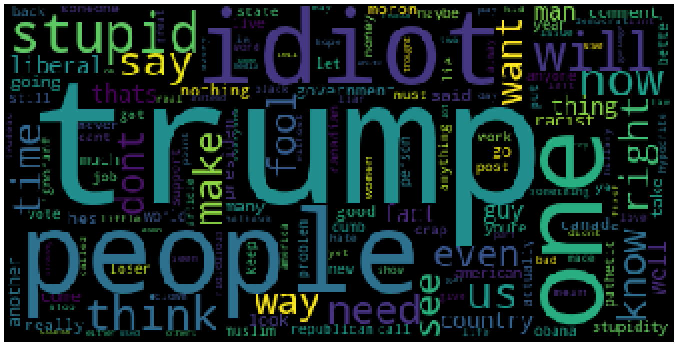
Springboard - Data Science Career Track

Capstone Project 1
Identifying Toxic Commentary

By Ellen Savoye - July 7, 2020



Introduction

Openly discussing things that you feel strongly about, or care about can be difficult; even more so online or in discussion boards where the threat of abuse and harassment can be prominent. Many messaging and discussion-based platforms struggle to filter out toxic and outwardly offensive comments including comments that are rude, disrespectful or otherwise likely to make someone feel vulnerable and leave a discussion. To combat this, ultimately, most platforms limit or completely remove comment sections as a result. With the advancement of computing technology, including AI, assessment and learning, can a code be developed to recognize toxic comments in online conversations with respect to mentions of identities?

Conversation AI, a joint venture between Jigsaw and Google (both subsidiaries of Alphabet), have been the primary business prompting investigation into this problem. Even though this is one of the focuses of the Conversation AI team, the question of recognizing these potentially hazardous comments has a far-reaching impact on more online platforms and businesses than just this joint-venture.

Data

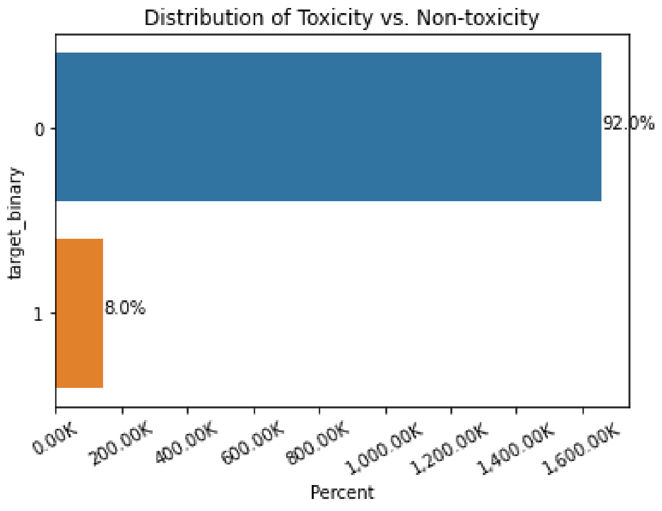
The data used for this project is publicly available at the following address: https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data. These data are originally from Civil Comments, a collection of about 2 million public comments. I sourced the 'train.csv' data used in my capstone report from the aforementioned link.

The original 'train' dataset consists of 45 columns: 24 identity labels, 1 target variable containing float values from 0 to 1, 1 comment_text variable, 6 target label subsets, and 13 metadata variables.

Given the size of the original dataset, approximately 1.8 million records, I took a 10% sample to test various NLP cleaning techniques. The purpose was two-fold. One, test bits of code to ensure it functions as it should. Two, quantify the impact of the code to determine if using a particular cleaning technique was necessary and impactful.

I eventually created a function, called normalize_corpus, to apply to all ~1.8 million records. The function removed emoji, emoticons, punctuation, stop words, and special characters. It also converted my text field entirely to lower case values. I did not remove digits nor did I lemmatize the text. Without running a cleaned corpus through my baseline regression classifiers with what I had done so far, I was hesitant to continue wrangling. It's easy to get lost in the weeds of cleaning up a dataset as there always seems to be something new to consider.

