Capstone 2 Report

Toxic Comment Challenge

By Minh Ngoc Pham
Data Science Career Track

Disclaimer: the dataset and this project for this competition contains text that may be considered profane, vulgar, or offensive.

I. Problem Statement

There is an increasingly high number of toxic behaviours and comments over the internet which are making it difficult to have meaningful discussion on the net. As the world is becoming more and more technologically involved and people demonstrate more presence on the internet compared to in real life, it is important to make it a healthy and safe place to express opinions.

One does not need to look far but at Youtube, where presence of toxic and hatred comments in ubiquitous in almost any video. This has led many Youtube channels to disable the comment section, which in itself might be a bit of an extreme measure given the fact that genuine commentators might be interested in the content and want to engage in meaningful discussion. The need for a mechanism to deal with outright toxic and unhealthy comments are more than ever relevant in this day and age.

II. The Dataset

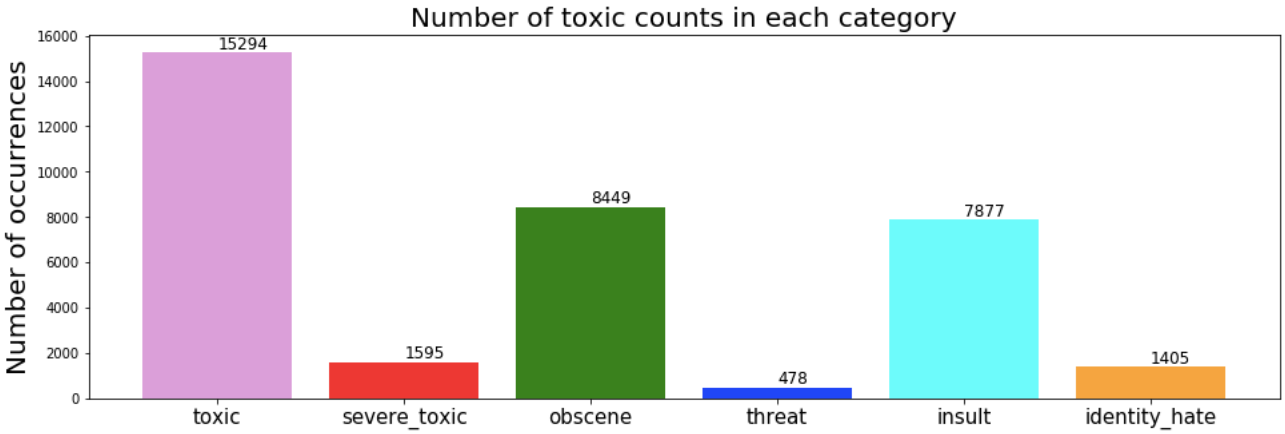
Dataset: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/

The dataset chosen for addressing this issues is the Toxic Comment Classification Challenge by Kaggle that was published in March 2018. Conversation AI is a research initiative by Jigsaw and Google aiming at improving online conversation. They have created a lot of publicly available models but there are still errors involved. This competition challenged other to find a model that can better detect negative online behaviour.

This dataset contains Wikipedia comments that have been rated for toxic behaviour. There are 6 classes for 6 types of toxic behaviour including: toxic, severe_toxic, obscene, threat, insult, identity_hate. There are 159,571 comments in the training set (which also includes comments that are rated as neutral, meaning being scored 0 for all classes). The test set include 63,978 comments.

