# Using NLTK to Detect Sarcasm

Damon Porter

IST 664 – Natural Language Processing

## Introduction

“Obviously!”, “Good luck with that!” While these phrases can be taken at face value to express agreement or well wishing, they are commonly utilized in a sarcastic manner conversationally. Even when used in face to face conversations, sarcasm can be, at times, difficult to process; This becomes even more difficult when the sarcasm is written, typed, or texted. According to a study by psychological scientist of Chatham University that measured individual’s accuracy at gauging the emotional tone of emails sent by both friends and strangers, “We’re terrible at it- even when we’re corresponding with our friends.” Their study found that humans, even if they feel confident in their interpretation of an email or text, often fail to accurately identify the emotional toll of the interaction.

These are interactions are being misconstrued by humans, so could a machine possibly accurately predict sarcastic text? Through the use of text mining and Natural Language Processing/Understanding machines have grown to understand how many different human languages are interpreted, the different parts of speech, and even the sentiment of speech, but can it predict when a text says one thing, but means the exact opposite?

To determine if, through the use of classification algorithms and natural language processing, a machine is able to classify sarcasm data, data was collected from Reddit with comments labeled as “Non-Sarcastic” vs “Sarcastic”.

## Data Collection and Preprocessing

### Data Collection

The data for this research project was collected from the Kaggle dataset ‘Sarcasm on Reddit.” This dataset contains 1.3 million comments from Reddit. These comments were combined into a csv and balanced by taking an even amount of sarcastic, labeled as ‘1’, and non-sarcastic comments, labeled as ‘0’. To perform this analysis, a subset of data, 20,00 records evenly distributed was utilized. This data was then converted into a corpus by utilizing the built in NLTK Corpus creator and read back into Python.

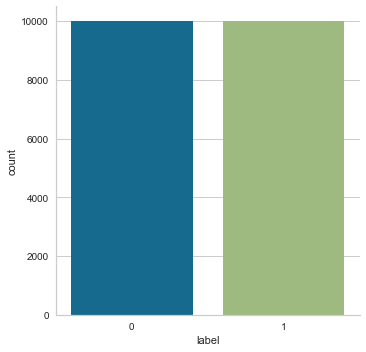


Figure : Data distribution of sarcastic vs non-sarcastic comments

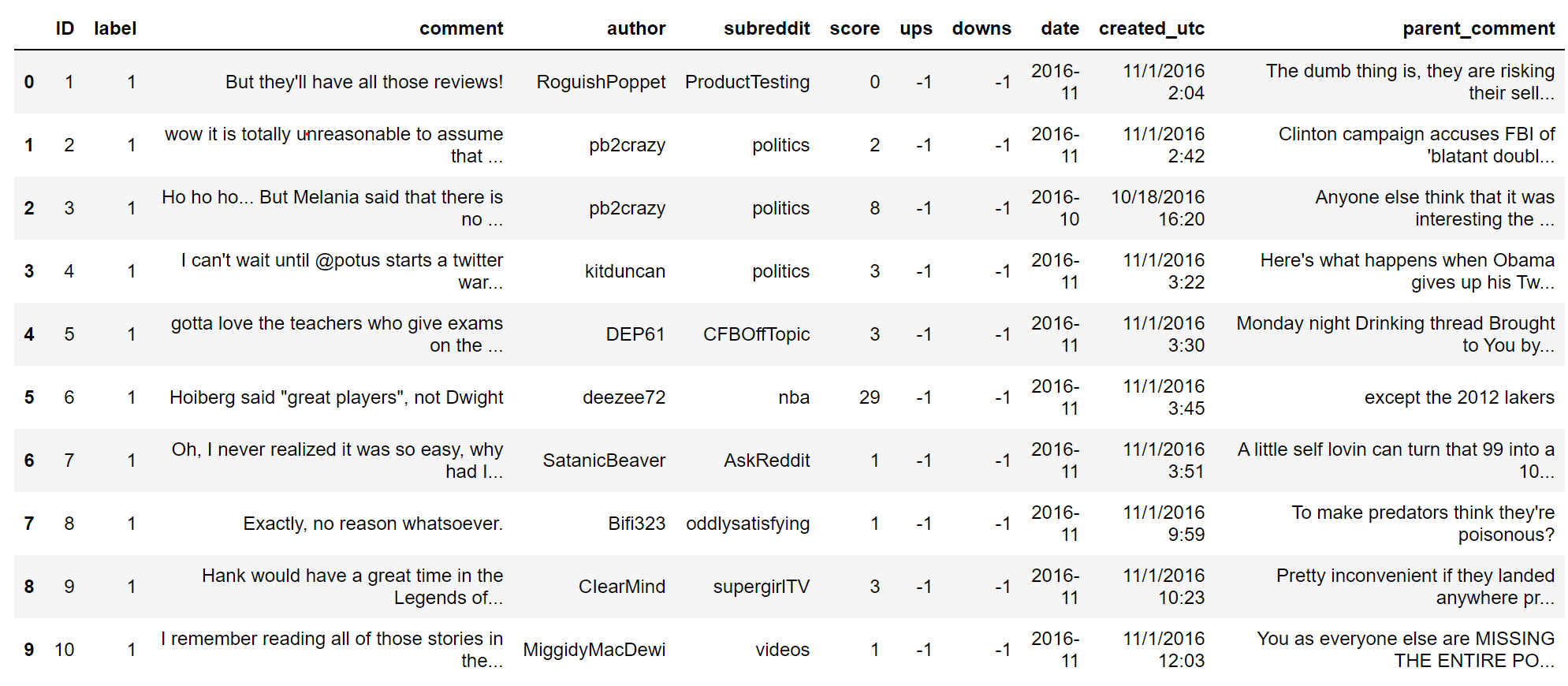
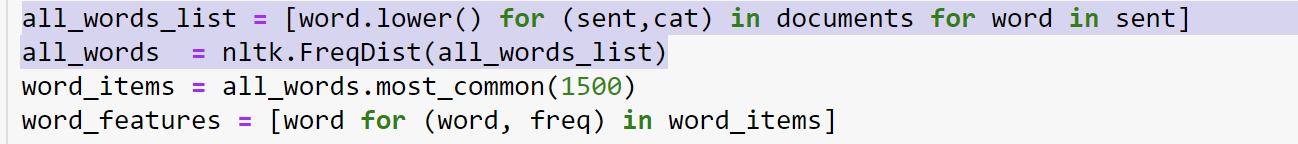


