Case Study 3
Textual Analysis of Movie Reviews: When Two is Better Than One and Three Is Not a Crowd

Group 1

Claire Danaher, Jonny Friedman, Janvi Kothari, Renee Sweeney, Erin Teeple

November 16, 2017

Introduction

Natural language processing merges computer science, artificial intelligence, and linguistic analysis methods to develop algorithmic approaches for automated computer interpretation of natural language text [1]. Application areas for natural language processing methods include automated document sentiment classification (as in this case study), as well as a number of other complex, time-intensive tasks such as language recognition, responding to customer service requests, and text content monitoring.

In Case Study 3, we applied scikit-learn (SKlearn) machine learning algorithms for natural language processing to classify movie reviews as either positive or negative. Automated classification of movie reviews is an interesting and important problem. The movie reviews analyzed as part of this project are free-form text documents. These are written in a range of styles by different authors expressing individual opinions using unique word combinations. The content of these reviews also includes varying proportions of content summaries, comments on reviewer expectations, comments on artistic choices, and personal reactions. While some words in the reviews have context-independent positive or negative connotations, understanding the overall opinion of a particular reviewer for a particular movie is a more complex task, which requires the capacity to achieve a broader recognition of patterned language use across sentences, paragraphs, and the text document as a whole.

The movie review data set used for this analysis included a collection of movie reviews classified previously as either positive or negative. Our team was tasked with identifying a machine learning model that could provide the highest prediction accuracy. Since the reviews had already been labelled as either positive or negative, supervised machine learning approaches were used for this investigation to explore how accurately text processing could predict movie review classifications. We then applied data visualization techniques to explore and illuminate signals detected within our sample.

[1] https://en.wikipedia.org/wiki/Natural_language_processing

Methods

1. Data Collected

Movie reviews used for this analysis were downloaded from the v2.0 polarity dataset located as http://www.cs.cornell.edu/people/pabo/movie-review-data. This data set contains 2000 written reviews of movies that have been divided into separate folders of labelled positive and negative reviews. The data was split into testing and training data with the training data comprising 75% of the data and 25% of the data reserved for testing.

2. Exploring the TfidVectorizer Class

The TfidVectorizer class provides a method for transforming a set of text documents into a TF-IDF feature matrix. TFidVectorizer was used to construct of a classifier pipeline that filtered words ("tokens") that were too common or infrequent in our text documents using TfidVectorizer[2].

Natural language data presents unique challenges for data scientists. The term frequency- inverse document frequency(TF-IDF) statistic is a quantitative metric that was developed in order to attempt to address one of the many idiosyncrasies of natural language data. TF-IDF is a weighting system that uses the frequency of words as a proxy for importance. The algorithm is designed to adjust for removal of low frequency words as well as for high frequency words. The removal of high frequency words is important because these provide little useful information for differentiation purposes as they are so commonly present.

The parameters used to calculate TF-IDF are listed below:

· *max_df* - This parameter is used to exclude the terms from our vocabulary that has a high document frequency; greater than a specified threshold. This parameter helps us eliminate words that occur too frequently in our documents i.e. corpus-specific stop words; thus, making them non-relevant.

---

[2] (http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)

· *min_df* - This parameter is used to exclude the terms from our vocabulary that has a very low document frequency; lower than a specified threshold. This parameter helps us eliminate words that occur seldom in our documents; thus, making them non-relevant.

· *N-gram range* - This range is used to extract terms from our documents as a set of words. That is if n=1 it gives terms only as a set of 1 word. These are called the unigrams. If for instance, n=2 it gives terms only as a set of 2 words. These are called as bigrams. In such a way, we can get terms from our documents as set of range i.e. min_n and max_n and thus returns terms from the dictionary as set of minimum words (min_n) in a set to maximum words (max_n). This forms the n-gram range.

3. Fitting and Optimizing Parameters for the TFidVectorizer and Machine Learning Algorithms

GridsearchCV[3] was used to tune the parameters for the TF-IDF. This function allows for the passing of parameters which are then cross-validated. Using this approach, our team was able to identify the tuning parameters which maximized the test rate.

A review of a number of models was used to identify the model which returned the highest accuracy rate for determining movie review positivity or negativity. K-Nearest Neighbors and Linear Support Vector Classifier models were the preliminary models that were used. For each of these approaches, GridSearchCV[4] was used to identify optimal parameters for our classification models.

