

Sentiment Analysis and Key Phrase Extraction

AIDI 1002 - Final Project

Michael Molnar

100806823

Business Problem Statement

Business leaders and marketing departments must constantly answer difficult questions.

- How do customers feel about our business' products and brand?
- Are our customers satisfied with our service?
- How do our company's policies impact on how our customers see our brand?
- What do consumers like about our competitors?



Sentiment Analysis for Businesses

Today, an estimated 90% of data is unstructured, and Machine Learning can be leveraged to accurately and automatically answer these business questions. A Sentiment Analysis tool provides businesses with:

- The ability to make better decisions by understanding how their brand is viewed in real time
- The ability to track how sentiment changes after a change in direction or the launch of a new product
- The ability to prioritize unhappy customers and improve customer service

Having these insights will allow a business to boost sales and improve their products and customer service. An accurate sentiment analysis predictor would be an asset to any business.






Project Scope

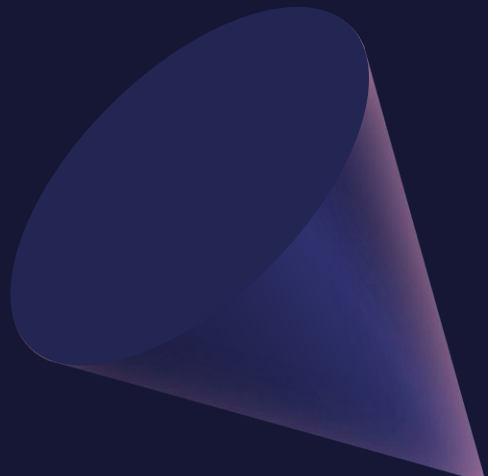
I have built a functioning and web-accessible Sentiment Analysis tool that predicts whether a review represents a positive, neutral, or negative sentiment. It is accessible at:

<https://sentiment-analyzer-mm.herokuapp.com/>

My model uses a Logistic Regression Classifier built with a Count Vectorizer that splits text into unigrams, bigrams, and trigrams. I have performed a number of data cleaning techniques to improve accuracy.



The model was built on approximately 80,000 real Amazon reviews after class distributions were equally distributed. It achieved 97.41% accuracy on the training set and 86.07% accuracy on the held out testing set.



Dataset

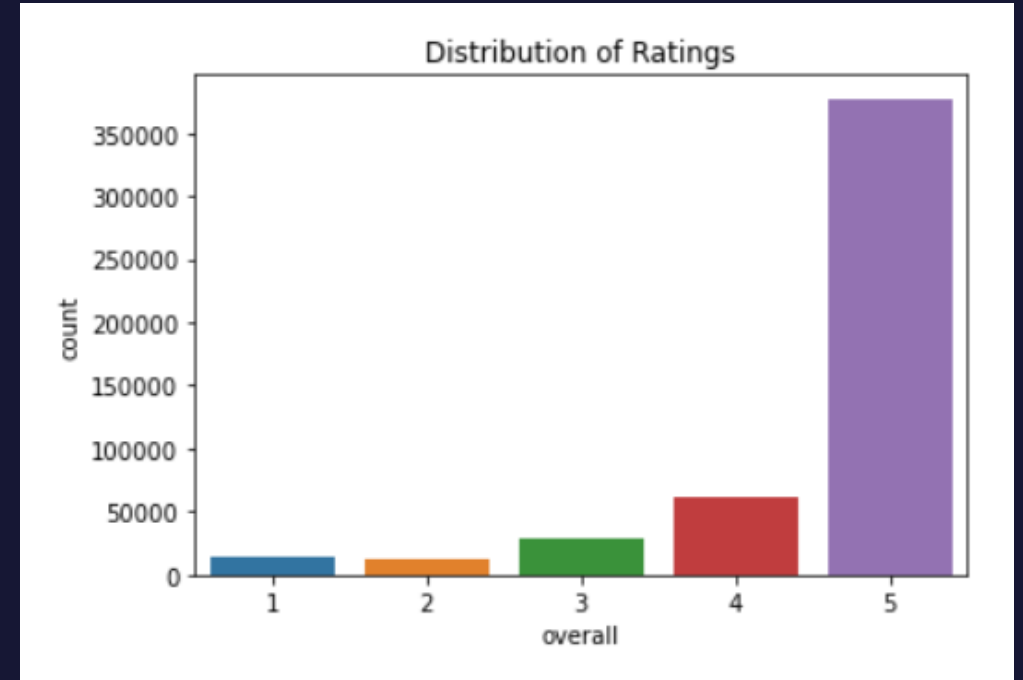
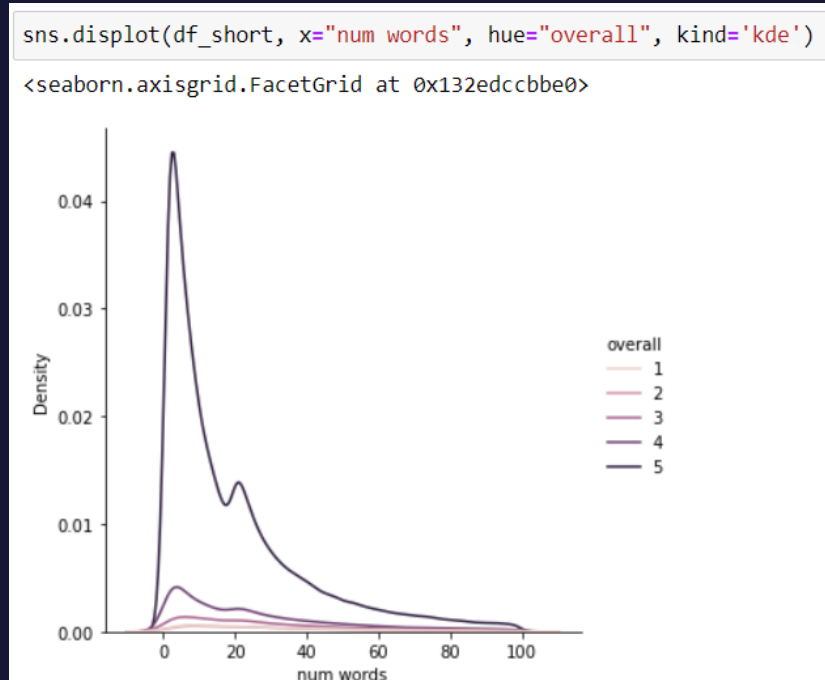
I worked with the Amazon Review Data (2018). This is a collection of 233.1 million Amazon reviews maintained by Jianmo Ni. I chose to work with the “Arts, Crafts and Swing” subset of the data, which provides 494,485 reviews and scores.

```
df = pd.read_json('Arts_Crafts_and_Sewing_5.json.gz', orient='records', lines=True)
df.shape
```