Exploratory & Statistical Analysis



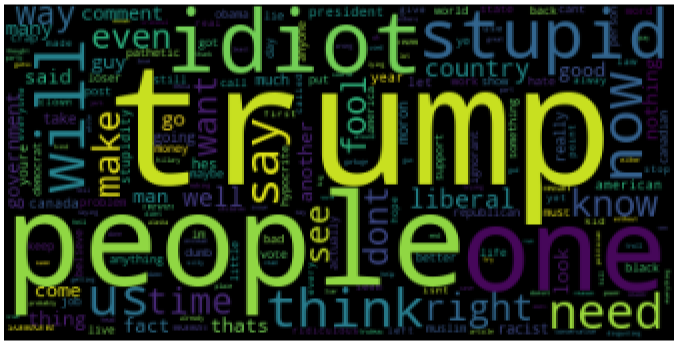
During exploratory analysis, I wanted to determine if my dataset was balanced between toxic and non-toxic comments. My gut instinct was that it wasn't balanced but I didn't realize how unbalanced it actually was. In the graph to the right, you can see that 92% of the

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

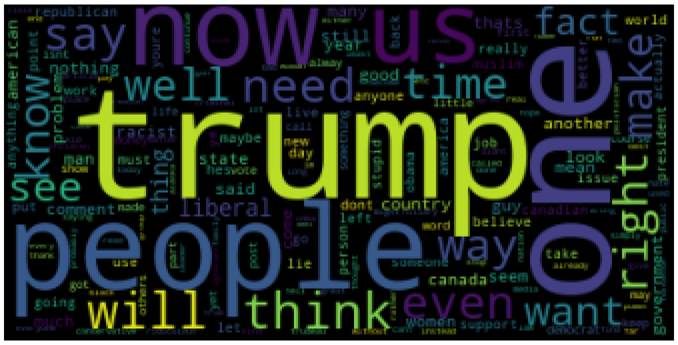
data is labeled 0 or non-toxic. This solidified my belief that I needed to use a confusion matrix with Precision, Recall, and f1 scores to aid in determining model success which we will see in the classifier section later. After checking the balance of data, I was curious if the stop words had as much impact as they did. Using uni-, bi-, and tri-grams, I determined the top 20 words for each. Unfortunately, stop words were prevalent in all 3. As a result, I created word clouds without stop words over 3 buckets using the target variable: high, neutral, and low toxicity.

```
There are 14405533 words in the combination of all 'high toxicity' comments.
There are 43155436 words in the combination of all 'neutral toxicity' comments.
There are 459023008 words in the combination of all 'low toxicity' comments.
```

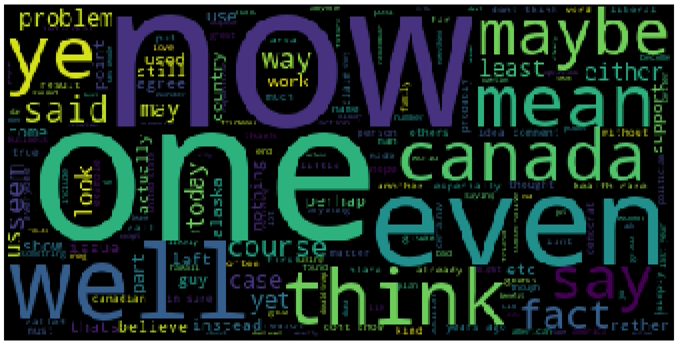
High:



Neutral:



Low:



3
1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Between 'high' and 'neutral', *trump, one, think,* and *people* stand out the most. In comparison to 'low', only *think* and *one* cross all 3 word clouds. The words that stand out in 'low', *canada, even, now*, and *maybe*, tend to have a neutral connotation. Comparatively, *stupid, idiot*, and *trump* from 'high' have more of a negative connotation, especially in the fraught political environment of society today. By breaking the text into bins and creating word clouds, we are given a snapshot to how negative, neutral, and possibly positive words align with the target value assigned.