4. Visualization of Review Classifications

The final objective for the case study was to attempt to identify a two-dimensional plot in which positive and negative reviews were separated.Our group tried a number of dimension reduction and visualization techniques: PCA, Linear Discriminant Analysis, Locally Linear Embedding (LLE), Local Tangent Space Alignment (LTSA), Hessian Eigenmapping, Modified Locally Linear Embedding, Isomap, Multi-dimensional Scaling (MDS), Spectral Embedding, Random Forest, Gradient Boosting, and Adaptive Boosting.

---

[3] (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

[4] (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

Results

1. Basic Model

A basic model for sentiment analysis is provided as part of the documentation for SKlearn. The example code uses cross validation to tune and determine whether unigrams or bigrams perform better when analyzed using a support vector machine (Figure 1). Bigrams were found to perform slightly better.

Figure 1. Classification report and confusion matrix showing bigrams provide better predictive accuracy than unigrams in the example provided.

```
n_samples: 2000
0 params - {'vect__ngram_range': (1, 1)}; mean - 0.84; std - 0.01
1 params - {'vect__ngram_range': (1, 2)}; mean - 0.85; std - 0.01
             precision    recall  f1-score   support

        neg       0.88      0.85      0.86       268
        pos       0.83      0.86      0.85       232

avg / total       0.85      0.85      0.85       500

[[227  41]
 [ 32 200]]
```

---

2. TfidVectorizer- Tuning Parameters

Further tuning was then conducted on the TFidVectorizer parameters. Tuning the parameters required significant computing resources and was time-intensive.

Exploring the min_df and max_df parameters of the TfidVectorizer altered the words included by defining the range of included frequencies for our features. Exploring the ngram_range parameter changed the number of grouped words included as features (for example, a unigram consists of one word, a bigram includes two words, etc.). Including grouping of words as well as individual words has the potential to increase classification accuracy but dramatically increases computation time required for model building and classification.

Our team explored a number of tuning parameters for min_df and max_df at differing levels of granularity. We first ran the data set with a smaller number of test parameters. The parameters were as follows:

```
#SMALL SET
'vect__min_df':[1,2,3,4,5,6,10,15,20],
'vect__max_df':[.85,.88,.90,.92,.94,.96],
```

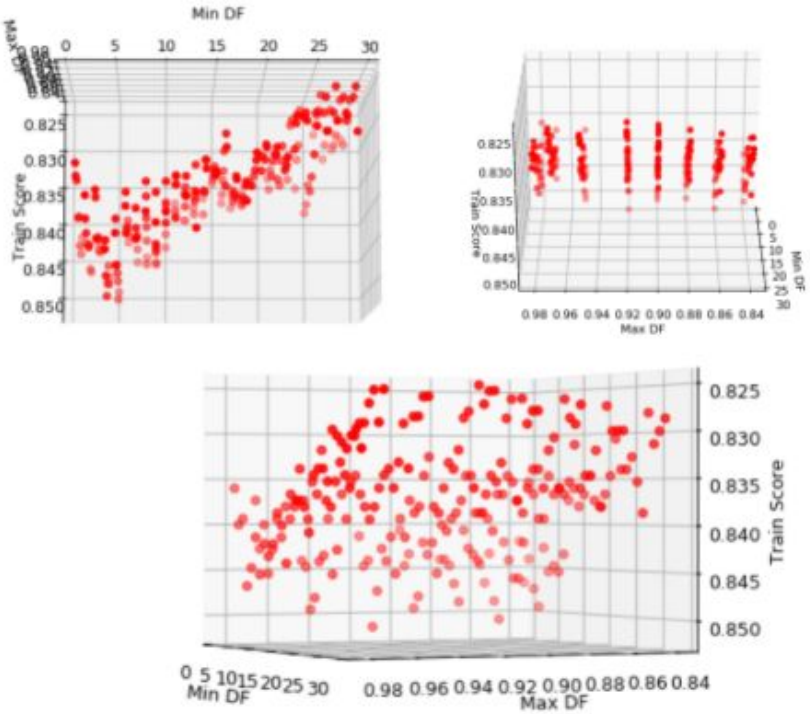
The figure below shows the processing time, confusion matrix and optimal values using this approach.

```
Fitting 3 folds for each of 54 candidates, totalling 162 fits
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   36.9s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  6.0min
[Parallel(n_jobs=-1)]: Done 162 out of 162 | elapsed:  6.3min finished
             precision    recall  f1-score   support

        neg       0.89      0.84      0.86       262
        pos       0.83      0.89      0.86       238

avg / total       0.86      0.86      0.86       500
```

```
print(grid_search.best_params_)
```

```
{'vect__max_df': 0.88, 'vect__min_df': 6, 'vect__ngram_range': (1, 2)}
```