(494485, 12)

```
df.head()
```

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary	unixReviewTime	vote	image
0	4	True	03 29, 2016	AIE8N9U317ZBM	0449819906	{'Format': 'Kindle Edition'}	Zelmira, Ph.D.	Contains some interesting stitches.	Four Stars	1459209600	NaN	NaN
1	5	True	08 12, 2015	A3ECOW0TWLH9V6	0449819906	{'Format': 'Paperback'}	Dangerous when Cooking	I'm a fairly experienced knitter of the one-co...	My current favorite go-to guide for inspiration	1439337600	18	NaN
2	4	True	04 5, 2015	A278N8QX9TY2OS	0449819906	{'Format': 'Paperback'}	Just us	Great book but the index is terrible. Had to w...	lots of great examples, good instructions, col...	1428192000	3	NaN
3	5	True	10 11, 2014	A123W8HIK76XCN	0449819906	{'Format': 'Kindle Edition'}	Amazon Customer	I purchased the Kindle edition which is incred...	Another little gem by Melissa Leapman	1412985600	NaN	NaN
4	5	True	05 8, 2014	A2A6MZ2QB4AE0L	0449819906	{'Format': 'Paperback'}	Sustainability	Very well laid out and very easy to read.\n\nT...	Very comprehensive	1399507200	NaN	NaN



Exploratory Data Analysis

The class distribution is extremely imbalanced and was addressed by under sampling the larger classes before modelling.

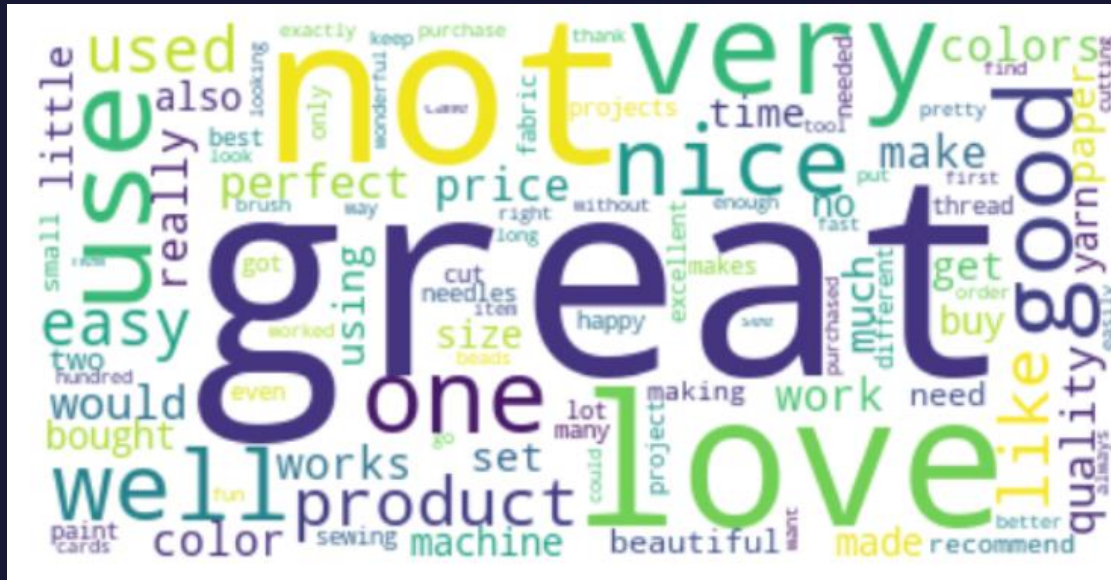
Examining the length of the reviews, I found that the majority are very short. There are outliers, but the plot shows the distribution of review lengths by class, filtered for display purposes by reviews that are less than 100 words in length.