Figure : Data Example

### Data Preprocessing

In order to consume and extract useful data from the comments text, it was necessary to vectorize, tokenize, and standardize the text data. For the first experiment, the text was lowered and the most common 1500 words were extracted from the Corpus.



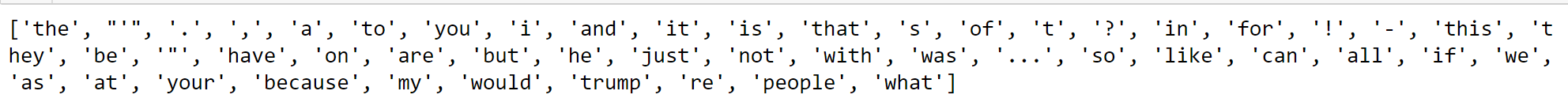
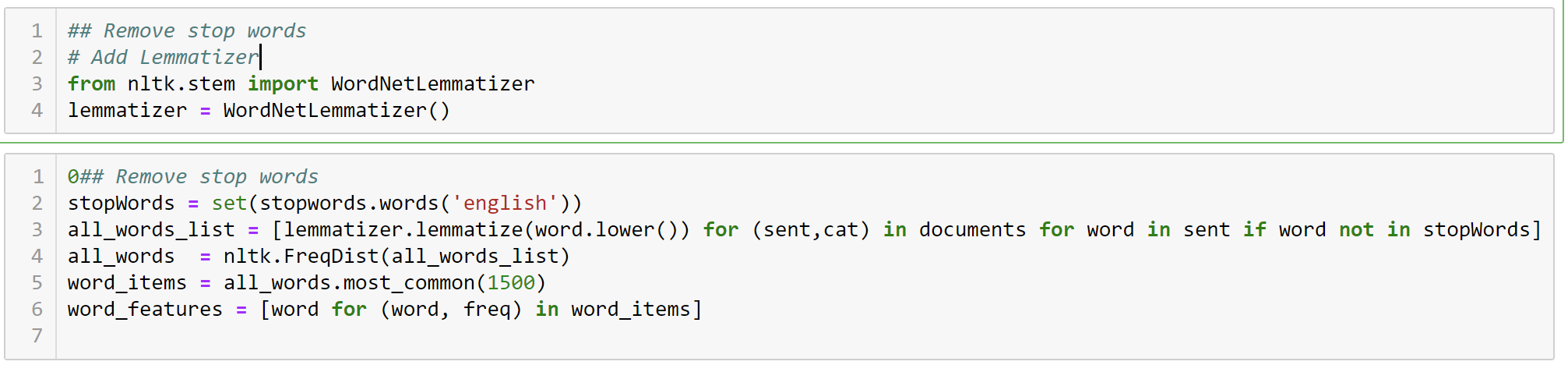
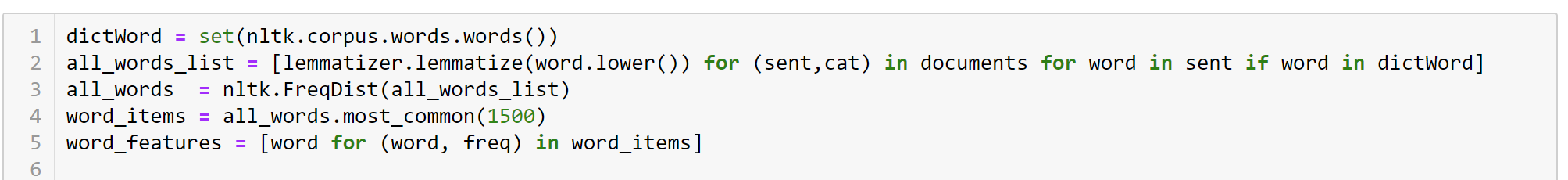