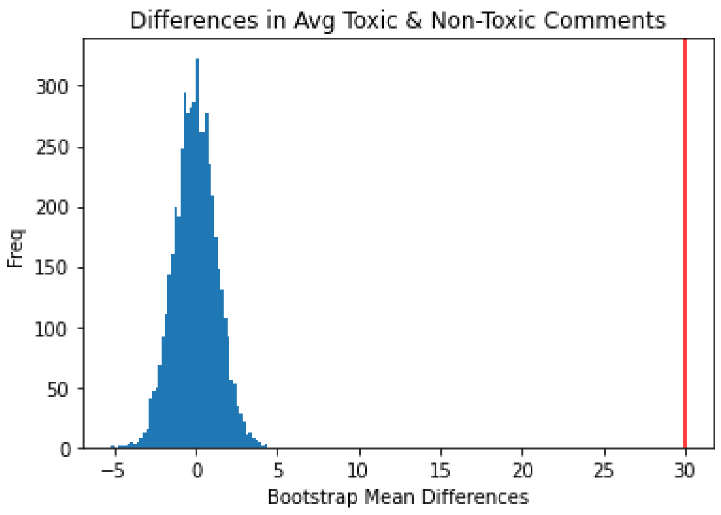
In addition to seeing the distribution of words across toxicity buckets, I was curious as to the length of comments and whether or not toxic comments are the same as non-toxic. In splitting the target variable to toxic and non-toxic, I tested my null hypothesis that the average length of toxic comments and average length of non-toxic comments are the same.

My alternative hypothesis was that there is a variation between the averages of toxic and non-toxic comments.

I used a 5% significance level (α=0.05) and a bootstrapped hypothesis to calculate the p-value of the observed difference between toxic and non-toxic comment lengths.

The p-value calculated was 0.0. Strictly speaking, this means that the observed variation is statistically significant. However, one thing to keep in mind is the larger the data set, the easier it is to get a low p-value. In terms of practical significance and practical considerations, yes it is important.

Knowing that the observed difference between the length of toxic and non-toxic comments is statistically significant also has practical significance in that it lends a certain context. I was originally under the impression that toxic comments were going to be long-winded and longer in length. Between EDA and testing my null