A few plots were created which explored the relationship between the training accuracy rate, min_DF and max_DF.



Our team then reran the cross validation using a more granular set of parameters as outlined in the figure below.

```
#LARGE SET
'vect__min_df':range(1,30),
'vect__max_df':np.arange(0.85,0.9,0.005),
```

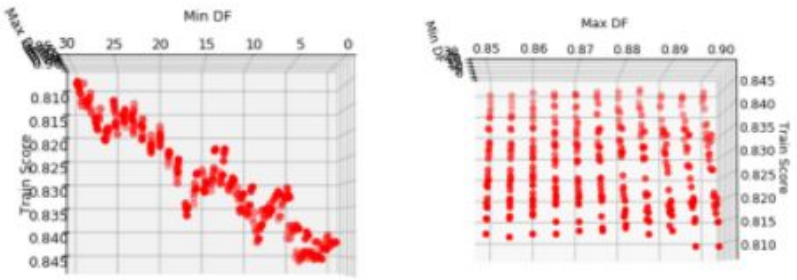
The results of this run are provided in the figure below.

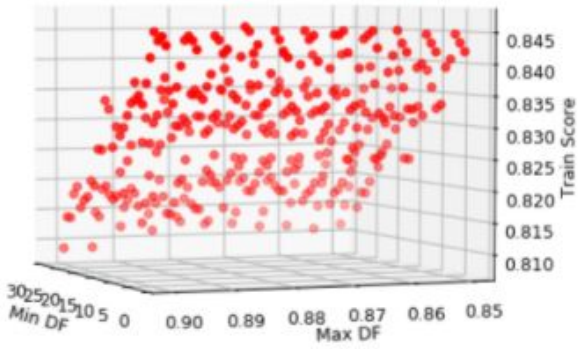
```
Fitting 3 folds for each of 319 candidates, totalling 957 fits

[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   41.7s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  5.5min
[Parallel(n_jobs=-1)]: Done 280 tasks      | elapsed: 10.0min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed: 15.7min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed: 22.8min
[Parallel(n_jobs=-1)]: Done 874 tasks      | elapsed: 31.1min
[Parallel(n_jobs=-1)]: Done 957 out of 957 | elapsed: 33.9min finished

             precision    recall  f1-score   support

        neg       0.88      0.85      0.86       241
        pos       0.86      0.89      0.87       259

avg / total       0.87      0.87      0.87       500
```

```
print(grid_search.best_params_)
```

```
{'vect__max_df': 0.88500000000000001, 'vect__min_df': 4, 'vect__ngram_range': (1, 2)}
```

The plots were then recreated which explore the relationship between the training accuracy rate, min_DF and max_DF.





We also explored the ngam feature for qualitative purposes. As part of question one we determined that bigrams produced the best results.Keeping min_df = 0.01, max_df = 0.95 constant, we vary the n-gram range and get the results as follows:

28	1998	3.895046
29	1999	3.983153
...	...	...
4526	worthwhile	4.946591
4527	worthy	4.155004
4528	wouldn	3.344074
4529	wound	5.541298
4530	wounds	5.541298
4531	wow	5.135833
4532	wrapped	5.095011
4533	wreck	5.178393
4534	wrestling	5.541298
4535	wright	5.541298
4536	write	3.944439
4537	writer	2.995767
4538	writers	3.919438
4539	writes	4.912689
4540	writing	3.423538
4541	written	2.711768
4542	wrong	2.875808
4543	wrote	3.760010
4544	yeah	4.270836
4545	year	2.187018
4546	years	2.180489
4547	yelling	5.541298

