# ML COURSE PROJECT REPORT

# YAHOO TROLL QUESTION DETECTION

**TEAM NOVA**

**Sai Teja IMT2020538**
**Rithvik Ramasani IMT2020543**

# 1. PROJECT INTRODUCTION

In this era of Quora and Reddit, to improve the legitimacy in the posts and to decrease the toxicity in its community, Yahoo has asked us to create a machine learning model that can detect spam and troll questions and remove them when posted. So, by the use of various NLP techniques, we designed a model that can detect a troll post.

# 2. DATA ANALYSIS

Firstly, we went through the whole dataset and made the following conclusions.

- We searched for the null values in the data set. We found there are no null values in the dataset.
- We searched for duplicate values and found no duplicate rows in the dataset.
- We compared the output values to check whether the dataset is biassed or not. We found that data is biassed, the number of 1's in the dataset are 61870 and 0's are 9,38,130. So, trolls in the dataset are just 6%, the data here is so imbalanced that the number of trolls are very less in comparison to non-troll posts.

# 3. DATA CLEANING

- *Lower Casing Data* - Here we are converting the whole data in the question_text column to lowercase, by doing so we can project the whole data into the same feature space. However, this can cause a few problems, like in cases of acronyms.
- *Expanding Contractions* - Contractions are expanded to their Counterparts in the dataset and this is done by the use of the contractions package.
- *Unwanted Text Removal* - Various forms of unwanted texts are removed in this section. Some of the forms are URL's, HTML tags, Non-ASCII characters, emoticons, symbols & pictographs, punctuations, etc.
- *Other Cleaning* - Correcting the typos, maintaining the slang, expanding some common acronyms and abbreviations.

# 4. DATA PRE-PROCESSING

After completing the data-analysis and data cleaning, we proceeded towards the data pre-processing step, which in this case is natural language processing.

1. *Dealing with Imbalance* - As we have seen in data-analysis step that this data set is imbalanced, so we tried to deal with the imbalance by the use of RandomOverSampler, RandomUnderSampler, SMOTE.
2. *Tokenization* - Tokenizing the text
3. *Stop words removal* - Removing most commonly used words.
4. *Word analysis* - This analysis is performed by doing stemming and lemmatization on the posts. For stemming, we tried PorterStemmer and LancasterStemmer. For lemmatization, we tried WordNetLemmatizer.
5. *POS Tagging* - The part of speech (noun, verb, adjective, etc.) of each word in the text is identified using part of speech (POS) tagging. Since we may infer a word's contextual meaning by determining its POS, this is a crucial stage for many NLP applications
6. *Feature Extraction* - In this part we tried two different types of vectorizers
    a. *TF-IDF Vectorizer* - We performed both character and word vectorization. The hyper parameters choses are
        i. lowercase = True
        ii. strip_accents = “unicode”
        iii. ngrams = (1,3)
        iv. min df = 2, max df = 0.5
    b. *Count Vectorizer* - We performed both character and word vectorization. The hyper parameters choses are
        i. lowercase = True
        ii. strip_accents = “unicode”
        iii. ngrams = (1,3)
        iv. min df = 2, max df = 0.5
        v. sublinear_tf = True
        vi. *stop_words =* ‘english’
        vii. *analyzer =* ‘word’, ‘char’

# 5. MODEL FITTING

We tried fitting the data across various models. The model with the highest F1 score is chosen and then we perform hyperparameter tuning on that model.

1. *Naive Bayes* - Basically by choosing this model we tried to fit the data to a gaussian model. We implemented two types of naive bayes classifiers
    a. *Multinomial Naive Bayes* - It is used for discrete counts.We implemented the multinomial naive bayes model and in this model we consider a feature vector where a given term represents the number of times it appears or very often i.e. frequency.The best F1 score recorded for this model is 0.498.
    b. *Bernoulli Naive Bayes* - We implemented the bernoulli naive bayes model considering the data set to be bernoulli i.e. the feature is present or not. The best F1 score recorded for this model is 0.445.
2. *Logistic Regression* - We implemented the logistic regression model, by choosing this model we assume that there is a linear relation between the input data i.e. the features and the output. The F1 score recorded for this model is 0.599. We used predict_proba to predict the probability of the output classes and altered the threshold to becoming 1 to get desired results.
3. *Random Forest Regressor* - We tried to implement the tree model for the classification model. With initial max_depth at 100 we implemented the model and found out the F1 score was not the best of all and was 0.51 and later we tried with increased max_depth, but the results did not improve. One possible reason for this outcome could be that random forest couldn't extrapolate the linear trend and moreover we can understand that random forest is not a good choice for linear and sparse data sets.
4. *Ridge Classifier* - We tried to implement the ridge classifier, by doing so we aimed at reducing the standard error by adding some bias in the estimates of the regression. The best result achieved with this model was 0.55, hyperparameters are set to, alpha = 0.5,normalize = True,solver ='auto','sag', 'saga', 'lbfgs',tol = 0.001. We tested with four different types of solvers and the best F1 score is given by 'auto'.