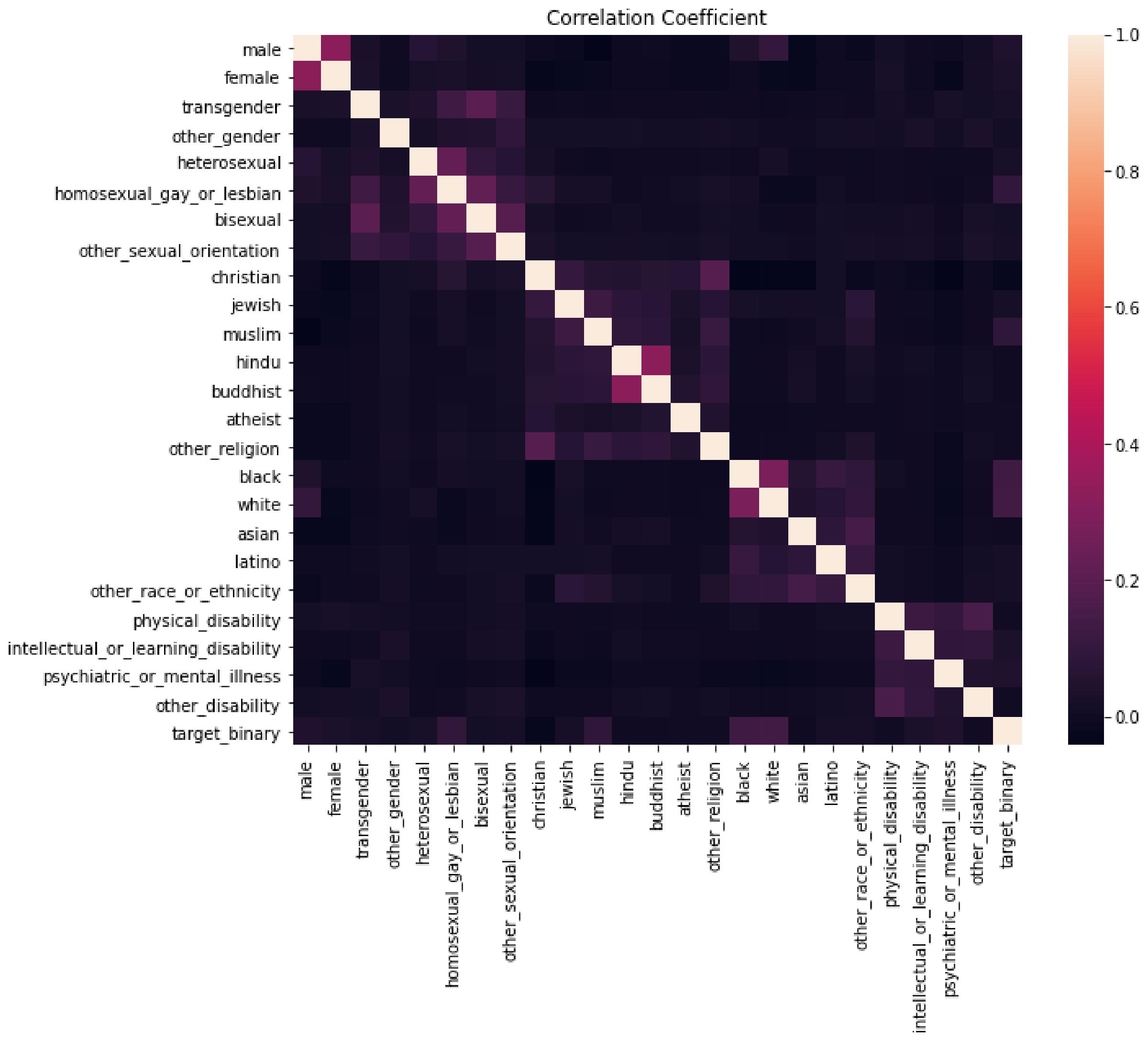


1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

hypothesis, I was shown to be incorrect.

So in the end, I rejected my null hypothesis.

Lastly, I checked the correlation between each identity label and the target variable. We see male/female, hindu/buddhist, and black/white have the highest correlation near 0.4. All remaining correlations do not appear to break above 0.3. By using a dark colored heatmap, the correlations, or lack thereof, are easily discernible.



1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Predictive Classifiers

After creating new features during exploratory analysis and using statistical analysis to determine correlation between features, I went on to build a baseline predictive model using said data.

When I discovered my data was imbalanced, I hypothesized my predictive model would be negatively influenced by this.

To test my hypothesis, I first built 2 baseline models: Logistic Regression and Naive-Bayes. Both Logistic Regression and Naive-Bayes take an X and y input. After applying the **`sklearn.feature_extraction.text.CountVectorizer`**[1] class which converts the collection of text documents to a matrix of word counts, the data is split into train and test sets. I'm using **`stratify`**, an optional input for countvectorizer, to keep the existing proportion between the number of toxic and non-toxic comments. Without using **`stratify`**, the imbalance in my data has the potential to be even worse.

Logistic Regression - Baseline

When using accuracy to determine how 'good' a model is, one must keep in mind whether or not the data is balanced. Most datasets are not balanced and predictive models have a tendency to be biased towards the dominant class. The overall results show high accuracy given how large the dominant class is (92% of the data). To gauge the goodness of the model better, I put together a classification report on both the train and test sets.

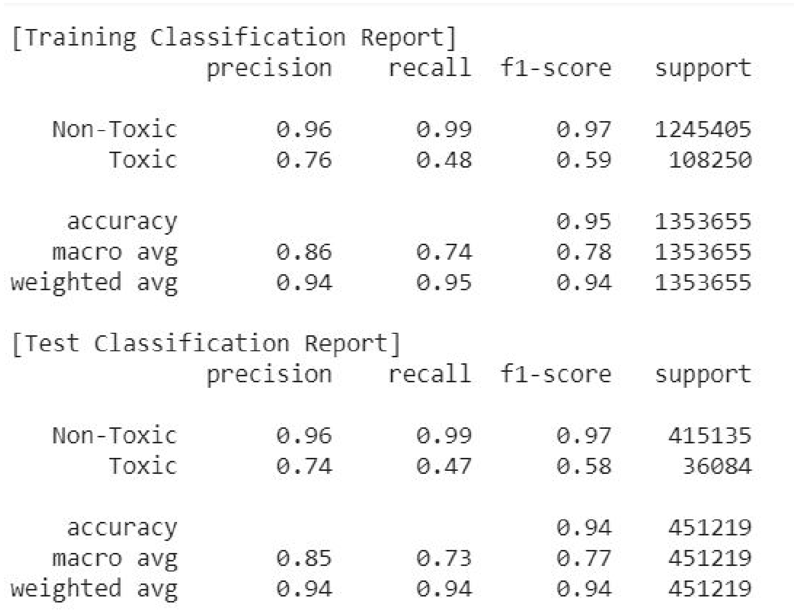
```
[Test] Accuracy score: (ytest, y_predict_test) 0.9443684773912446

[Training] Accuracy score: (ytrain, y_predict_training) 0.9464472114386605
```