N-gram range 1, 1 – 4556 features

29	1997	4.037221
...	...	...
5139	writing	3.493605
5140	written	2.734157
5141	written directed	5.480673
5142	wrong	2.841616
5143	wrote	3.708717
5144	yeah	4.219542
5145	year	2.180489
5146	year old	3.534763
5147	years	2.180489
5148	years ago	4.139500
5149	years earlier	5.541298
5150	years later	4.422067
5151	years old	5.135833
5152	yelling	5.541298
5153	yes	3.207941
5154	york	3.469700
5155	york city	4.650325
5156	young	2.365852
5157	young boy	5.369448
5158	young girl	5.480673
5159	young man	4.625007
5160	young woman	4.787526
5161	younger	4.094379

N-gram range 1, 2 – 5169 features

28	1996	4.507224
29	1997	3.996399
...	...	...
5127	writer	2.990877
5128	writer director	3.919438
5129	writers	3.970081
5130	writes	4.981682
5131	writing	3.477605
5132	written	2.796434
5133	written directed	5.269364
5134	wrong	2.845827
5135	wrote	3.883070
5136	yeah	4.362643
5137	year	2.180489
5138	year old	3.431085
5139	years	2.191394
5140	years ago	3.907168
5141	years later	4.343595
5142	years old	5.135833
5143	yelling	5.423515
5144	yes	3.251292
5145	york	3.518096
5146	york city	4.702969
5147	young	2.350307
5148	young boy	5.178393
5149	young girl	5.222844

N-gram range 1, 3 - 5157 features

600	watching film	4.817379
601	watching movie	4.817379
602	way film	5.269364
603	way movie	5.423515
604	wes craven	5.423515
605	witch project	5.369448
606	woman named	5.480673
607	woody allen	5.095011
608	woody harrelson	5.423515
609	work film	5.480673
610	world war	5.095011
611	world war ii	5.541298
612	worst film	5.423515
613	worst movie	5.318155
614	worth seeing	4.879900
615	worth watching	5.269364
616	writer director	3.919438
617	written directed	5.423515
618	year old	3.446352
619	years ago	3.983153
620	years later	4.306554
621	years old	5.135833
622	york city	4.817379
623	young boy	5.318155
624	young girl	5.369448
625	young man	4.600315

N-gram range 2,3 – 627 features

3. Machine Learning Algorithms

Comparing the performance of the Support Vector Classifier with the K-Nearest Neighbors model, we observed that the optimized K-Nearest Neighbors model did not perform as well as the SVC.

Figure 2: SVC classification performance summary

```
{'clf__C': 100, 'vect__max_df': 0.885, 'vect__min_df': 4, 'vect__ngram_range': (1, 2)}
             precision    recall  f1-score   support

        neg       0.88      0.84      0.86       262
        pos       0.83      0.88      0.85       238

avg / total       0.86      0.86      0.86       500

[[219  43]
 [ 29 209]]
```

Figure 3. K-Nearest Neighbors classification performance summary.

```
{'clf__n_neighbors': 36, 'vect__max_df': 0.885, 'vect__min_df': 4, 'vect__ngram_range': (1, 2)}
             precision    recall  f1-score   support

        neg       0.82      0.56      0.67       262
        pos       0.64      0.87      0.74       238

avg / total       0.74      0.71      0.70       500

[[147 115]
 [ 32 206]]
```

Looking at two incorrectly predicted examples, we considered occurrences of misclassification for the SVC using bigrams with constraint parameter (c) = 100. The two misclassified documents we examined were the following:

Y_predicted[11] - should be 1, classified as 0: This is a positive review of *The Hunt for Red October*, but the reviewer spends a fair amount of time discussing the type of people who might be disappointed by this movie adaptation of the popular book. The use of negative language, including words and phrases such as “disappointed” and “never live up to” may have resulted in misclassification in this case, despite the author ending the review with a positive assessment of the film itself.

