

# Why So Harsh?

## ML PROJECT REPORT

@Aimers (Ranked 1st - in public score and 2nd - in private score)

**Team Name - @Aimers**



### Team Members :

Akshay Thite	MT2021010	Akshay.Thite@iiitb.ac.in
Gaurav Tirodkar	MT2021047	gaurav.tirodkar@iiitb.ac.in
Niraj Gujarathi	MT2021087	Niraj.Gujarathi@iiitb.ac.in

## Introduction :

The problem statement aimed at analysing tweets and predicting whether or not it was a toxic comment. And categorizing based on the severity of the content.

### Task :

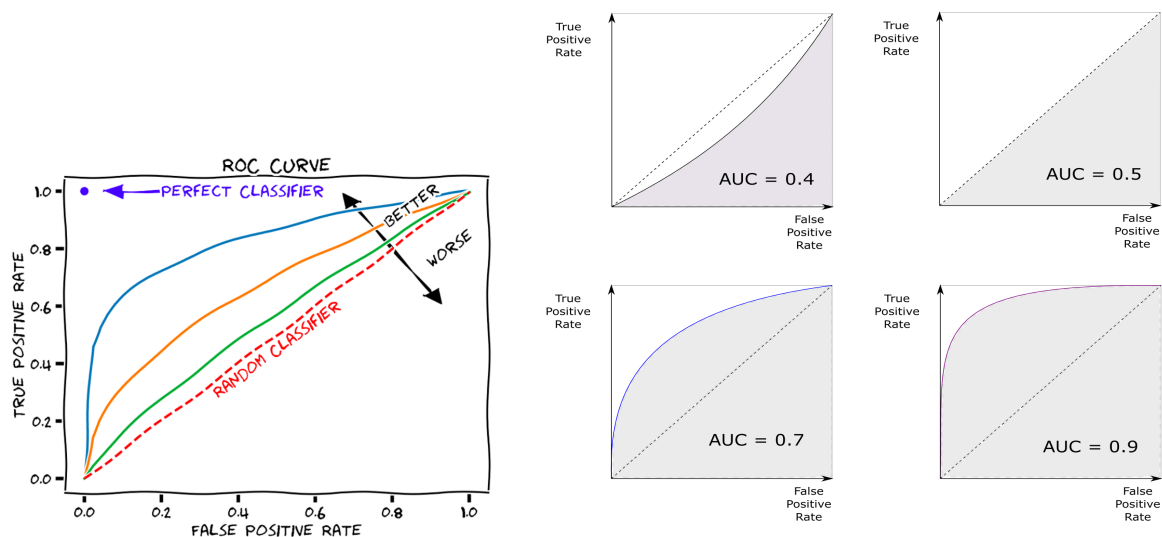
Given a comment, classify it into “harsh”, “extremely harsh”, “vulgar”, “threatening”, “disrespect”, “targeted hate”. For example, consider a comment, “You useless piece of trash! you are such an asshole that you deserve to rot in the gutter alongside sewage.” That is certainly harsh! So, the label for this would be [1 1 0 0 1]. Translating to the comment being harsh, extremely harsh, disrespectful, and targeted hate.

### Evaluation Metric :

Kaggle submissions are evaluated on the **Area Under the ROC curve (AUC)** -

ROC means receiver operating characteristic curve - it is a graph showing the performance of a classification model at all classification thresholds.

AUC - provides an aggregate measure of performance across all possible classification thresholds.



Score less than 0.5 is considered worse and as it approaches 1 we will get the perfect classifier model.

Final kaggle score is computed between the predicted probability and the observed target value. A mean is calculated for each of the classes.

## Dataset :

The dataset contains the text associated with a particular comment and the corresponding labels for each of the classes.

**The classes are :** harsh, extremely\_harsh, vulgar, threatening, disrespect, targeted\_hate

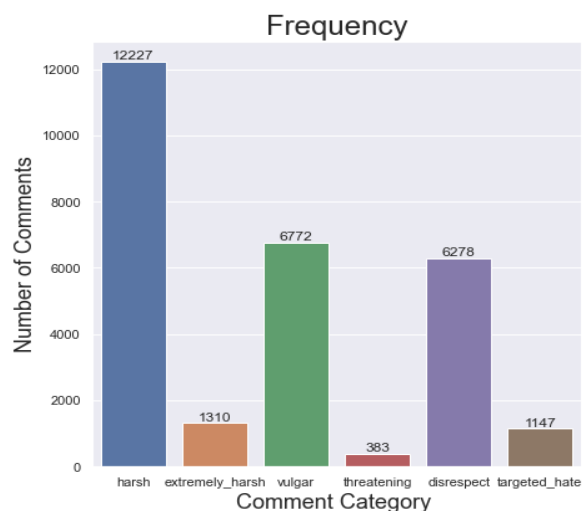
Total Columns in Dataset :