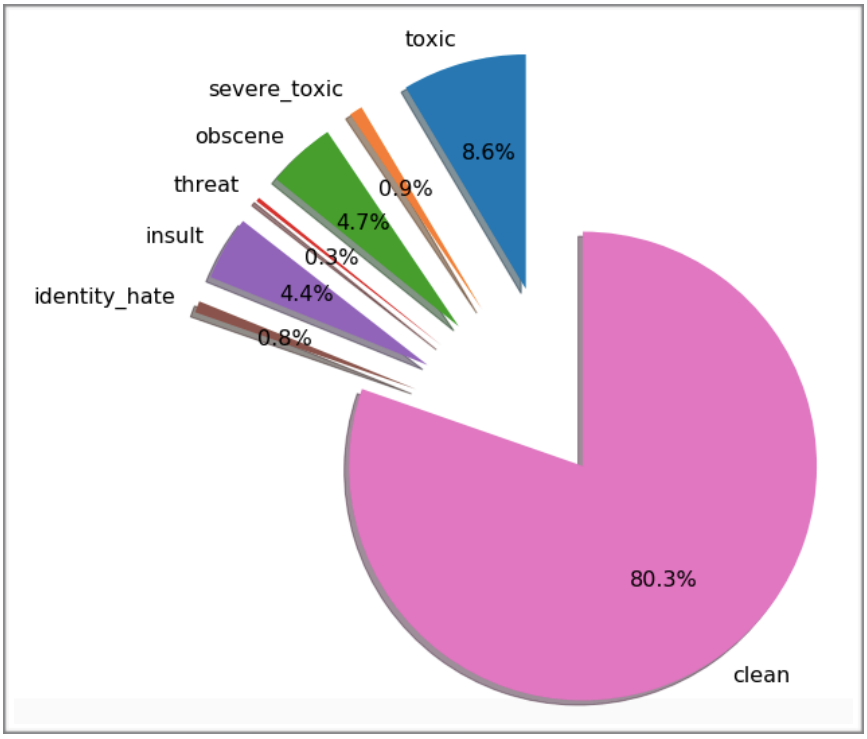
III. Initial Data Visualisation

We will first have a look at our dataset to see how many comments being classified as a certain label:



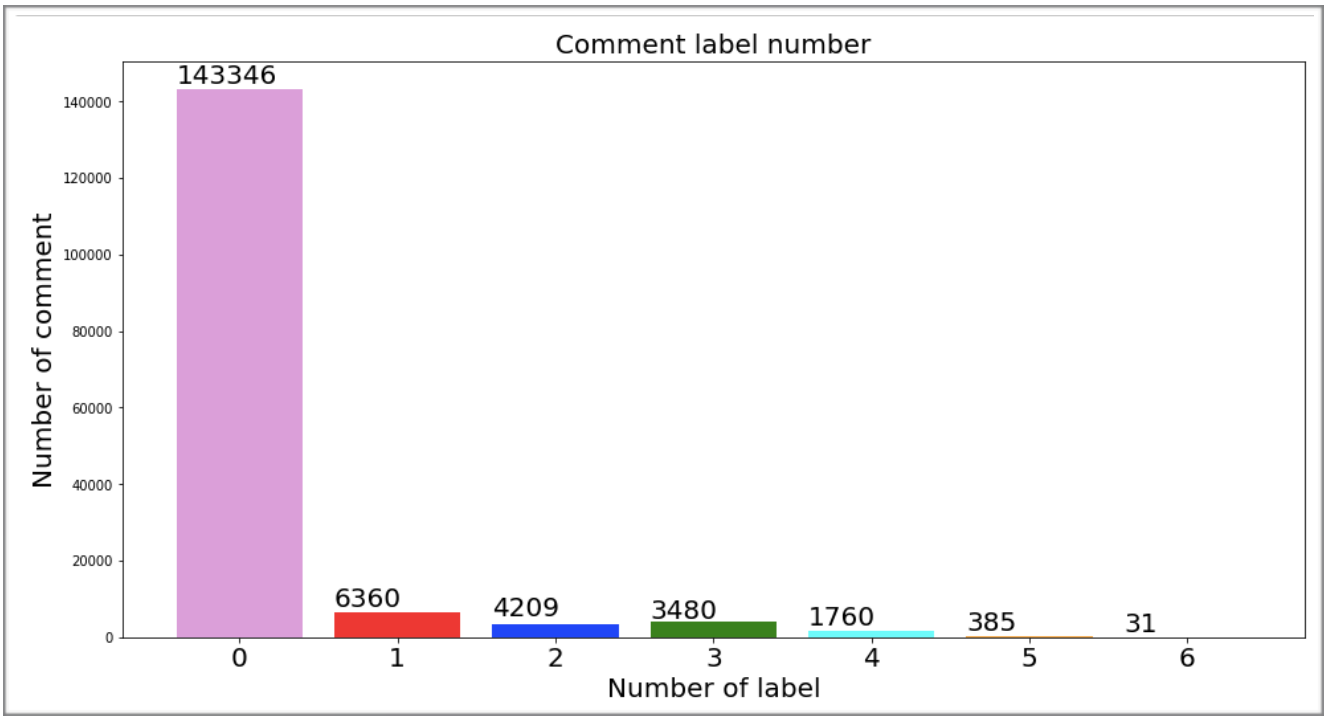
We can see from the above graph that "toxic" category has the highest number of comments with 15,294 comments classified as toxic. The "threat" class has the lowest number of comments with only 478 comments in this class. We notice that the whole training dataset contains 159,571 comments in total. With this in mind, the class with the highest number of comments ('toxic") account for even less than 10% of all the comments. This suggests a class imbalance issue with a very high number of clean comment.