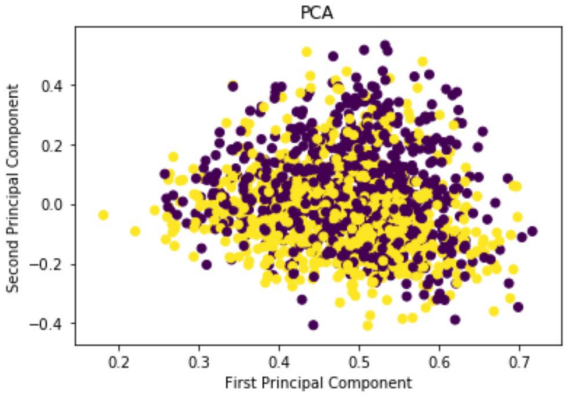
Y_predicted[484] - should be 0, class as 1
This is a negative review of the film *Session 9* that is misclassified as positive. In this case, the reviewer spends much of the review describing the movie’s many dramatic

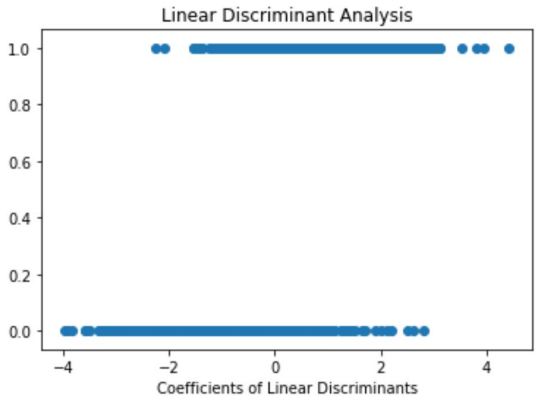
special effects, while the amount of the review that includes language that explicitly states the reviewer's low opinion of the film is rather brief.

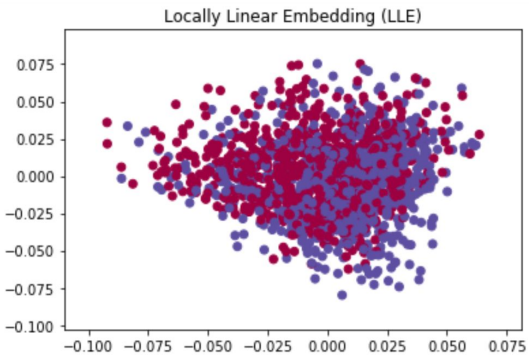
In both of these cases, the reviews were misclassified when portions of the text were inconsistent with the overall conclusion of the review.

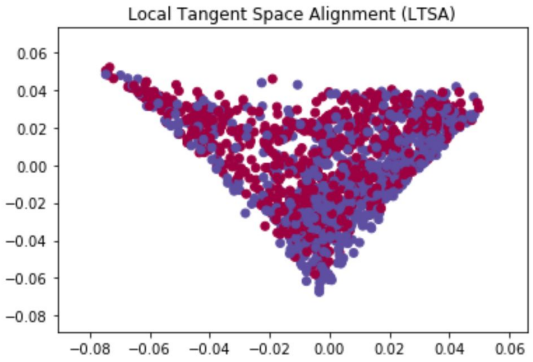
4. Visualization of Review Classifications

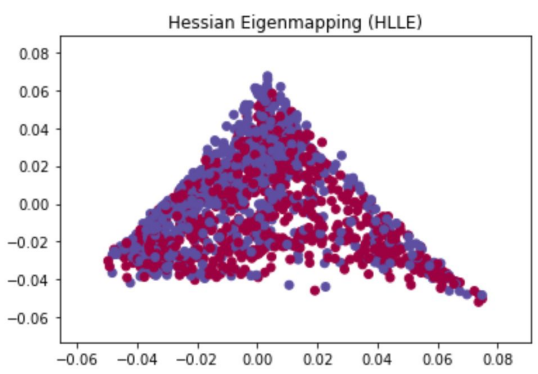
The final objective for the case study was to attempt to identify a two-dimensional plot in which positive and negative reviews were separated. Two-dimensional plots for each method tried are shown below. One technique that we wanted to try was an autoencoder, but scikit learn did not have deep learning capabilities, so that is a possibility for future work.