- id - id
- text - The content of the comment.
- harsh - if the comment was harsh or not
- extremely\_harsh - if the comment was extremely harsh or not
- vulgar - if the comment was vulgar or not
- threatening - if the comment was threatening or not
- disrespect - if the comment was disrespectful or not
- targeted\_hate - if the comment was targeted hate or not

## Data Analysis :

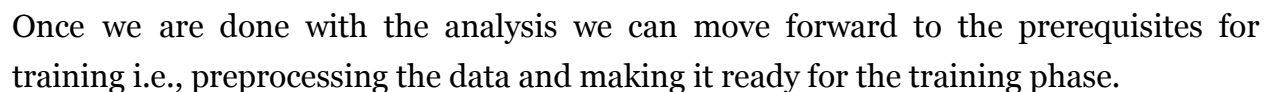
The data provided had 127656 tuples and 8 columns, 6 were output labels.

Digging a little deeper, we checked the data distribution for each of the labels.



As we can see the data is majorly covered by harsh comments, though the other columns have values in thousands they are less as compared to the “*harsh*” section.

WordCloud gives an image of words in the text, the size of each word depends on the frequency of occurrences. We ran this code through our model and not to our surprise we found abusive words written in bold and huge fonts, indicating the number of harsh comments were going to be high.



## Preprocessing :

The biggest difference between a good ML model and the best is the quality of data. Given a well cleaned and preprocessed data, even a simple algorithm can generate good accuracy.

With this in mind we cleaned our data as follows :

### Filtering slang - keeping English Alphabets :

A lot of tweets contained slang and short forms of words, so after going through the dataset we replaced words like “what’s” with “what is”, “can’t” with “cannot” and so on. Removing non english characters.

### Removing urls / HTML elements (clean HTML):

Tweets had http links to different websites, and we needed to filter them as it added no sense. Using regular expressions we cleaned this section of our data.

### Tweet Tokenizer -removing emoticons and Punctuations :

As we are dealing with tweets data, these tuples have a lot of emoticons depicting feelings stronger than words. Using tweet tokenizers we try to make sense of these emoticons as well.

### Final Data after preprocessing :

	id	text	harsh	extremely_harsh	vulgar	threatening	disrespect	targeted_hate
0	52e0f91a5d7b74552c55	new main picture main picture sadly dont...	0	0	0	0	0	0
1	e2c8e370a8e53ba26bae	think th like population charts every year...	0	0	0	0	0	0
2	03c807f61149a13c8404	page se little misleading reason midpoi...	0	0	0	0	0	0
3	fc63a1ba3372899db19f	actually accounts never deleted	0	0	0	0	0	0
4	0c2bfd9cde8974d9915f	yeah yeah ok still meant know nothing ...	0	0	0	0	0	0

### Principal Component Analysis (PCA) :

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss, by creating new uncorrelated variables that successively maximize variance.

## POS Tagging :

It is a process of converting a sentence to forms – list of words, list of tuples (where each tuple is having a form (word, tag)).

Applied both PCA and POS tagging on text data, didn't observe significant difference in accuracy.

## Vectorizer - (TF IDF) :

### Word / Char vectorizer:

A model cannot train on words but can only deal with numbers, so we convert our text into numbers using a vectorizer as belows.

### n- grams :

Given a sequence of n-1 words, an n-gram model predicts which is the most probable next word. ( n\_grams in characters and word vectorizers have improved accuracy )

```
word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    analyzer='word',
    token_pattern=r'\w{1,}',
    ngram_range=(1, 1),
    max_features=20000)
char_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    analyzer='char',
    ngram_range=(2, 6),
    max_features=30000)
```

**TF-IDF (term frequency-inverse document frequency)** is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

## Approaches :

- **Models :**
  - Logistic Regression
  - Naive Bayes
  - SVM
  - Random Forest
  - Ridge classifier

- **Training approach :**
  - One vs Rest
  - Binary Relevance
  - Classifier chains
  - Label Powerset

## **Models :**

### **1. Logistic Regression**

- It is a statistical model which uses logistic function to model a binary dependent variable.
- SAG solver is used in cases of multiclass problems to handle multinomial loss.
- The N gram model will be used to form the sequence of N-1 words and assign them probabilities.

