**Toxic Comment Prediction Case Study**

**1. INTRODUCTION** -Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to evidently facilitate conversations, leading many communities to limit or completely shut down user comments.

**2. MOTIVATION** -So far we have a range of publicly available models served through the Perspective API, including toxicity. But the current models still make errors, and they don't allow users to select which type of toxicity they're interested in finding. (E.g. some platforms may be ne with profanity, but not with other types of toxic content)

**3. PROBLEM STATEMENT** -Building a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insult and identity-based hate.

**4. DATASET** -The dataset used was Wikipedia corpus dataset which was rated by human raters for toxicity. The corpus contains comments from discussions relating to use pages and articles dating from 2004-2015.

**5. DATA OVERVIEW**- The dataset used was Wikipedia corpus dataset which was rated by human raters for toxicity. The corpus contains comments from discussions relating to use pages and articles dating from 2004-2015. The dataset was hosted on Kaggle. The comments were manually classified into following categories:

Toxic

Severe toxic

Obscene

Threat

Insult

Identity hate

**6. APPROACH**

**6.1 HOW PROBABILITY WAS CALCULATED**? Though there many multi class classifiers, we do not have a suitable multi label classier which was able to give probability with which target belongs to a label. So, we used scikit-learn OneVsRestClassier with various estimators, with the help of predict\_proba, we predicted the probability with which a comment belongs to a particular label.

**7. ANALYSIS OF DATASET**

**7.1 VISUALIZATION**

**7.1.1 COUNT THE NUMBER OF COMMENTS IN EACH CLASS** -Three major labels are: toxic obscene insult **7.1.2 Pie chart of Label**- Distribution over comments (without “none” category).

**7.1.3 COUNT FOR EACH LABEL COMBINATION-** Now, let's take a look at number of comment for each label combination. This helps us in finding correlation between categories. Following can be inferred from above table: The table shows that number of comments with only none label are high. Toxic which is the label high after none is present in all top 6 combinations. Among the top combinations, obscene and insult comes 4 times in 6. As the combinations increase the count drops very fast.

**7.1.4 CORRELATION MATRIX-** Following can be inferred from above matrix: Toxic is highly correlated with obscene and insult. Insult and obscene have highest correlation factor of 0.74 Interesting things to be observed: Though, a severe toxic comment is also a Toxic comment, the correlation between them is only 0.31.

**8. FEATURE ENGINEERING**

**8.1 CLEANING THE COMMENTS** - Since, the comments in the dataset were collected from the internet they may contain 'HTML' elements in them. So, we removed the HTML We then converted each comment into lower case and then split it into individual words. There were some words in the dataset which had length > 100, since there are no words in the English language whose length > 100, we remove such words. First, we tried building the features removing stop words and then trained some models thinking that it may help the model in learning the semantics of toxicity, but we found out that the model learns better if there are stop words in the comment. Possible reason is, generally a hate/toxic comment is used towards a person, seeing the data we found out that those persons are generally referred by pronouns, which are nothing but stop words.

**8.2 STEMMERS AND LEMMATIZERS** –

1. Definitions: Stemming usually refers to a crude heuristic process that chops o the ends of words in the hope of achieving this goal correctly most of the time, and often includes removal of derivational affixes. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove endings only and to return the base or dictionary form of a word, which is known as the lemma. International Journal of Trend in Scientific Research and Development .

2 Reasons to use: We used both snowball stemmer, porter stemmer and WordNet lemmatizer. For grammatical reasons, documents are going to use different forms of a word, such as organizes, organize and organizing. But they all represent thesame semantics. So, using stemmer/Lemmatizer for those three words gives a single word, which helps algorithm learn better.

3. Results: On public dataset: Snowball > WordNet > Porter On private dataset: WordNet > Snowball > Porter Decreasing order of accuracy.

**8.3 VECTORIZATION**- Python's scikit-learn deals with numeric data only. To convert the text data into numerical form, tf-idf vectorizer is used. TF-IDF vectorizer converts a collection of raw documents to a matrix of Tf-idf features. We set the predictor variable on the dataset with tf-idf vectorizer, in two different ways. First, by setting the parameter analyzer as 'word'(select words) and the second by setting it to 'char'(select characters). Using 'char' was important because the data had many 'foreign languages' and they were di cult to deal with by considering only the 'word' analyzer. We set the parameter n-gram range (an n-gram is a continuous sequence of n-items from a given sample of text or speech). After trying various values, we set the n-gram as (1, 1) for 'word' analyzer and (1, 4) for 'char' analyzer. We also set the max\_features as 30000 for both word and char analyzer after many trails. We then combined the word and character features and transformed the dataset into two sparse matrixes for train and test sets, respectively using tf-idf vectorizer.

**8.4 ADDING DATA RELATED FEATURES** -We tried adding features to the dataset that are computed from the data itself. Those features are: Length of comments Number of exclamation marks - Data showed severe toxic comments with multiple exclamation marks. Number of question marks Number of punctuation symbols - Assumption is that angry people might not use punctuation symbols. Number of symbols - there are some comments with words like f\*\*k, $#\*t etc. Number of words Number of unique words - Data showed that angry comments are some- times repeated many times. Proportion of unique words

Conclusion: All the above features had correlation of <0.06 with all labels. So, we decided that adding these features does not benefit the model.

**9. MODEL BUILDING** - Our basic pipeline consisted of count vectorizer or a tf-idf vectorizer and a classifier. We used OneVsRest Classifier model. We trained the model with Logistic Regression (LR), Random Forest (RF) and Gradient Boosting (GB) classifiers. Among them LR gave good probabilities with default parameters. So, we then improved the LR model by changing its parameters.

**10. TRAINING, VALIDATION AND TEST METRICS**

10**.1 TRAINING AND VALIDATION SPLIT** -To know whether was generalizable or not, we divided the into train and validation sets in 80:20 ratio. We then trained various models on the training data, then we ran the models on validation data and we checked whether the model is generalizable or not. Also, we trained different models on training data and tested those on validation data, then we arrived at our best model.

**10.2 TEST METRIC**- We used Receiver Operating Characteristic (ROC) along with Area under the curve (AUC) as test metric.

**10.3 RESULTS FOR VARIOUS MODELS**

**10.3.1 BASE MODEL:** We created a model without any preprocessing or parameter tuning, we used this model as our model, and measured our progress using this model.

**10.3.2 Random Forest**: Next, we created our model using Random Forest. We used n\_estimators = 10 and random\_state = 1 as parameters. We observed the following results: 1. Cross Validation Results 2. ROC – AUC Curve From the Cross Validation results table and ROC-AUC Curve, it is clear that Random Forest performs poorly compared to our base model itself, so we proceeded to tune parameters for Logistic Regression for better accuracy.

**10.3.3 LOGISTIC REGRESSION**

I. We created one model using C = 4 as parameter. The following results were observed. 1. Cross Validation Results 2. ROC –AUC Curve

II. We created another Logistic Regression by selecting the best parameters by cross - validating the following parameters. 3. Cross Validation Results 4. ROC – AUC Curve Though, (i) gave better score compared to (ii) on validation set, with difference in order of 0.0001. When run on the actual data (ii) was found to better than (i)

**11. CONCLUSION** -After checking the kaggle discussion board of the actual competition, standard Machine Learning approaches yield a maximum score of 0.9792, irrespective of any approach. In order to get a large margin over this score one has to employ Deep Learning (DL) techniques.