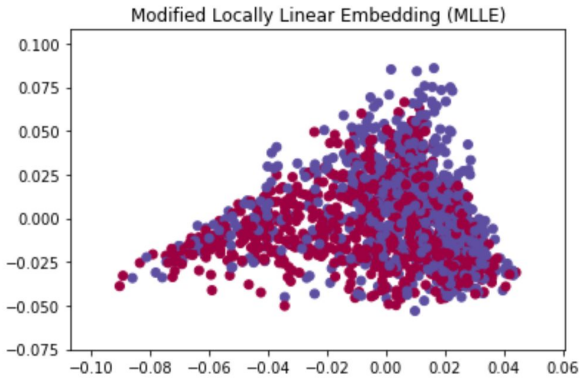


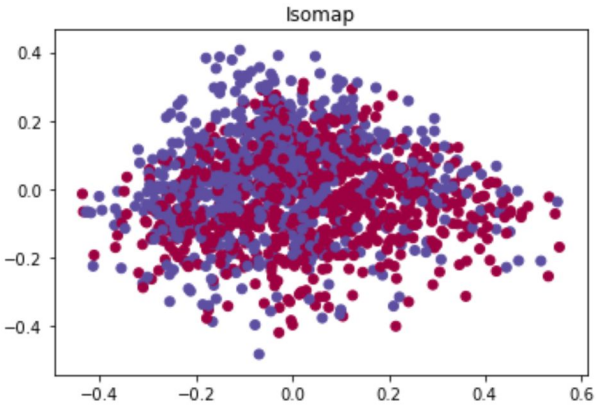


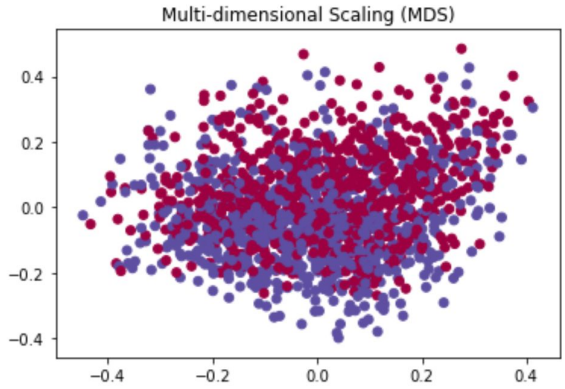


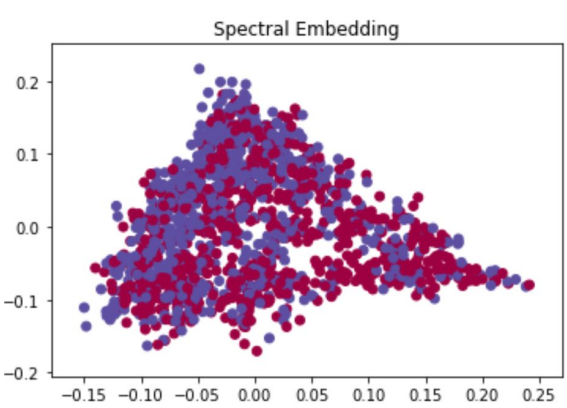


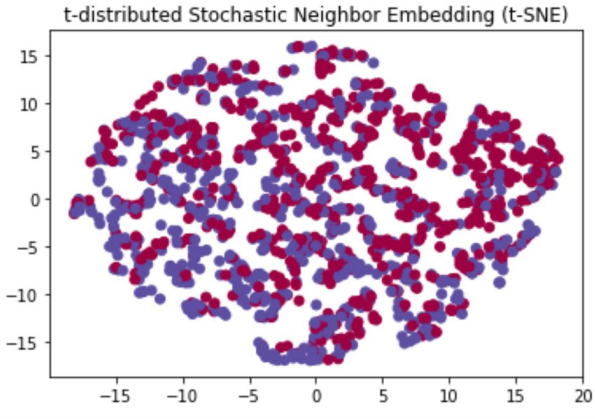

Hessian Eigenmapping (HLLE)

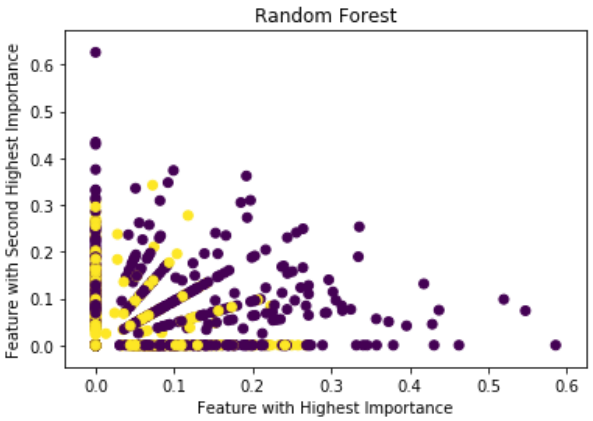

Modified Locally Linear Embedding (MLLE)