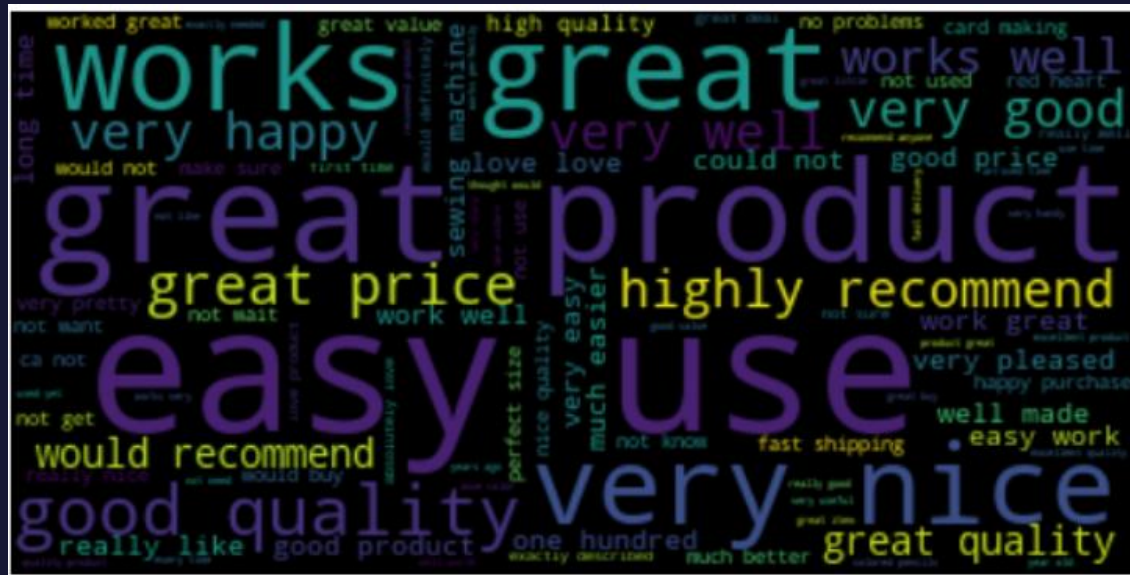


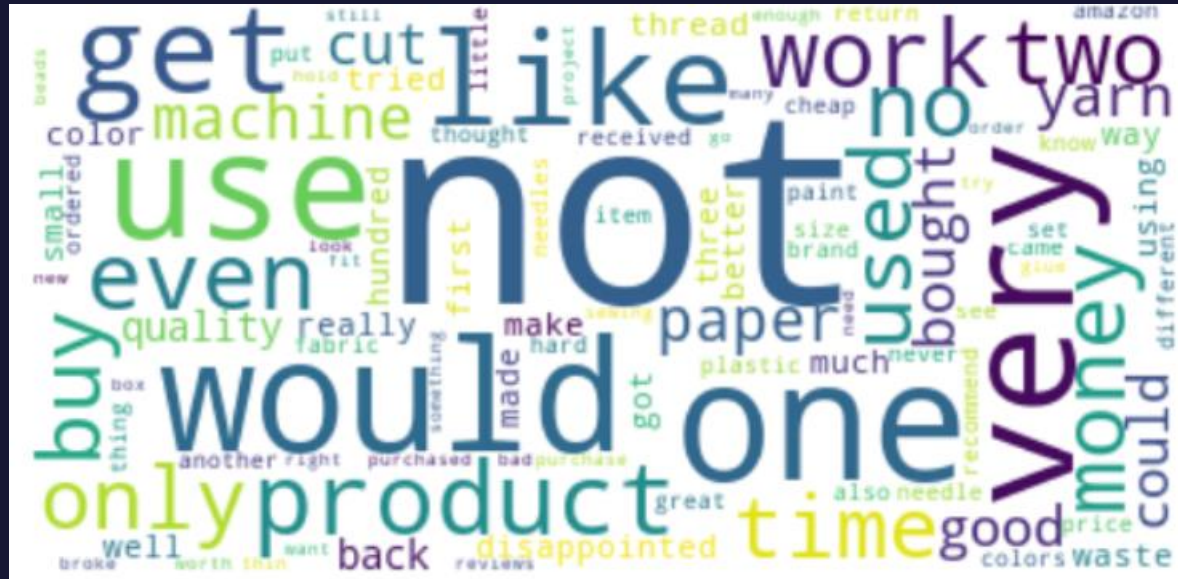
Exploratory Data Analysis

- The most common words that appear across the set of reviews are common English stopwords - "the", "and", "to", "it", "for", and so on. These are removed to simplify the data.
- I examine the most common words and two word combinations for positive, neutral, and negative reviews. The size of the word or phrase in the following word clouds is representative of its prevalence in the data.

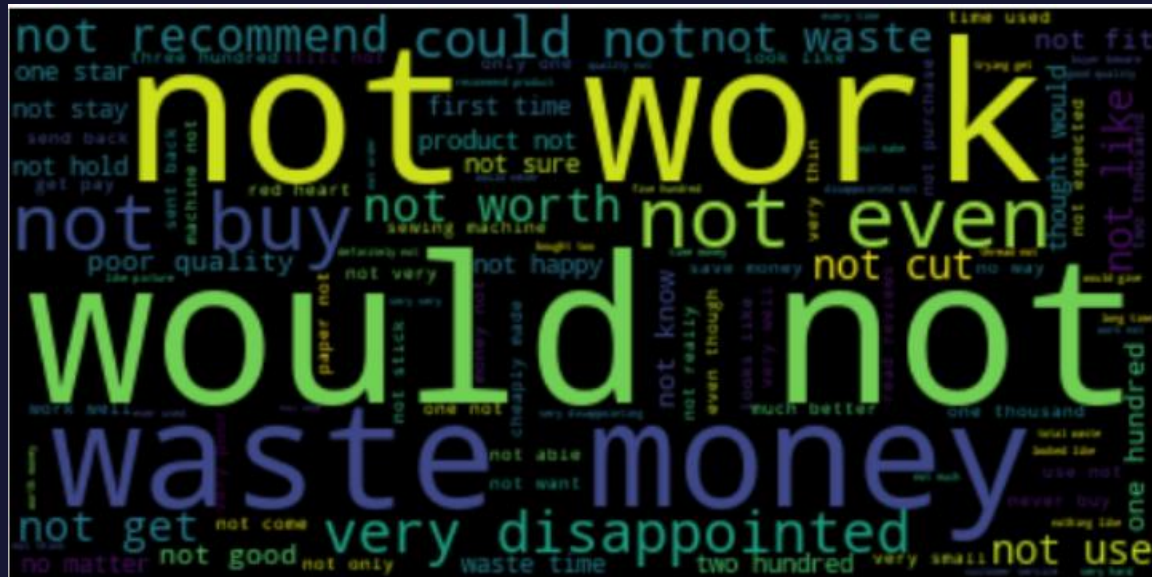


Positive Reviews





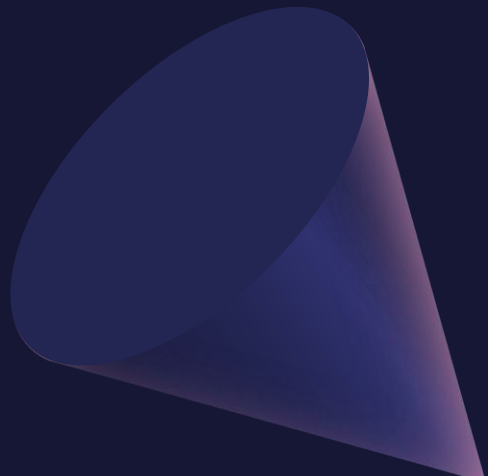
Negative Reviews





Data Preprocessing

I perform the following data cleaning techniques:

- Expanding all contractions
 - Removing line breaks
 - Removing all punctuation and special characters
 - Converting numbers to text
 - Lowercasing all text
 - Removing stopwords
 - Stemming
- 

Preparation for Modelling

- I combine ratings of 4 and 5 into the "positive" class, 3 into the "neutral" class, and 1 and 2 into the "negative" class. I split the data into training and testing sets.
- I under sample the positive and neutral classes to evenly distribute the classes in the training data.

