

**Malignant Comment Classification**

Submitted by:

S.Praveena

**INTRODUCTION**

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but “u are an idiot” is clearly offensive.

Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

**Analytical Problem Framing**

**Remove excessive length comments**

Some very large length comments can be seen, in our dataset. These pose serious problems like adding excessively more words to the training dataset, causing training time to increase and accuracy to decrease!  
Hence, a threshold of 400 characters will be created and only comments which have length smaller than 400 will be used further.

# Preprocessing

Preprocessing involved the following steps, but these will be performed in a slightly different manner:

* Removing Punctuations and other special characters
* Splitting the comments into individual words
* Removing Stop Words
* Stemming and Lemmatising
* Applying Count Vectoriser
* Splitting dataset into Training and Testing

## Updating the list of stop words

**Stop words** are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on our statement.E.g. is, this, us,etc.  
Single letter words if existing or created due to any preprocessing step do not convey any useful meaning and hence can be directly removed. Hence letters from b to z, will be added to the list of stop words imported directly.

## Stemming and Lemmatizing

**Stemming** is the process of converting inflected/derived words to their word stem or the root form. Basically, a large number of similar origin words are converted to the same word.E.g. words like "stems", "stemmer", "stemming", "stemmed" as based on "stem". This helps in achieving the training process with a better accuracy.  
**Lemmatizing** is the process of grouping together the inflected forms of a word so they can be analysed as a single item. This is quite similar to stemming in its working but differs since it depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.  
The **wordnet library in nltk** will be used for this purpose. Stemmer and Lemmatizer are also imported from nltk.

**Splitting dataset into training and testing**

* Since the system was going out of memory using train\_test\_split, I had jumbled all the indexes in the beginning itself.
* The shuffle function defined here performs the task of assigning first 2/3rd values to train and remaining 1/3rd values to the test set.

**Model/s Development and Evaluation**

### Binary Relevance (BR) Method with MultinomialNB classifiers

**These include the Binary Relevance, Label Powerset and Classifier Chain methods. Implementations of these methods is available in the scikit-multilearn library.**

* I will be implementing the most basic method,which is the **Binary Relevance** method from scratch. It does not take into account the interdependence of labels and basically creates a separate classifier for each of the labels.
* Scikit-multilearn library's classifier will also be imported and tested with different classifiers to observe if it gives similar results.

### BR Method with SVM classifier

### BR Method with GausseanNB classifier

**CONCLUSION**

* While showing among the best problem transformation method models, hamming-loss was considered (this is because for BP-MLL neural network we had to round the final results to get the hamming-loss because of the output being multivalued probabilities)
* But while choosing among the best Adaptation Algorithm model, log loss was preferred.