To further shed light on this issue, we will create a new column classifying a comment as clean or not.



The pie chart shows that we have about 80.3% of the comments that do not belong to any of the 6 classes.

We might face a class imbalance situation in this case. As a result, accuracy score might not be the most meaningful determiner of how good a model is. We might need to consider: ***confusion matrix, precision, f1 score and recall.***



Checking the number of labels per comment

Each comment can be classified as more than one label. They can have many label classification simultaneously with the highest number of label of 6 and the lowest of 0.

The bar chart shows the count of comments with certain number of labels:

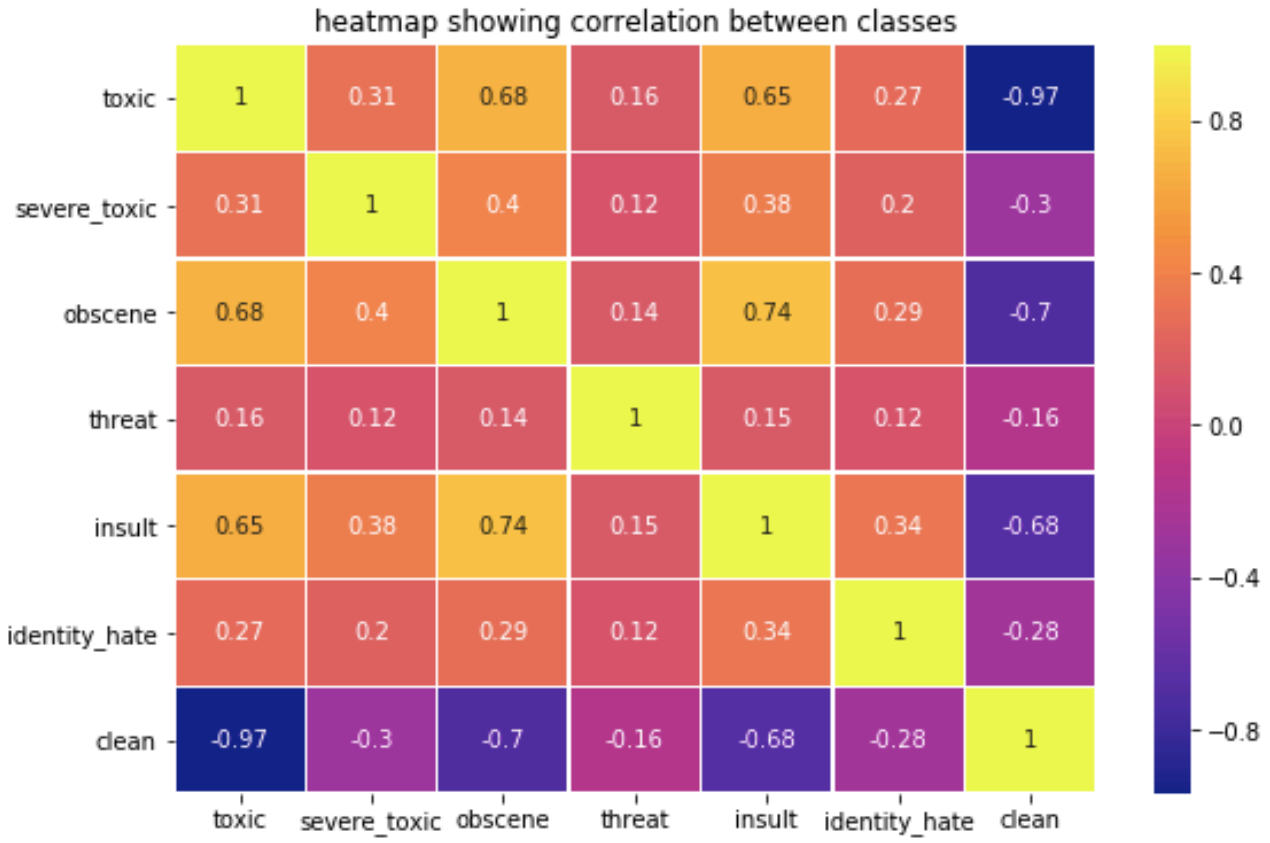
As expected, we see a high number of comment with 0 label ('clean' comment) of 143,346 comments. It is then followed by the count of comment with 1 label (6,360 comments). There are only 31 comments that are classified as all 6 of the labels.

```
Checking for null values in the Train set
comment_text      0
toxic             0
severe_toxic      0
obscene           0
threat            0
insult            0
identity_hate     0
dtype: int64
Checking for null value in the Test set
comment_text    0
dtype: int64
```