Distribution of classes
in training data

```
positive    25757  
negative    25757  
neutral     25757  
Name: label, dtype: int64
```


Model Testing

- I have tested six classification algorithms:

- Decision Tree
- Random Forest
- Logistic Regression
- Support Vector Machines
- Naïve Bayes
- K-Nearest Neighbours

- I transform the data using a Count Vectorizer with specifications:

- Unigrams only
- Bigrams only
- Unigrams and Bigrams
- Unigrams, Bigrams, and Trigrams

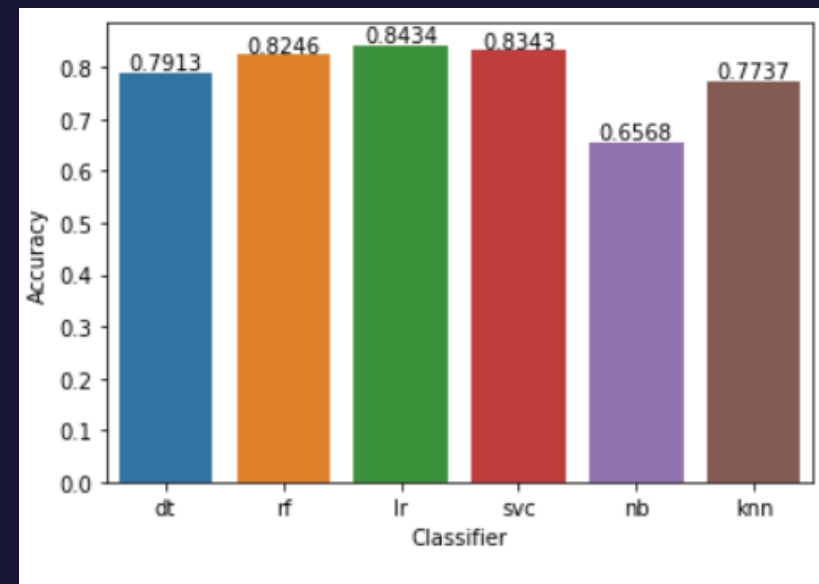
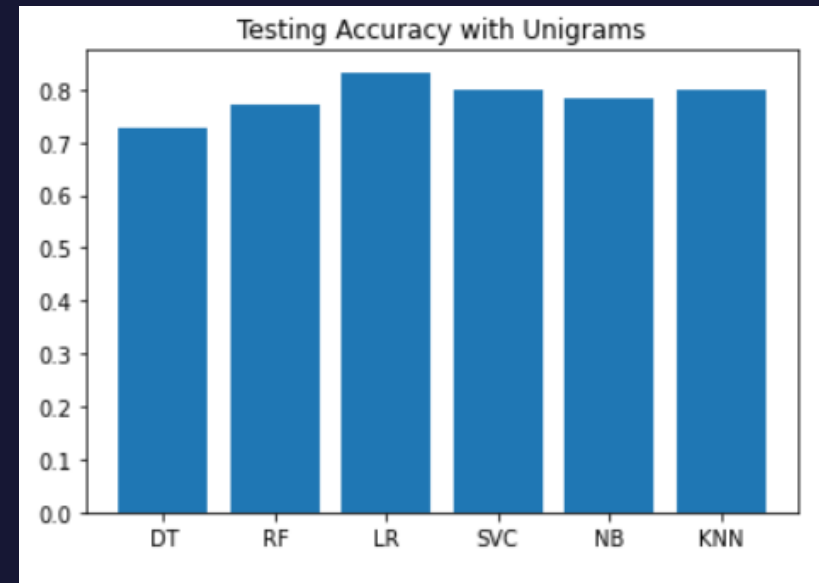


Accuracies

These are the performances of the six algorithms on the held out testing dataset using unigrams only and bigrams only.

At this point I focussed on the Logistic Regression Classifier for two reasons:

- Very high accuracy
- Very fast training time compared to SVC and Random Forest





Tuning the Logistic Regression Classifier



	Vectorizer	Solver	Multi Class	C	Testing Accuracy
Logistic Regression	Unigrams	Liblinear	Auto	1	0.8328
Logistic Regression	Bigrams	Liblinear	Auto	1	0.8434
Stemming Added					
Logistic Regression	Bigrams	Liblinear	Auto	1	0.8509
Logistic Regression	Unigrams + Bigrams	Liblinear	Auto	1	0.8577
Logistic Regression	Unigrams + Bigrams + Trigrams	Liblinear	Auto	1	0.8607
Logistic Regression	Unigrams + Bigrams + Trigrams	Liblinear	One Vs Rest	1	0.8607
Logistic Regression	Unigrams + Bigrams + Trigrams	Newton-CG	One Vs Rest	1	0.8509
Logistic Regression	Unigrams + Bigrams + Trigrams	Newton-CG	Multinomial	1	0.8517
Logistic Regression	Unigrams + Bigrams + Trigrams	Liblinear	One Vs Rest	0.5	0.8542
Logistic Regression	Unigrams + Bigrams + Trigrams	Liblinear	One Vs Rest	0.1	0.8590