Figure : Word examples

For subsequent experimentation, words were lemmatized, similar words were grouped, to see if similar words were used in the same context to denote sarcasm. Stop words, or overly common words, were removed to determine the value they added to detect sarcasm or not.



In order to test the predictability of n-grams (multiple words), bigrams and trigrams were tokenized. Bigrams are combinations of two words, while Trigrams are the combinations of three words. In order to determine a level of importance in these word combinations, instead of just taking the raw frequency of bigram and trigram occurrences, Pointwise mutual information, or PMI, was used. PMI is a statistical measure of association between the word combinations. It provides the statistical weight of if a word occur how likely is this other word to follow. The thought behind utilizing PMI in conjunction with the n-grams was to analyze if the algorithm would find common sarcastic comments such as ‘Yeah…Okay’ or ‘Good luck with that!’ For this analysis, it was also necessary to limit the words to the those in the English language. During the data exploration phase, the n-grams were commonly returning the user name of the commenter, so the NLTK.Dictionary was utilized to limit the words selected for the n-grams. It should also be noted that punctuation and emoticons were left inside the corpus due to their ability to often denote sarcasm. Phrases such as ‘…OK’ or ‘;-P’ often are signs that what the commenter is typing should be taken sarcastically or in jest.



## Analysis

### Experiment 1: Baseline test

The initial experiment was tested against a minimally tokenized corpus. This was to determine a baseline for the Naïve Bayes classifier’s ability to detect sarcasm. The only feature extraction utilized for the initial test was to lower all the words in the corpus and to take the most commonly occurring 1500 words in the corpus. Once this feature set was developed, it was spilt, 90/10, into a training set and a test set. The training set was then utilized to train the Naïve Bayes classifier and the test set was used to determine the effectiveness of the classifier.

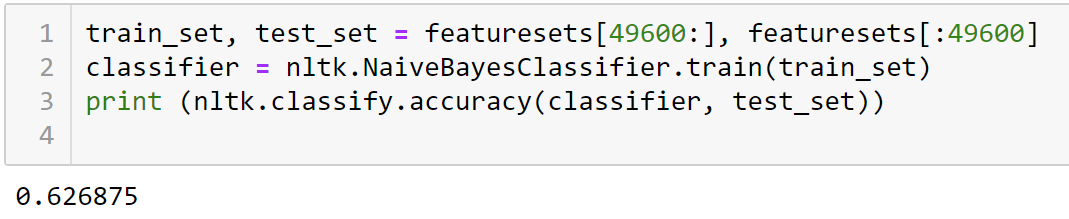


Figure : The accuracy of the first classifier is 63%.