Data Cleaning and Wrangling

We will have a look at the train data and the test data to see if there is any null value:
As expected, there is no null values in any of the set. As this is a Kaggle competition dataset. The data was probably already clean.

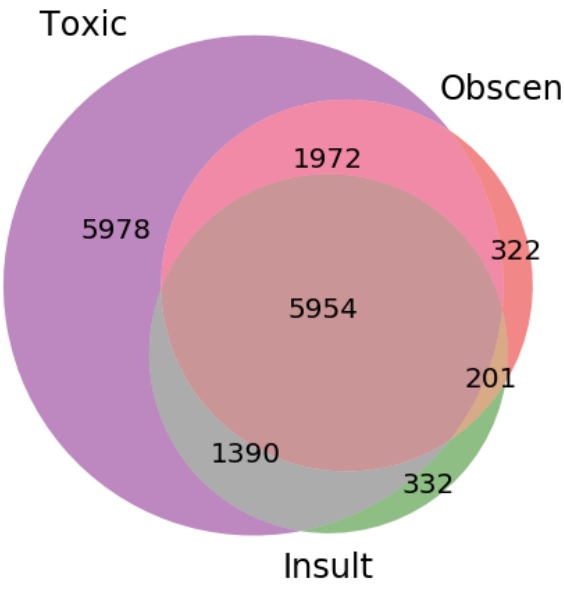
✦ There is no null value in either the Train set or the Test set



Testing for correlation between feature variables

As we can see, the 7 feature variables (including clean) are dichotomous categorical variables (with values as either 0 or 1) so we can apply the Pearson correlation (also known as point-biserial correlation coefficient) in this case. We will do a correlation heatmap in order to look at the correlation between these variables.