The Final Model

- I achieved the highest accuracy using the combination of unigrams, bigrams, and trigrams
- Overall, my model is performing very well.
- Its recall and precision is best on positive reviews.
- For neutral reviews there is a relatively large gap between precision and recall, indicating the model is predicting a neutral sentiment a lot more than it should be.

```
# As noted, I will use a Count Vectorizer with a combination of unigrams, bigrams, and trigrams
# This combination produced the highest accuracy in the algorithm testing notebook
```

```
vectorizer = CountVectorizer(ngram_range=(1,3))
```

```
# Vectorization
```

```
X_train_cv = vectorizer.fit_transform(X_train)
```

```
X_test_cv = vectorizer.transform(X_test)
```

```
# Fitting the Logistic Regression classifier and checking the accuracies
```

```
lr = LogisticRegression(solver='liblinear', multi_class='ovr')
```

```
lr.fit(X_train_cv, y_train)
```

```
lr_train_preds = lr.predict(X_train_cv)
```

```
lr_preds = lr.predict(X_test_cv)
```

```
lr_train_acc = accuracy_score(y_train, lr_train_preds)
```

```
lr_test_acc = accuracy_score(y_test, lr_preds)
```

```
print("Training Accuracy:", lr_train_acc)
```

```
print("Testing Accuracy:", lr_test_acc)
```

```
Training Accuracy: 0.974078244101927
```

```
Testing Accuracy: 0.8607281868310087
```

	precision	recall	f1-score	support
negative	0.54	0.79	0.64	1373
neutral	0.30	0.66	0.41	1440
positive	0.99	0.88	0.93	21851
accuracy			0.86	24664
macro avg	0.61	0.78	0.66	24664
weighted avg	0.92	0.86	0.88	24664

Model Examination

Comparing the class coefficients to the features of the Count Vectorizer, I can produce a list of words and phrases deemed most predictive for each class. Here I display the top five for each class.

Negative

```
('junk', 3.0726358223736043)
('useless', 2.7383975515160652)
('no good', 2.5882426497003044)
('horribl', 2.5234650915169516)
('terribl', 2.3688897595392047)
```

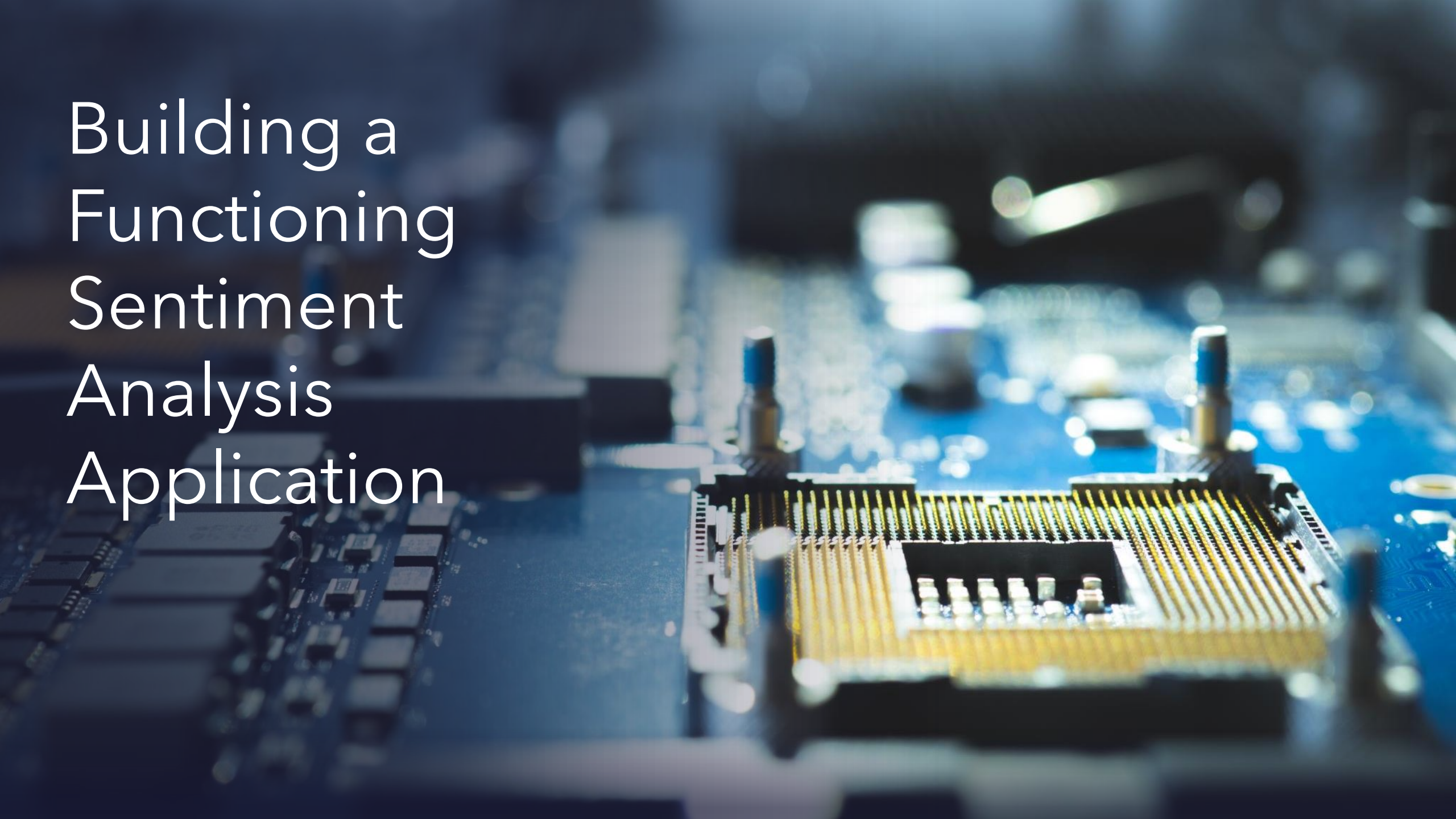
Neutral

```
('three star', 4.0288150141789885)
('okay', 2.350508982553373)
('ok', 2.1440595823248274)
('averag', 2.0967954420772332)
('not bad', 2.0171706545013692)
```

