning rate* of 0.01, 1000 *estimators* and a *sub-sample* of 0.4.

Clustering Numerical Classes

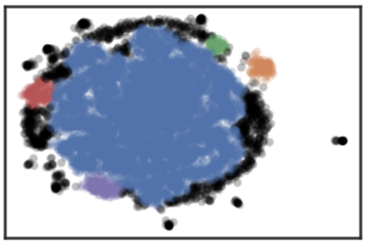
For clustering, I created a document term matrix created using *Tfidf Vectorizer* and used *UMAP* to reduce dimensions. I then clustered using *Kmeans*, *DBSCAN* and *HDBSCAN*. Beginning with *Kmeans*, shown in *Figure 3* on the right with clusters separated by color, I tried a variety of different number of cluster sizes and could not find much evidence of separation within the rating classes. As seen in the figure there are clearly clusters but the distribution of ratings within each cluster are almost equal.

Figure 3: Kmeans



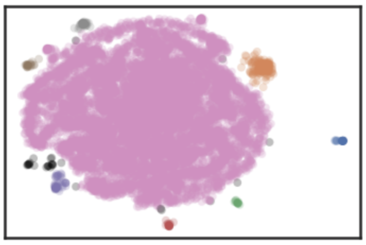
As for *DBSCAN*, there was not much evidence either. *Figure 4* shown on the right, colored by cluster, shows a large cluster in the center with smaller clusters around the edges. I tried a variety of different sample sizes and none of them seemed to be able to separate the clusters based on the rating class. Similar to *Kmeans* the clusters nearly equally distributed the rating classes.

Figure 4: DBSCAN



Lastly, *HDBSCAN* was able to separate the data into more clusters than the previous two options but these additional clusters did not offer much in revealing the rating class. As seen in *Figure 5* on the right, there is a large cluster in the center, similar to *DBSCAN* but there are many more smaller clusters around the edges. Once again these clusters were fairly equally distributed throughout all of the rating classes.

Figure 5: HDBSCAN



Overall, I do not think that clustering alone is a good method to identify the rating class. The methods specified above did not offer much of anything in trying to reveal the differences in rating classes.

Most Deterministic Tokens

In order to try and determine the most deterministic tokens for each rating I decided use the *CountVectorizer* document term matrix. I converted this to a data frame and created a pivot table of the data in order to see the percent of a specific word in each of the numerical classes for ratings. I then took the top 50 most used words for each class and compared them. There were many duplicates and I wanted to remove some to get a better understanding so I decided to remove them but did not remove duplicate words for classes that were adjacent to one another (i.e. For a rating of 2, not removing duplicates that came from rating 1 and 3). I did this because many words were used in all rating classes and I thought this would give a better look into the main differences between good and bad reviews. Also, if the top word is in all classes then it is not very deterministic of the rating if they all have it. By doing this I end up with lists of most used words for each rating class of around 5-10 words in length.

Rating	Word 1	2	3	4	5	6	7	8	9	10	11	12
Rating 1	money	tri	bad	wast	review	back	got	never	know	purchas	peopl	
Rating 2	tri	disappoint	product	two	end	seem	charact	say	got			
Rating 3	littl	still	lot	howev	seem	interest	end	charact				
Rating 4	littl	song	also	enjoy	still	album	music	recommend	best	see	give	lot
Rating 5	best	recommend	music	easi	find	new	life	mani	everi			

After looking at the different words in each of the classes it is understandable as to why many of these words are present from some rating classes and absent from others. For instance, also seen in the figure above, the words *'best'* and *'recommend'* are at the top of the list for a rating of five and the words *'bad'* and *'wast'* are near the top of the list for a rating of one. One word that surprised me was *'money'*. I would have thought *'money'* would be fairly evenly distributed through all classes, but it appeared as the top word for rating one.