There are 2 end goals: one is to determine a predictive model works best and two, test the hypothesis that the model is influenced by the imbalance. We already used stratification when creating the train/test split to mitigate some of the influence of the imbalance by not allowing it to become worse.

Since we can't simply say a model is excellent based on accuracy alone, as I explained previously, we need to take into consideration Precision and Recall. Precision will help us

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

see how precise/accurate our model is by showing how many of the predicted positives (toxic/ True Positive) are actually positive. Recall will help us see how many of the actual positives our model truly captures through labeling it as a positive (toxic/ True Positive). Looking at the results of the Logistic Regression (without hyper-parameter tuning), we see that for the non-toxic label both Precision and Recall, ~0.96 and 0.99 respectively, are approximately the same between the train and test set. This means our model was accurate in identifying 99% of actual positives and mislabeling only 1%. Similarly, Precision is ~96% showing that of the predicted positives, 4% were actually toxic.



```
[Training Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.96      0.99      0.97   1245405
       Toxic       0.76      0.48      0.59    108250

    accuracy                           0.95   1353655
   macro avg       0.86      0.74      0.78   1353655
weighted avg       0.94      0.95      0.94   1353655

[Test Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.96      0.99      0.97    415135
       Toxic       0.74      0.47      0.58     36084

    accuracy                           0.94    451219
   macro avg       0.85      0.73      0.77    451219
weighted avg       0.94      0.94      0.94    451219
```

However, in predicting toxic comments, the model didn't do as well. The train set's Precision is a 77% with a Recall of 48%. Of the identified actual toxic comments, only 48% were accurately identified. The test set performed worse with a Precision of 74% and a Recall of 47%.

Naive-Bayes - Baseline

Given the difference in Precision and Recall between toxic and non-toxic for Logistic Regression in both the training and test reports, I'm going to see if using Naive-Bayes as my predictive model gives me a more accurate model.

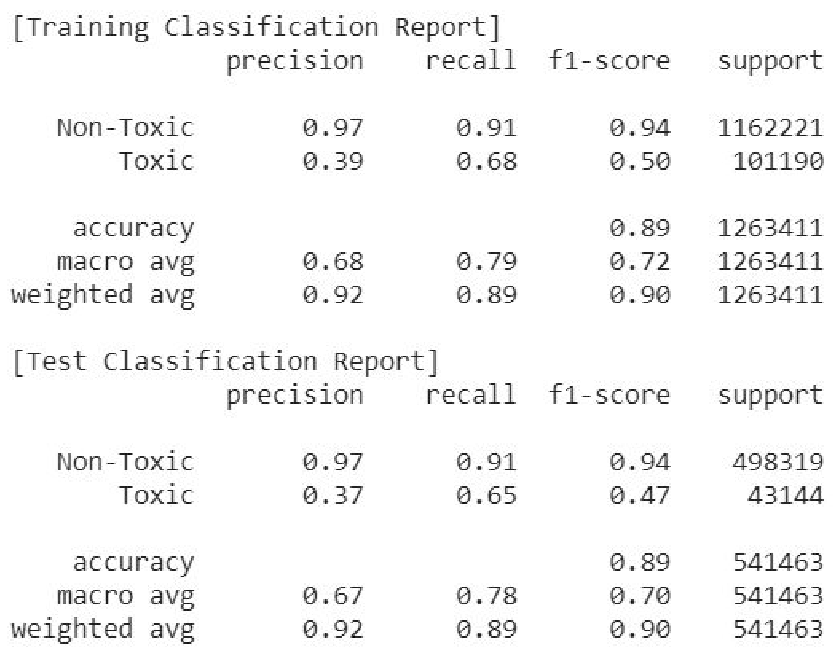
```
The training accuracy is 0.929560 and the test accuracy is 0.916508
```

The gap between training and test accuracy does not imply overfitting. However, we will still explore cross-validation and hyper-parameter tuning to generate the classification report and potentially a more accurate model right off the bat.