Positive

```
('awesom', 3.0363366564732814)
('perfect', 3.0031275090763097)
('love', 2.8913258130581916)
('excel', 2.8235221812581006)
('never disappoint', 2.524678366389776)
```

Building a Functioning Sentiment Analysis Application



Data Pipeline

With this function I take and process user text. I use the model to predict the sentiment and probabilities for each.

I use the coefficients and the vectorizer's features to generate the key phrases of the user's input.

```
def predict_sentiment_with_analysis():  
    # Prompt the user for text  
    user_text = input('Enter Your Text Here: \n\n')  
    # Process  
    processed_text = [process_input(user_text)]  
    # Transform the text according to the Count Vectorizer that has been fit  
    vec_text = vectorizer.transform(processed_text)  
  
    # Predict the class label  
    predicted_sentiment = lr.predict(vec_text)  
    # Generate the probabilities for the labels  
    confidence = lr.predict_proba(vec_text)  
  
    # Get the features of the input text and store the English words or phrases  
    phrases = []  
    for item in vec_text.indices:  
        phrases.append(vectorizer.get_feature_names()[item])  
  
    # Create a dictionary of the model's coefficients for these features  
    importances = dict()  
    for phrase in phrases:  
        if predicted_sentiment == 'negative':  
            importances[phrase] = negative_coef.get(phrase)  
        elif predicted_sentiment == 'neutral':  
            importances[phrase] = neutral_coef.get(phrase)  
        elif predicted_sentiment == 'positive':  
            importances[phrase] = positive_coef.get(phrase)
```

Sample Output

```
predict_sentiment_with_analysis()
```

Enter Your Text Here:

I was looking to get back into painting and these paints were incredible! I was unsure of whether to purchase or not because they were much more affordable than some other options I saw but I am so glad I purchased these. Whether these are for children or for yourself, you can enjoy and you really get the feel that they are high quality. I've worked with expensive acrylic paints before and these are very similar. They blend very seamlessly together, and the colour selection is absolutely gorgeous. The paint tubes are actually a decent size so you have good value there as well. None of the paints were dried up or anything, and so far they have exceeded my expectations. They arrived in very nice packaging as well so I would imagine that it would make a great gift. Highly recommend this!!

The Predicted Sentiment is: POSITIVE

Analysis:

Probability of Negative Label: 0.0 %

Probability of Neutral Label: 0.13 %

Probability of Positive Label: 99.87 %

The Five Most Important Stemmed Words or Phrases Are:

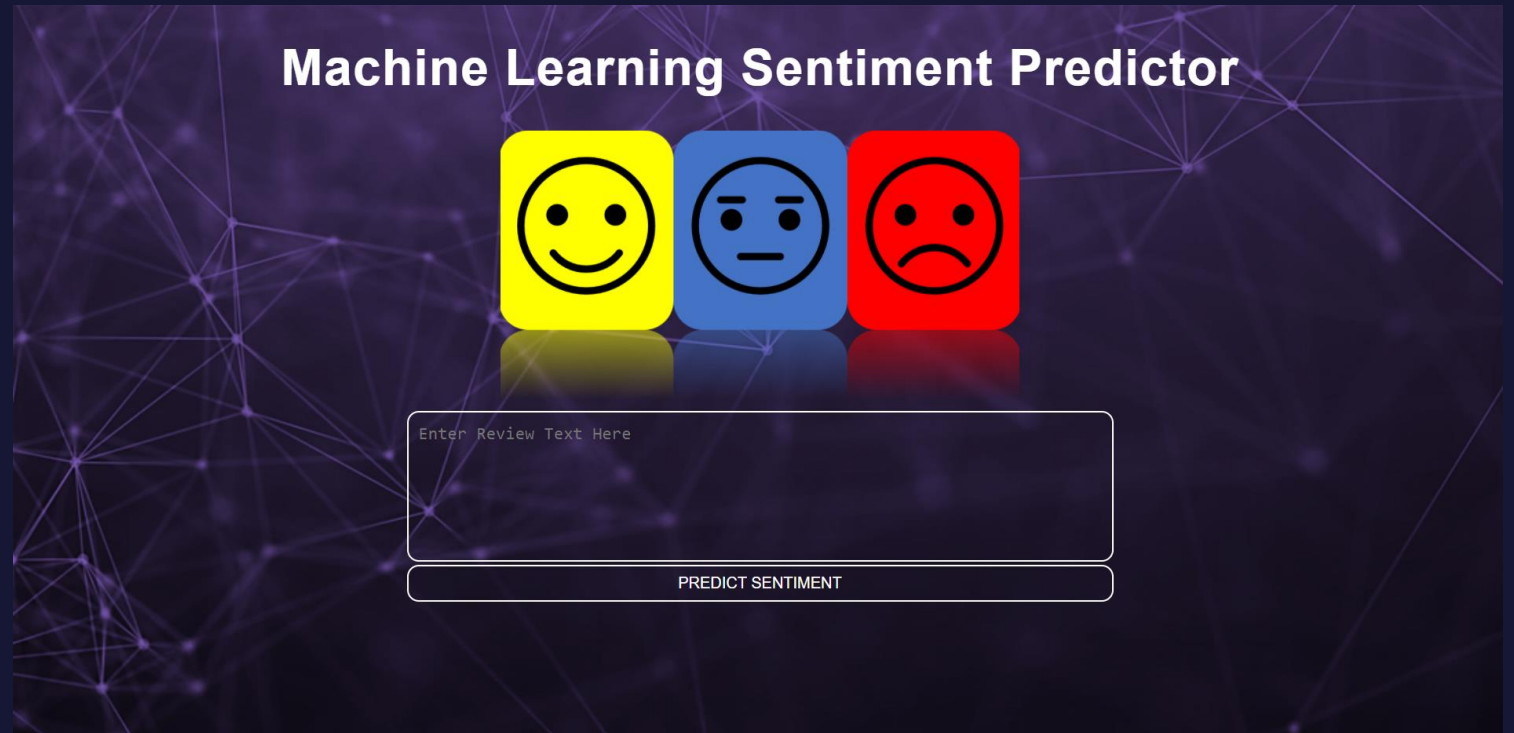
	Word or Phrase	Model Coefficient
First	great	2.518729
Second	glad	1.994195
Third	gorgeous	1.860844
Fourth	enjoy	1.394096
Fifth	nice	1.380592

Deployment

I have built a Flask application and styled it using HTML and CSS.

It is available for use at:

<https://sentiment-analyzer-mm.herokuapp.com/>

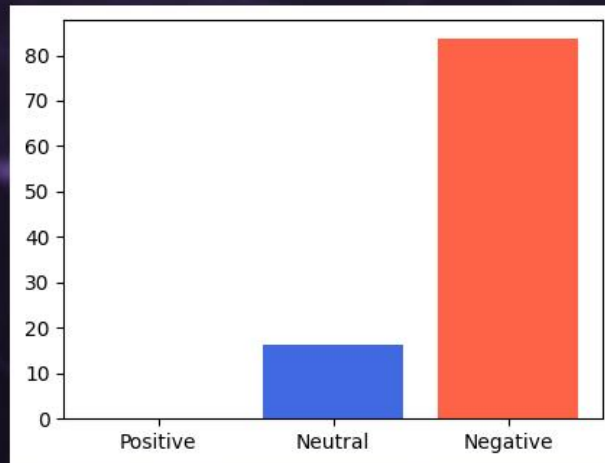


PREDICT SENTIMENT

Your Text:

We ordered the full 3 sets and before we knew it they were breaking constantly. Thinking it was our sharpener, we used a different one and the problem persisted. You will get breakages but not losing 50mm in less than a hour. Before long that pencil will be gone. We returned ours for a full refund and bought some Castles, much better. Don't know how these are constructed, but it looks like a 2 piece outer with pressured stain, which disguises the grain whether good or bad. We contacted the seller who was very sympathetic and sent us a free box of Cobras, but sadly no different, actually these were worse.

The Predicted Sentiment is: **NEGATIVE**



Probabilities:

Positive: 0.0%
Neutral: 16.39%
Negative: 83.61%

Sample Output

Conclusion

I have created and deployed a Sentiment Analysis tool that accurately predicts the sentiment of any input review text.

As described in the beginning, this application can be used by businesses to automatically process and analyze everything that is being written about them.

These insights will allow businesses to better their sales and customer experience by making decisions based on real-time analysis of how their actions impact upon how their company is viewed.



Thank you