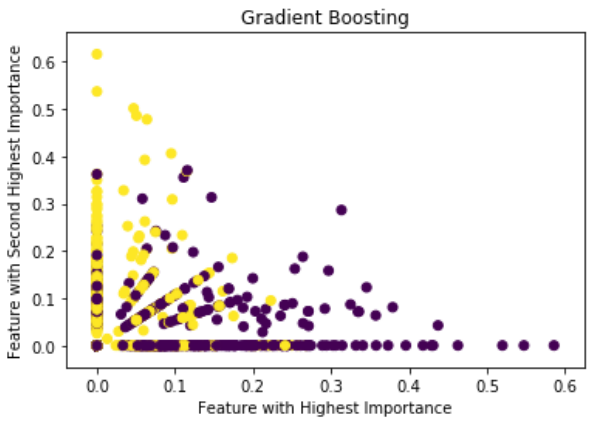

Isomap

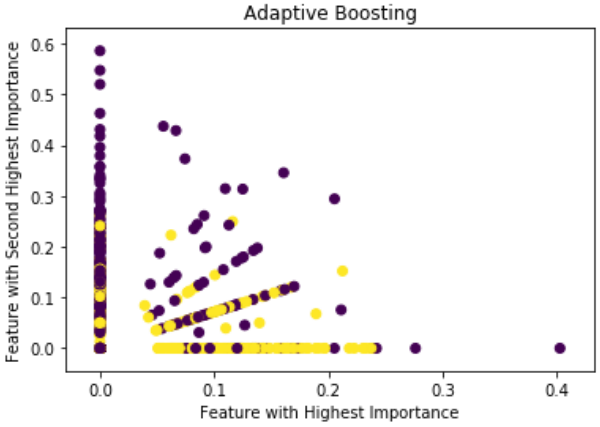

Multi-dimensional Scaling (MDS)


Spectral Embedding


t-distributed Stochastic Neighbor Embedding (t-SNE)







Discussion

In this case study, we were able to achieve relatively high classification accuracy for the task of determining movie review positive or negative sentiment using machine learning methods for text processing. Visualization of our insights in a 2-dimensional figure proved challenging, although some separation could be demonstrated with different methods.

Detecting opinions and sentiments contained within free text writing has a number of important business applications. Specific to the movie industry, using natural language processing algorithms to process movie reviews provides a time-efficient and standardized approach that yields a quantitative assessment of critical reception that can be compared directly between different films. Average response sentiment for movies can then be explored relative to cost of production, featured talent, genre, or other movie characteristics of interest to provide useful insights and to contrast these

different comparisons with movie profitability. By transforming each of these dimensions into standardized measures, this allows for a multidimensional examination of the overlapping relationships between critical acclaim, profitability, genre, and cost, among other business metrics. Improved quantitative models can then be used to guide future investments and project selection to maximize profitability.

Similarly, at a broader level, the ability to accurately identify text as expressing positive or negative sentiment relative to a specific product has many useful applications across multiple industries. For example, using technology of this type, messages sent to a customer service email address could be filtered and differentially assigned for follow-up protocols specific to customer complaints versus satisfied customer contacts. Or product reviews posted online could be be assessed to determine the ratio of positive to negative reviews that have been posted. In all cases, the ability to automate text processing and achieve accurate predictions reduces the amount of human labor required to perform this task and results in a more standardized approach.

Conjectures we made during our data processing involved an exploration of the specific content of documents that were misclassified. In both of the misclassified reviews examined, the content of the reviews included stretches of text that were inconsistent with the final review content - one included a description of people who might not like a movie the reviewer him/herself had found enjoyable and the other included a flattering listing of impressive special effects in a movie that the reviewer did not like overall. Both of these misclassifications highlight an expected limitation of the text processing methods used. These were instances where the content itself was inconsistent with the overall classification. Such exceptions would be expected to impact algorithm performance and highlight the complexity of the task of interpreting nuanced natural language text.

One thing that was surprising was the degree of accuracy in classification that could be achieved in this data set. In order to determine whether this level of accuracy is possible for other text document forms, further studies using additional data sets are needed for cross-validation.