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

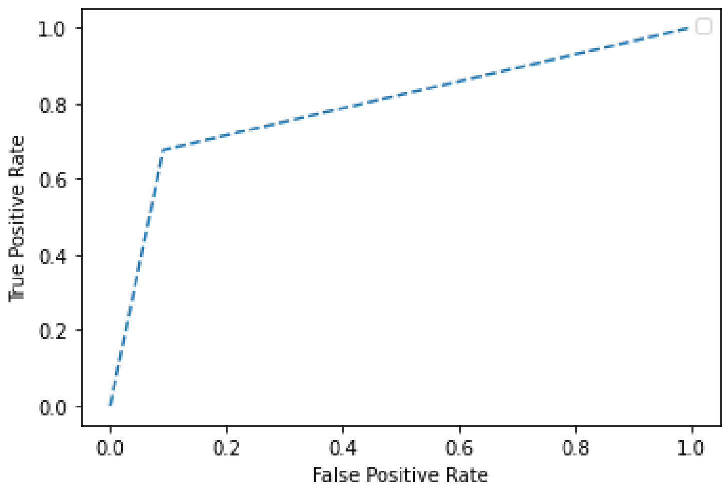
We only use the training data for hyper-parameter tuning. The test set is set aside and used to score the model at the end.

Looking at the results of the hypertuned Naive-Bayes, we see that for non-toxic both Precision and Recall, 0.97 and 0.91 respectively, are approximately the same between the train and test set. This means our model was accurate in identifying 97% of actual positives and mislabeling only 3%. Similarly, Precision is 91% showing that of the predicted positives, 9% were actually toxic.



```
[Training Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.97      0.91      0.94   1162221
       Toxic       0.39      0.68      0.50    101190

    accuracy                           0.89   1263411
   macro avg       0.68      0.79      0.72   1263411
weighted avg       0.92      0.89      0.90   1263411

[Test Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.97      0.91      0.94    498319
       Toxic       0.37      0.65      0.47     43144

    accuracy                           0.89    541463
   macro avg       0.67      0.78      0.70    541463
weighted avg       0.92      0.89      0.90    541463
```

However, in predicting toxic comments, the model didn't do as well which is similar to the results of the Logistic Regression. For Naive-Bayes, the trade off is flipped from the Logistic Regression. The train set's Precision is a 39% with a Recall of 68%. The train set mislabeled over 50% of the data it said was toxic. Of the identified actual toxic comments, Naive-Bayes identified 65% versus 48% for Logistic Regression. The test set performed worse with a Precision of 37% and a Recall of 65%.



A good measure of separability implying an excellent model is an AUC near 1. When AUC is 0.5, it means the model has no class separation capacity whatsoever. Given an AUC of 0.793, it means

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

there is ~80% chance that the model will be able to distinguish between positive class (toxic) and negative class (non-toxic).

Further Analysis

When thinking of model performance in regards to Precision and Recall, we must stop and consider which is more important in this instance with respect to our business problem. Precision means the percentage of your results which are relevant. On the other hand, Recall refers to the percentage of total relevant results correctly classified by your model. Think of it this way, what are the chances that you'll *Recall* all instances of something *precisely*? Would you rather Recall an action more times than it actually happened? Or would you want to remember actions precisely but not remember every instance? In the case of identifying toxic comments, there is a greater inclination to potentially over identify comments rather than miss something.

Interestingly enough, the Logistic Regression model has poor Recall while the Naive-Bayes model has poor Precision. The Logistic Regression classifier has an f1-score of .59 for toxic compared to the .5 f1-score of Naive-Bayes. As such, I'm going to focus on improving Recall. One way to try and improve Recall is by applying resampling techniques, combined with a classification algorithm. In particular, I will be using the Synthetic Minority Oversampling Technique (SMOTE) with my previously used classification algorithms - Logistic Regression and Naive-Bayes. In applying this method to my training data set, I'll be able to oversample my toxic label to potentially, train it better thus enabling a more accurate classifier.

The original 'y' training set had a split of non-toxic: 1,245,405 and toxic: 108,250. Comparatively, the new SMOTE y training set has a split of non-toxic: 498,162 and toxic: 249,081. The split went from 8% toxic to 33% toxic.