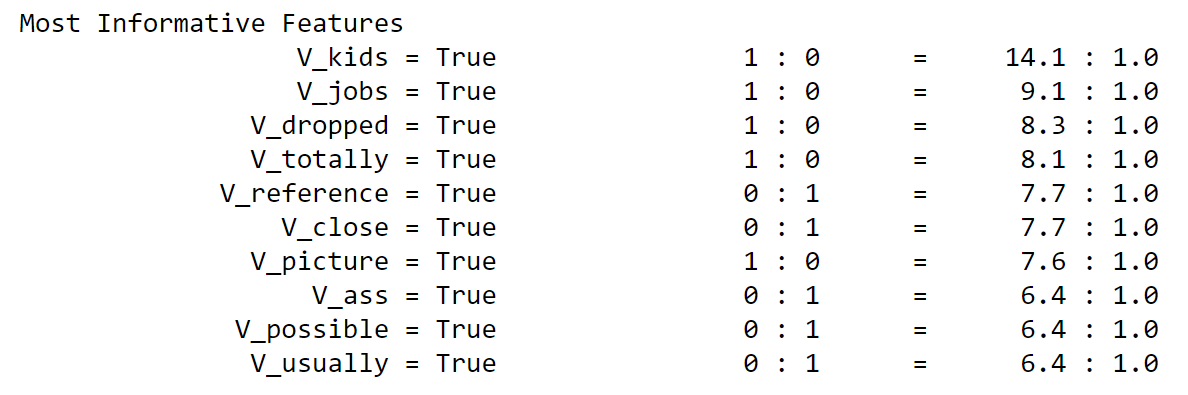
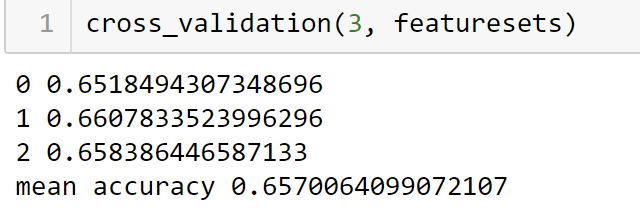


Figure : This list displays which features(words) were most informative in detecting sarcasm. The following displays that it is most likely that the words 'kids, jobs, dropped, and totally" were effective in finding sarcastic comments. While ‘reference’ and ‘close’ were more likely found in non-sarcastic comments.

In order to determine how well this classifier performed against other sets of that, cross validation was utilized to reperform the classifier against different samples of the training data. To ascertain a true score, the cross validation was run 3 times to get a mean score. The mean score of the model is now 66% which is a slight increase.



Finally, the predictor is compared against the actual label to find the precision, recall, and F1. The **precision** is the correctly predicted positive observations to the total predicted positive observations. In this case, it is **64%** for non-sarcastic comments and **63%** for sarcastic comments. The **recall** is the ratio of correctly predicted positive observations to the all observations in the actual class. The recall for this model **65%** for non-sarcastic to **62%** for the sarcastic. The F1 is the weighted average for both recall and precision. Since the dataset we used is balanced it doesn’t skew much from the accuracy. The non-sarcastic **F1** is **65%** to **63%** sarcastic F1.

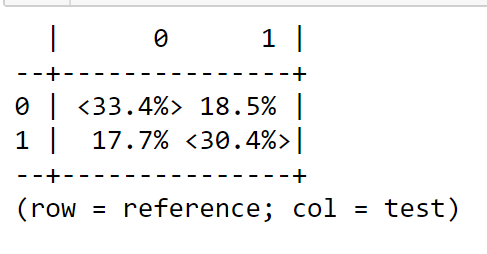


Figure : Confusion Matrix displaying accuracy of the model

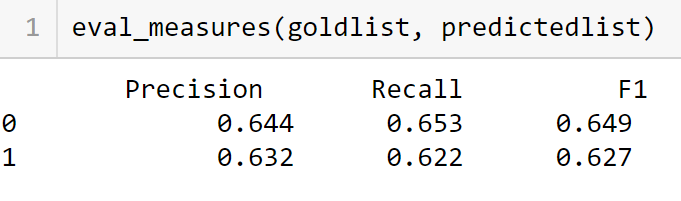
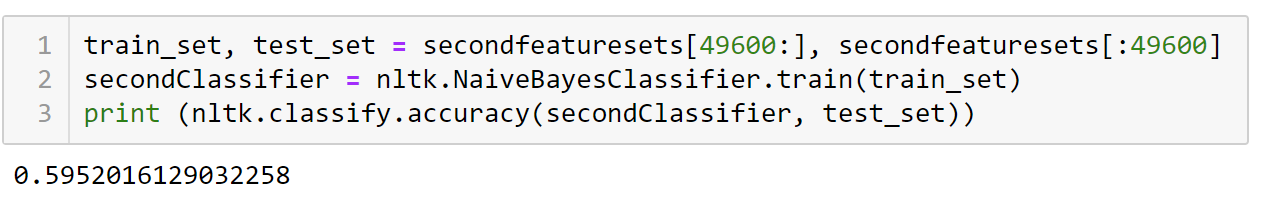


Figure : Precision, Recall, and F1

### Experiment 2: Removing Stop words/Lemmatizing vocabulary

The second experiment modified the corpus by removing extremely common English language words and lemmatizing the data. As seen below, removing these words actually had a slightly decreased the accuracy of the predictor. The new accuracy score is **60%**.