5. *Support Vector Machine* - As a last attempt to find the best possible training we tried to implement SVM, we tested the model with four different kernels, Linear, RBF, Poly, Sigmoid. By the use of this kernel we are able to transform the non-linear relation among the input features to a linear relation in a higher dimensional space. And, the best F1 score was given by the ‘linear’ kernel, and it is 0.496. The possible reasons for this model to not work is that, SVM does not perform well on imbalance dataset and moreover noise sustainability of SVM is very poor.

From the various models mentioned above, we can notice that the best possible model would be ‘Logistic Regression’ as it is giving the best results. So, now we can perform the hyperparameter tuning on the ‘Logistic Regression’ model.

## 6. HyperParameter Tuning

After trying various models, we ended with the ‘Logistic Regression’ as the best model. Now, we perform hyper parameter tuning on the Logistic Regressor by the use of a hyperparameter method called GridSearchCV. The parameter grid passed to the GridSearchCV is -

```
param_grid = [{
'penalty' : ['l1','l2','elasticnet','none'],
'solver'  : ['lbfgs','sag','liblinear','saga','newton-cg']
}]
```

With this done we found that best model, was found with the following hyperparameters -

```
max_iter=1000,
C=0.215,
class_weight=wghts,
solver='liblinear',
penalty = ‘l2’
```

## 7. Finding the Best Fit

With various trails, we found out the best fitting model for our dataset is with Logistic Regression.

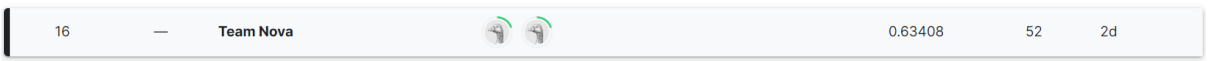
The complete procedure for model is -

1. Lowercasing the Data
2. Tokenizing the sentences
3. Extracting Features (CountVectorizer)
   HyperParameters -
   lowercase = True
   strip_accents = “unicode”
   ngrams = (1,3)
   min df = 2, max df = 0.5
   sublinear_tf = True
   analyzer = ‘word’
4. Logistic Regression Model Training
   HyperParameters -

```
max_iter=1000,
C=0.215,
class_weight=wghts,
solver='liblinear',
penalty = ‘l2’
class_weight = {0.2,0.4}
```

5. Setting the Threshold for predict_proba as 0.32.

This sums up our best fitting model.
The Local Best F1 score is 0.648
Kaggle Best F1 score (50% data) is 0.644
Kaggle Best F1 score (100% data) is 0.634



## 8. Reasons for Failure of Few Techniques

a. *Stop Words Removal* - While performing sentiment analysis it is always preferred to not remove stop words, as it can alter the complete meaning of the sentence. For example if we consider a sentence ‘Smoking is not good’, when removed stop words become ‘Smoking good’ which is completely opposite of what we tried to imply.
b. *Stemming* - Stemming, in general, combines various tokens into one. For example,if we consider dogs, dogged, both become dog after stemming. There is an overall improvement in recall (i.e., discovering more hits from the general pool) and a decrease in precision in information retrieval and classification (i.e. more hits are false positives, as the stemmed signal is diluted).
c. *Lemmatizing* - While dealing with the text data, tense plays a key role in deciding the meaning of the sentence. But when lemmatized a sentence meaning and structure could be completely altered. And moreover lemmatization alters the structure and the style of the sentence/document, which could result in changing the point of the post.

These Pre-processing techniques distort the actual Troll structure. So, they do not give desired results. And hence they are omitted.