### **2. Naive Bayes**

- Classification technique based on Bayes Theorem which assumes independence between the predictors.
- It assumes that the presence of one feature in the class is independent of other features.

### **3. SVM**

- It is a supervised learning algorithm.
- It plots a hyperplane to classify the data points in the data set and on the basis of this hyperplane the classification will be done.

### **4. Random Forest**

- It consists of a large number of decision trees which operate as an ensemble.
- Each independent tree gives a class prediction and the class with the most votes will be the prediction of the model.

### **5. Ridge**

- It uses the Ridge regression model for building the classifier.
- If we consider the binary classification then we split the data into +1 and -1 classes. Now, if the predicted value of the Ridge regression is positive then it belongs to the +1 class else it will belong to the -1 class.

## Training Approaches :

### 1. OneVsRest

- Decomposing the multilabel classification problem into multiple independent binary classification problems. Here each binary classification problem corresponds to a category.
- Approach is to build multiple independent classifiers and for an unseen instance choosing the class for which the confidence is maximized.
- Here, we assume that the labels are mutually exclusive.

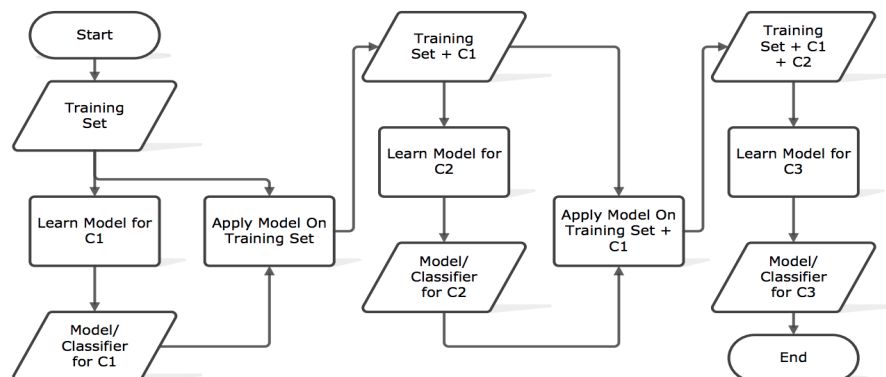
### 2. Binary Relevance

- Ensemble of single label binary classifiers is trained for every class. Each classifier predicts the membership of that class. Union of all predicted outputs is taken as final output of the multilabel classifier. It is easy to implement but does not consider the correlation between the labels.
- If there are 'n' labels then 'n' new datasets will be created and each of these would be used to train the single label classifiers.

### 3. Classifier Chains

- The number of classifiers used will be equal to the number of labels.
- For the prediction of one label the binary classifier will also consider the predictions of previous labels. In this way, the correlation between the labels will be considered.

Given below is the flow chart of classifier chains approach where C1, C2, C3 are the labels.





#### 4. Label Powerset

- Here, each member of the powerset of labels in the training set will be considered as a single label. So, the correlation between the labels is considered.
- In worst case,  $(2^{|C|})$  classifiers are required. It has a high computational complexity.
- As the number of classes increases, the number of distinct label combinations can grow exponentially. In this case, the computation will not be feasible.

### Accuracy Scores :

Approaches	Validation Accuracy	Kaggle Submission (ROC Score)
<b>Approach 1-</b> <b>SAG solver for Logistic Regression and n_grams</b>	<b>98.23%</b>	<b>98.642%</b>
Approach 2 - Ridge classifier for n_grams	97.9%	98.482%
Approach 3 - Liblinear solver for Logistic Regression simple word vectorizer	99.1%	96.664%

Approach 1 - validation scores using one vs rest training approach


```
Processing : harsh.  
Training accuracy score is 0.9750728422547064  
  
Processing : extremely_harsh.  
Training accuracy score is 0.9872125417574211  
  
Processing : vulgar.  
Training accuracy score is 0.9883380685678772  
  
Processing : threatening.  
Training accuracy score is 0.9833535513452584
```

```
Processing : disrespect.  
Training accuracy score is 0.9811133876425465  
  
Processing : targeted_hate.  
Training accuracy score is 0.9790612965902931  
  
Overall cross-val score is 0.9823586146930171
```


## Results :

After cleaning the data with multiple preprocessing techniques and training a variety of models, we reached the best one that gave **98.6 %** accuracy on unseen data.

### Public Score

#	Team Name	Notebook	Team Members	Score ?
1	aimers			0.98642

### Private Score

2	▼1 aimers			0.98542	18	4d
---	-----------	--	--	---------	----	----