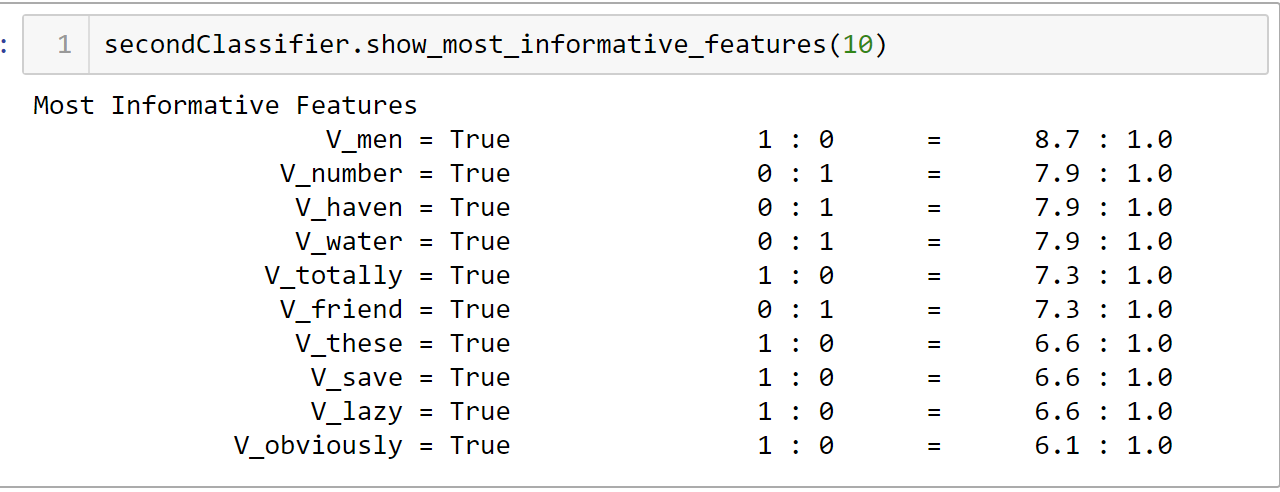


Figure : Removing the stop words and lemmatizing the data shifted the top predictors from predicting sarcasm to detecting non-sarcastic comments

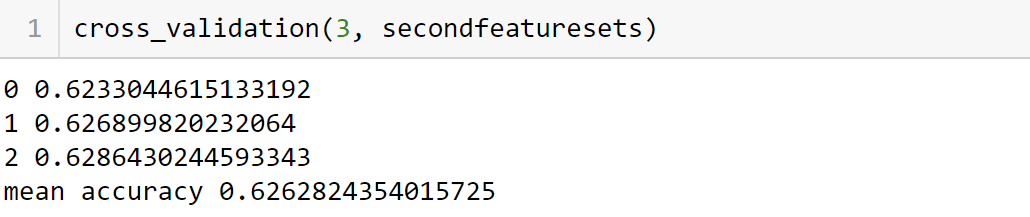
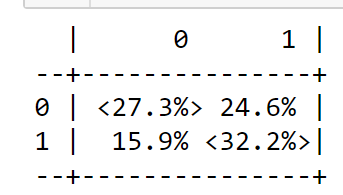


Figure : The accuracy score actually raises by 2% when crossvalidated



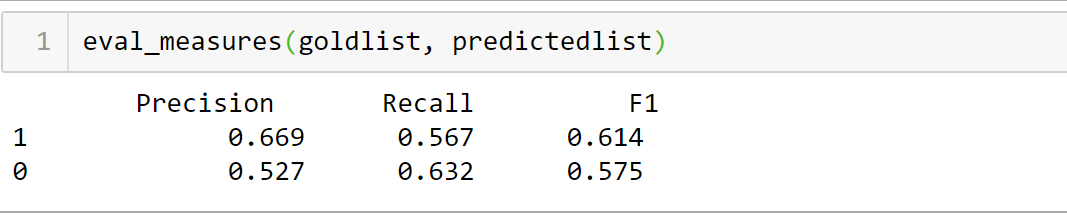


Figure : With this classifier, there is a slight increase in the model’s precision in detecting sarcasm and recall of non-sarcastic comments, the other measures are all much lower than the initial experiment.