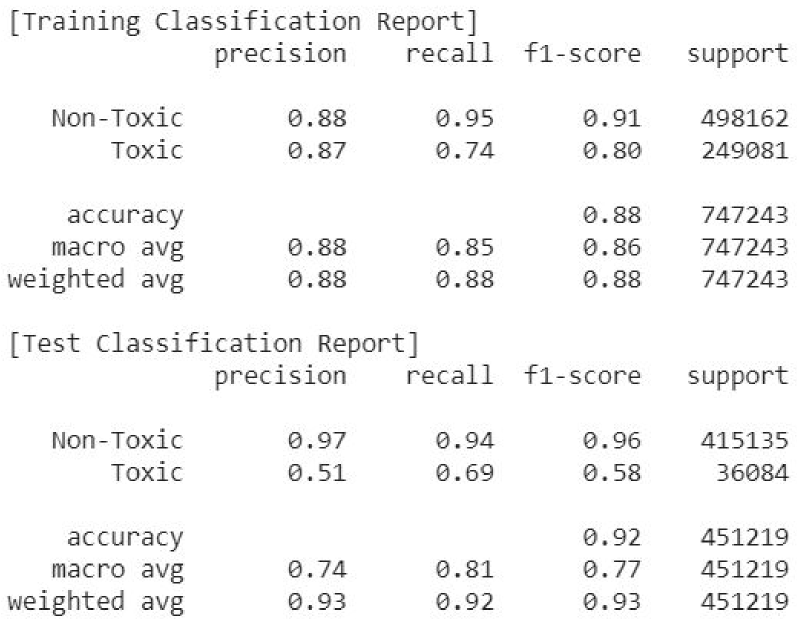
Logistic Regression + SMOTE

```
[Test] Accuracy score (y_predict_test, ytest): 0.9215680190772109

[Training] Accuracy score: (ytrain, y_predict_training)
0.8793297494924677
```

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Via SMOTE, my goal was to improve the Recall of my classifier. I ran my original test set through the updated Logistic Regression classifier and created a new classification report. In comparison to my training baseline, Precision increased from 0.76 to 0.87 and Recall increased from 0.48 to 0.74 with an f1-score of 0.8. On my test dataset, Precision decreased from 0.74 to 0.51. Recall increased from 0.47 to 0.69. My f1-score stayed the same at 0.58.

Typically, as Precision increases, Recall decreases and vice versa which we can see on our test dataset.

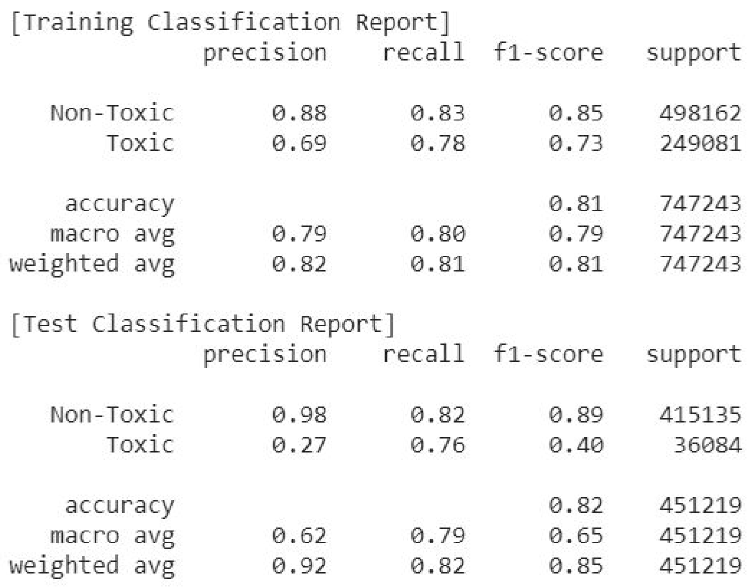


```
[Training Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.88      0.95      0.91    498162
       Toxic       0.87      0.74      0.80    249081

    accuracy                           0.88    747243
   macro avg       0.88      0.85      0.86    747243
weighted avg       0.88      0.88      0.88    747243

[Test Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.97      0.94      0.96    415135
       Toxic       0.51      0.69      0.58     36084

    accuracy                           0.92    451219
   macro avg       0.74      0.81      0.77    451219
weighted avg       0.93      0.92      0.93    451219
```

Naive-Bayes - SMOTE

The training accuracy is 0.809620 and the test accuracy is 0.816393.