<u>Code notice:</u> the .corr() formula simply ignore the comment_text variable as it is a non-numerical variable



✦There is high correlation between clean and toxic (negatively correlated) suggesting that the comments are likely to be classified as either toxic or clean

✦obscene and insult categories also have high correlation with toxic (0.68 and 0.65 respectively). Perhaps we can explore further with a Venn diagram to see

It is interesting to note here that:

- The majority of comments classified as obscene are also classified as toxic (1972 + 5954 = 7926 comments out of 8449 obscene comments which is roughly 93.8%)
- and the majority of comments classified as insult are also classified as toxic (5954+1390=7344 comments out of the 7877 insult comments which is roughly 93.2%)
- 5954 comments are classified as obscene, insult and toxic

Text Cleaning and Word Clouds

An initial look at the comment_text columns show that there are some characters in the text that can be cleaned and preprocessed. The following steps were taken to clean the texts:

- Combining the train data and the test data for cleaning (using pd.concat())
- Removing all punctuation
- Removing the \n in the comments
- removing the digits in the comments
- Splitting combined words
- Converting words into lowercase
- tokenize comments
- Setting and removing stopwords (set(stopwords.words('english'))
- Converting words in base form/lemmatize

- Split the data back in clean training data and clean test data

Once the text is nice, clean and tokenise. We can perform Word Cloud to have a look the words that appear a lot in certain labels.

The bigger the words in the Word Cloud, the more frequent it appears in the comment text. Notice some of the words with high frequency in some of the toxic labels are as follows:

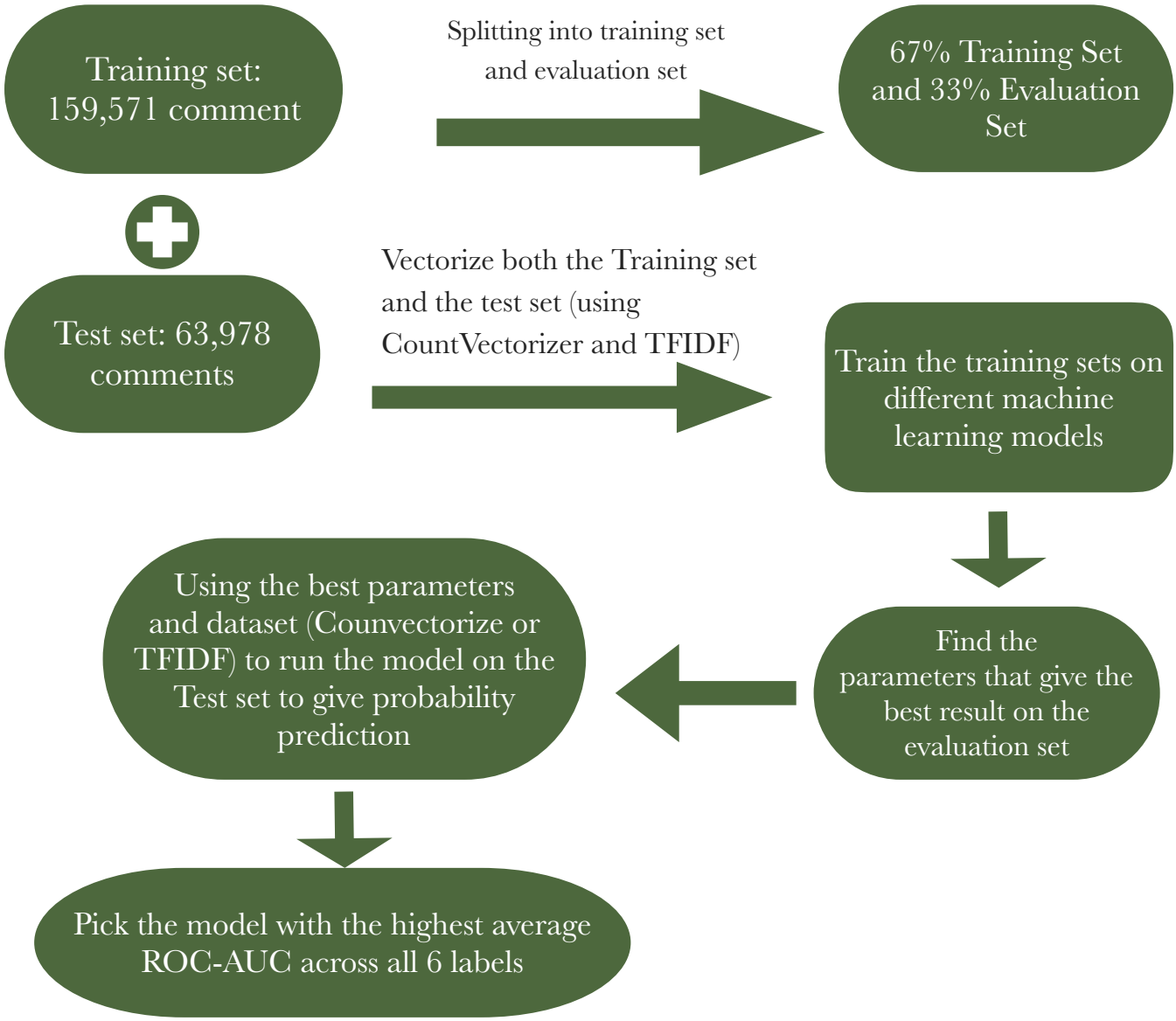
- Toxic: *fuck, suck, shit, don't like, wikipedia, faggot, jew*
- Severe toxic: *fuck, shit, as* (which is quite strange to show up here), *bitch, fucking, faggot*
- Obscene: *fuck, shit, suck, nigger, cunt, fucking*
- Threat: *die, block, as, kill, doing Jim*
- Insult: *fuck, suck, shit, faggot, nigger, shit, bitch*
- Identity_hate: *nigger, faggot, gay, jew, fat, fuck*