### Experiment 3: N-Grams

The next two classifiers were built with a bigram and trigram features. The stop words were added back into the vocabulary and it was also un-stemmed. As previously stated, a dictionary was created for these steps to remove non-English words and user’s names.

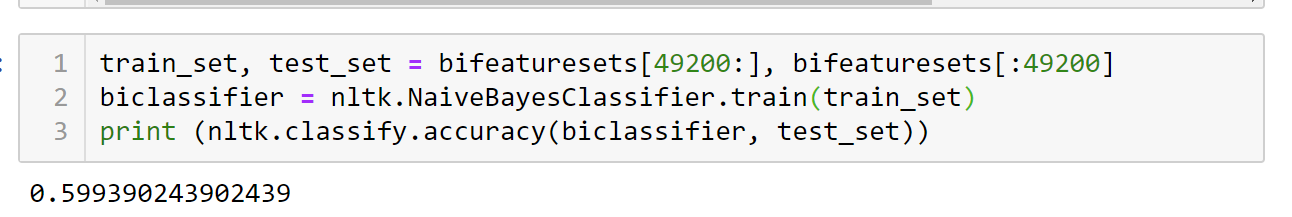


Figure : Bigram Accuracy: 60%

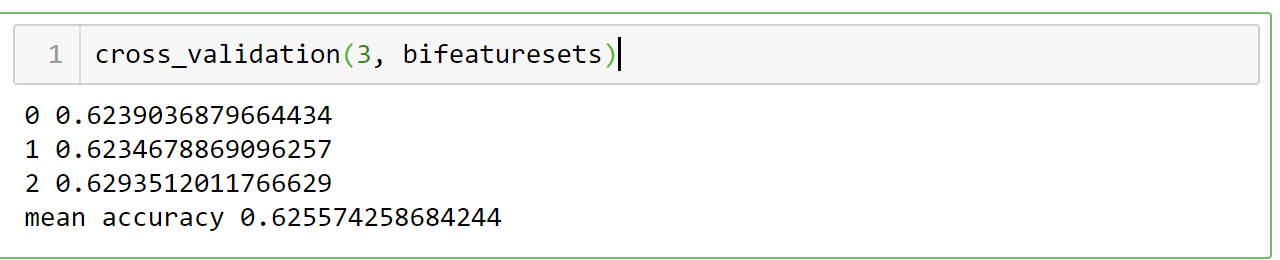


Figure : As seen with the other models, the cross validation increases the accuracy of the model by approximately 3%

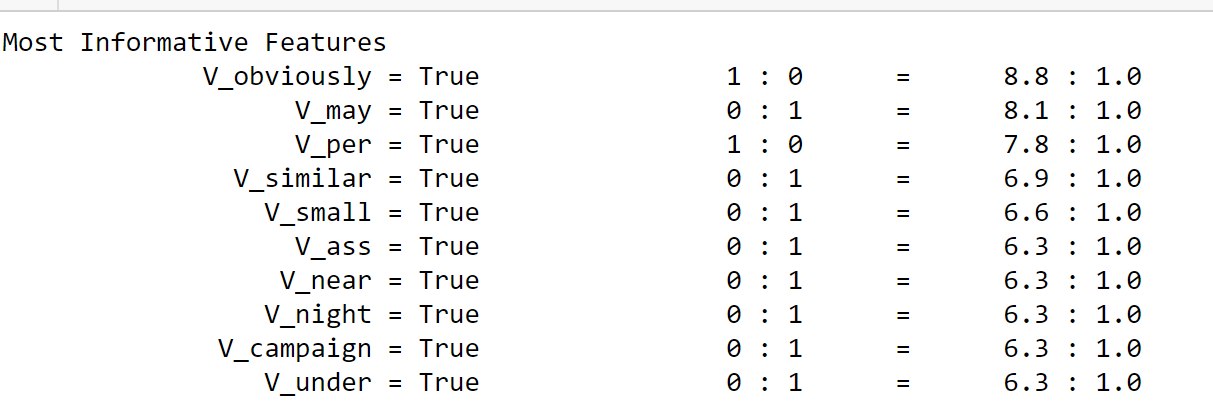
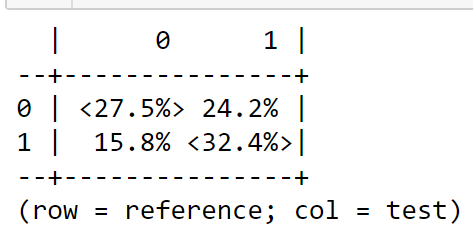


Figure : Interestingly, neither bigrams or trigrams were the most informative features in their feature sets.