The gap between training and test accuracy does not imply overfitting. However, one interesting point is the training accuracy decreased from 0.929560 and the test accuracy decreased from 0.916508.



```
[Training Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.88      0.83      0.85    498162
       Toxic       0.69      0.78      0.73    249081

    accuracy                           0.81    747243
   macro avg       0.79      0.80      0.79    747243
weighted avg       0.82      0.81      0.81    747243

[Test Classification Report]
              precision    recall  f1-score   support

   Non-Toxic       0.98      0.82      0.89    415135
       Toxic       0.27      0.76      0.40     36084

    accuracy                           0.82    451219
   macro avg       0.62      0.79      0.65    451219
weighted avg       0.92      0.82      0.85    451219
```

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Opposite to the Logistic Regression results, the non-toxic and toxic classification for both the train and test set decreased in Precision, Recall, and f1-score across the board. Applying SMOTE to my Naive-Bayes classifier made my model worse.

The baseline and SMOTE AUC are both 0.8 so applying SMOTE did not change. The implication being the model's ability to distinguish between positive class (toxic) and negative class (non-toxic) did not improve.

Wrap-Up

My hypothesis was that I believed my baseline classifier would be influenced by the imbalance that exists within my original dataset. In my baseline classifiers, there is a distinct difference between both classifiers' ability to identify the toxic label which supports my hypothesis. After applying SMOTE, my Logistic Regression classifier became more accurate in correctly classifying the toxic label. Not only did my Recall increase on my test set but both Precision and Recall increased on my training set.

In comparison, the Naive-Bayes classifier did not fare well post-SMOTE. My classifier became less accurate over all.

Based on the outcome of my baseline classifiers, I believe my baseline classifiers were influenced by the inherent imbalance within the data.

Recommendations for the Client

In choosing an algorithm for the intended client, I would recommend the Logistic Regression classifier due to the desire to correctly identify toxic comments. Even though this model has a lower Precision, it would be more prudent to catch as many toxic comments as possible, even at the expense of identifying some as toxic, which are not really so (a.k.a. false positives). The higher Recall would allow more comments to be captured even though some will be mis-identified. Thinking about the results in degrees of certainty, there would be a 69% certainty that toxic comments would be correctly identified.

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Future Work

- Regularization

  One item of interest to pursue further is the possible overfitting occuring within the model built using the training set. The benefit of addressing overfitting is the potential for better results during testing as the model would be less flexible. To do so, we would use the regularization parameter of the Logistic Regression model. Regularization significantly reduces the variance of the model, without substantial increase in its bias. The default of Logistic Regression is L2. Instead of using L2, we would use the L1 penalty to find the best 'C' in order to apply the level of regularization.

- **`TfidfVectorizer`**[2]

  Another item to pursue is using **`TfidfVectorizer`** with using bi-grams and removing stop words in place of **`CountVectorizer`**. Instead of every word counted as a single token in the matrix of word counts, bi-grams would be groups of two words. For example, 'I saw Sam' would be captured as 'I saw', 'saw Sam' instead of 'I', 'saw', 'Sam'. This can lend additional context. With **`TfidfVectorizer`**, the inverse document frequency adjusts for the fact that some words appear more frequently in general. Without the inverse document frequency part, common words such as "we" would have higher weights than words that are uniquely common in a document "bisexual" or "christian" which arguably tell us more about the document. The benefits of using a combination of **`TfidfVectorizer`** with bi-grams is a greater depth of context and the possibility of greater model accuracy.

- Undersampling combined with SMOTE

  Previously we used SMOTE by itself to try and increase Recall, a different approach would be using undersampling combined with SMOTE. Undersampling removes records from the training dataset that belong to the majority class (non-toxic) in order to better balance the class distribution. When combined with SMOTE, we would undersample the majority class and oversample (SMOTE) the minority class.

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

In applying this combination, we could potentially have greater model accuracy by reducing the impact of the majority class while increasing the number of records to better train the minority class.

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html