All these words are very derogatory in nature and it corresponds quite accurately with the nature of toxicity for example, identity hate words include those calling out on people's identity and threatening toxicity includes words with negative actions.

IV. Modelling

Approach: this is originally a Kaggle competition dataset. However, the deadline for the competition has passed and so the result of the test set is posted. I still decided to, however, approach the problem as a competition since computationally, it is not possible to carry out 5 machine learning models' GridSearchCV and so splitting the training dataset in to training set and evaluation set first to compute the ROC-AUC before testing it on the test dataset can help us detect overfitting issues if the model does not do well in the test set.

Flow Chart of Problem Approach



Evaluation metrics: As we notice from the EDA. This dataset is very imbalanced with 80.3% of the data clean and the remaining classified in one or more of the labels. That means if we predict everything as clean we have a 80.3% of accuracy rate. In that instance, it is more appropriate to use ROC-AUC score as evaluation metric. This value is averaged out across 6 labels to find the best model.

Vectorize the data: using both TfidfVectorizer() and CountVectorizer() to see which data performs better.

Tuning the model: Since the dataset is too large to carry out GridSearchCV, we will implement testing the model on different parameters to see which parameters performs best.

Model 1: Logistic Regression

	OneVsRestClassifier (with C=1, estimator penalty =l1)- TFIDF	ClassifierChain ((with C=1, estimator penalty =l1) - TFIDF	OneVsRestClassifier (with C=1, estimator penalty =l2) - TFIDF	OneVsRestClassifier (with C=1, estimator penalty =l2) - CountVectorizer data
Toxic	0.968258	0.968258	-	0.799235
Severe Toxic	0.984863	0.980751	-	0.765841
Obscene	0.981258	0.970786	-	0.778556
Threat	0.979104	0.965633	-	0.619170
Insult	0.971562	0.947265	-	0.774192
Identity Hate	0.970317	0.955580	-	0.671779
Average	0.975894	0.964712	0.977498	0.734796

As we can see from the graph above, the optimised model is the one using OneVsRestCassifier with C=1 and penalty = l2 (we got this result using GridSearchCV). The model does not perform very well with CounVectorizer dataset with only ROC-AUC score of 0.73.

We can also try to train the model using TFIDF vectoriser for both characters and words to see how the model train and examine the result on the test data.