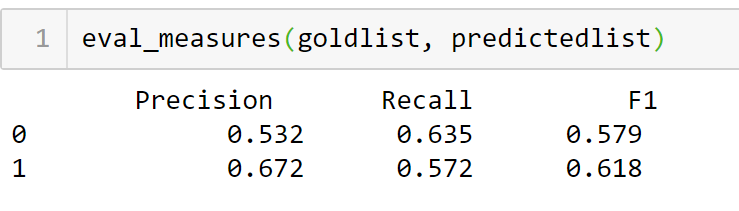
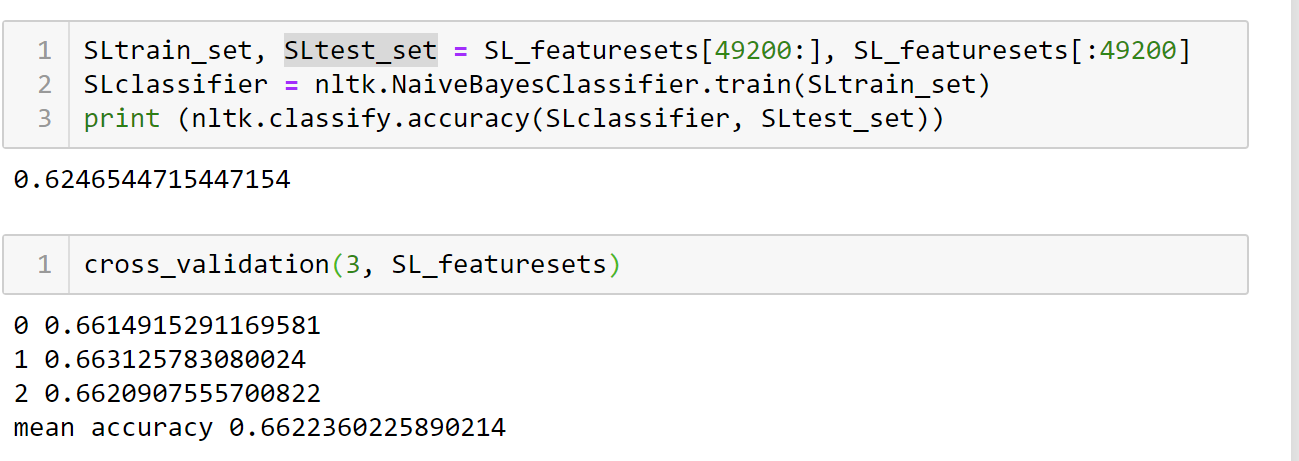


Figure : Once again, the model does well with the precision of sarcastic comments and the recall of non-sarcastic comments, but still isn't as accurate compared to baseline model.

These models actually decreased in accuracy when compared to the unigrams used in the other models.

### Experiment 4: Sentiment Lexicon

The last experiment focused on creating a sentiment lexicon and looking at the subjectivity of each word. This was an attempt to see if sarcasm could be predicted by detecting the sentiment behind the sentences. The findings we more on par to what was observed during the baseline test. The model’s accuracy was 63% with a cross validation score of 66 percent.



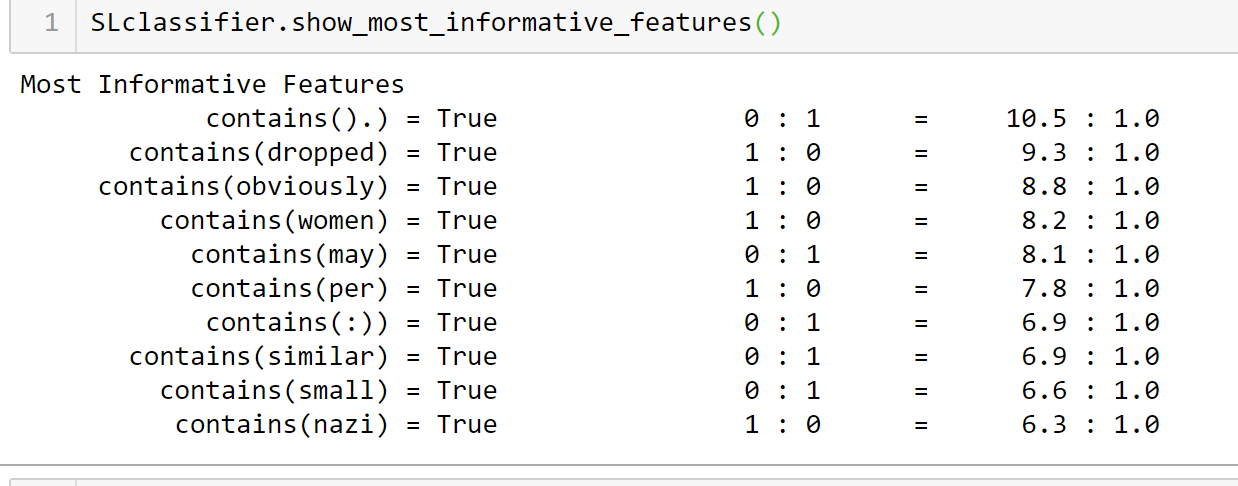


Figure : Obviously is a word that has appeared across multiple models as a strong predictor of sarcasm.

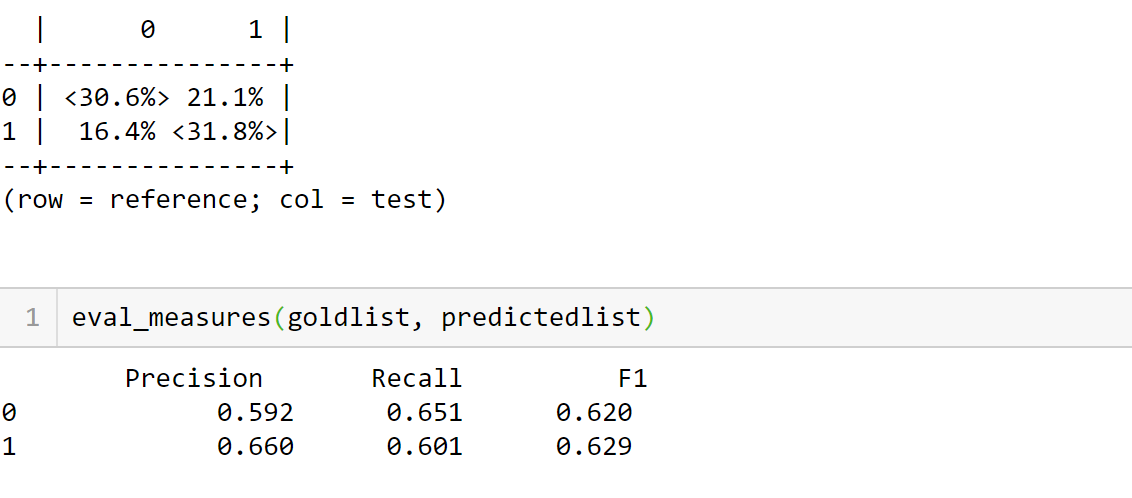


Figure : This model also performs as well as the baseline model on Precision, Recall, and F1

## Conclusion

Poe’s law is an adage on the internet that states that without a clear indication of the author’s intent, it is impossible to something so extreme that it can’t be misinterpret by some individuals. In fact, there is a subreddit(r/AteTheOnion) dedicated to people who take articles from the Onion.com, a known satirical website, as fact. With human’s struggling to draw the line between sarcasm and earnestness, it is no surprise that machine’s struggle somewhat as well.

This research seemed to further validate that theory as most of the models run, were only able to successful distinguish the difference between sarcasm and non-sarcasm with a 60-65% accuracy. This accuracy score was also consistent when the training data was utilized on other machine learning algorithms such as: Multinomial Naïve Bayes, Decision Tree, and Bernoulli Naïve Bayes.; (These test can be found in the accompanying Juptyer notebook) There were features extracted that were highly correlated with sarcasm such as: Obviously, Dropped, and Totally. Totally and Obviously seem straightforward in their conations of sarcasm, but dropped isn’t as apparent.

MIT has created an AI the focus on identifying the emotions in tweets. It has a success rate of 82% vs human’s ability (76%). Given the opportunity to revisit this analysis, it could be interesting to see how incorporating the original message and observing how the comment correlates. It could also be interesting to find a dictionary of words considered to be sarcastic, add weight to sentences that contain those words, and attempt to classify the comments again.