	TFIDF Vectorizer data for only 'words'	TFIDF Vectorizer data for both 'word' and 'char'
Result on evaluation dataset	0.9759	0.9831
Result on Test dataset	0.9729	0.979

Model 2: Naive Bayes

The 2nd model we will train the dataset on is the Naive Bayes model.

	MultinomialNB() with alpha = 0.1	MultinomialNB() with alpha = 1	MultinomialNB() with alpha = 5	MultinomialNB() with alpha = 10	MultinomialNB() with alpha = 50
Average ROC-AUC	0.943539	0.853696	0.781917	0.762324	0.737935

	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity Hate	Average
With TFIDF dataset	0.954660	0.973322	0.958055	0.912945	0.957876	0.933435	**0.948382**
With CV dataset	0.917226	0.929333	0.919417	0.848131	0.916570	0.864621	**0.899216**

As we can see from the table above, the model with TFIDF dataset and alpha=0.1 performs the best and the result on the test set is: 0.9364

Model 3: Decision Tree Classifier

	max_depth=10, criterion = 'gini' - TFIDF dataset	max_depth=10, criterion='entropy' - TFIDF dataset	OneVsRestClassifier, max_depth=10,criterion='entropy' - CountVectorize dataset	ClassifierChain, max_depth=10,criterion='entropy' - CountVectorize dataset	OneVsRestClassifier, max_depth=15,criterion='entropy' - CountVectorize dataset
Toxic	0.740011	0.841011	0.788924	0.788485	0.827153
Severe Toxic	0.805550	0.736647	0.800206	0.823473	0.684435
Obscene	0.841200	0.858569	0.862610	0.830148	0.858615
Threat	0.614732	0.703729	0.719132	0.558249	0.680063
Insult	0.766586	0.843947	0.836273	0.836210	0.822210
Identity Hate	0.747754	0.769022	0.780570	0.659727	0.777247
Average	0.752639	0.792154	0.780570	0.749382	0.774954

Based on the running of different model parameters, we find that Decision Tree Classifier seems to do best with OneVsRestClassifier, max_depth=10, criterion='entropy' and it performs better on CountVectorizer dataset compared with TFIDF dataset.

Prediction on Test set: 0.7941

Model 4: Random Forest Classifier

	OneVsRestClassifier, n_estimators = 100, max_depth=15 - TFIDF dataset	ClassifierChain, n_estimators = 100, max_depth=15 - TFIDF dataset	OneVsRestClassifier, n_estimators = 100, max_depth=15 - CV dataset	OneVsRestClassifier, n_estimators = 1000, max_depth=15 - TFIDF dataset	OneVsRestClassifier, n_estimators = 1000, max_depth=10 - TFIDF dataset
Toxic	0.933500	0.933731	0.920494	0.936943	0.925093
Severe Toxic	0.979556	0.977144	0.973829	0.982700	0.981119
Obscene	0.976782	0.974873	0.963179	0.979460	0.974446
Threat	0.921531	0.928160	0.928774	0.951529	0.950142
Insult	0.960064	0.959790	0.943864	0.963969	0.957683
Identity Hate	0.946986	0.948915	0.934240	0.961600	0.956791
Average	0.953070	0.953769	0.944063	0.962700	0.957546

Prediction on Test set: 0.9668

V. Conclusion

All four models perform relatively well in predicting the labels for the toxic comments.

	Logistic Regression	Naive Bayes	Decision Tree Classifier	Random Forest Classifier
Average ROC-AUC	0.979	0.948382	0.7941	0.9668

Online community is no longer what it used to be and toxic comments disrupt the healthy discussion that can exist and drive serious users away. Social media platforms such as instagram and facebook and among others can certainly make use of a more effective way of filtering toxic comments to ensure a better online experience for all.
While Instagram has setting for users to manually filter a set of keywords, it should not be up to the users to have to ensure they are not exposed to a toxic online environment.