“TOXIC COMMENTS CLASSIFICATION USING MACHINE LEARNING”

The Final Report

Submitted for the partial completion of Bachelor Degree in Computer Engineering Awarded by Pokhara University, Nepal.

SUBMITTED BY:

Amrit Gurung (17070289)   Madan Pandey (17070297)

Madhu Sudan Bhattarai (17070298)   Santosh Tharu (17070311)

Shankar Poudel (17070314)   Tej Narayan Chaudhary (17070318)

Lumbini Engineering, Management & Science College

Bhalwari, Rupandehi Jul,

2021

Recommendation

This is to certify that this project entitled "**TOXIC COMMENTS CLASSIFICATION USING MACHINE LEARNING** prepared and submitted by Amrit Gurung, Madan Pandey, Madhu Sudan Bhattarai, Santosh Tharu, Shankar Poudel and Tej Narayan Chaudhary in fulfillment of the requirements of the Bachelor Degree in Computer Engineering awarded by Pokhara University, has been completed under my supervision. Thus, I would like to recommend this project for final presentation as well as acceptance by the University.

…………………	………………..	………………..
HOD	Supervisor	Data Signed
Mr. Satish Kumar Karna	Mr. X	

Certificate

This project entitled "Toxic Comments Classification Using Machine Learning," prepared and submitted by Amrit Gurung, Madan Pandey, Madhu Sudan Bhattarai, Santosh Tharu, Shankar Poudel and Tej Narayan Chaudhary has been examined by us and is accepted for the award of Bachelor Degree in Computer Engineering by Pokhara University, Nepal.

……………………..	……………………	………………………..
External Examiner	HOD	Principal
	Mr. Satish Kumar Karna	Er. Raj Kapoor Shah

. . . .

Supervisor

Date Signed…………………

Mr. X

Declaration

We hereby declare that this project entitled "**TOXIC COMMENTS CLASSIFICATION USING MACHINE LEARNING**" based on our original research work, where it is indebted to the work of others, acknowledgements has duly been made.

1.Amrit Gurung (17070289)	…………………………….
2. Madan Pandey (17070297)	…………………………….
3. Madhu Sudan Bhattarai (17070298)	…………………………….
4. Santosh Tharu (17070311)	…………………………….
5.Shankar Poudel (17070314)	…………………………….
6. Tej Narayan Chaudhary (17070318)	…………………………….

Date signed: ………………………

Acknowledgement

We would like to heartfelt thanks to respectful personality Mr. Satish Kumar Karna, Head of Department of computer engineering, Lumbini Engineering College who gave us his valuable contribution, support, and advice, stimulating suggestion and encourages us to improve our enthusiasm for this project.

We extend our gratitude to Mr. Tulsi Sapkota, Member of Secretary and Er. Raj Kapoor Shah, Principal of Lumbini Engineering, Management and Science College, for their valuable support and inspiration to do this project.

We would cordially like to thank Mr. Narayan Dhakal, Vice- Principal of Lumbini Engineering College for his help, co-ordination, support, and motivation.

1.Amrit Gurung (17070289)

2. Madan Pandey (17070297)

3. Madhu Sudan Bhattarai (17070298)

4. Santosh Tharu (17070311)]

5.Shankar Poudel (17070314)

6. Tej Narayan Chaudhary (17070318)

Abstract

In this era of digital age, discussing things you care about online 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 effectively facilitate conversations, leading many communities to limit or completely shut down user comments. So, classification of the user comments or opinions for toxicity is important to make effective and abuse free online social platforms. In this project, we are trying to classify user comments into distinct categories so as to determine very harmful to benign comments on social platforms or any other platforms.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLE

Chapter 1

Introduction

1.1 Background

Over the decade, there has been a tremendous rise in social networking and social media platforms. People are now able to express themselves and their opinions and also discuss among other via these platforms. However, a conflicts may arise due to the difference in the opinions. Most of the times, this conflicts goes beyond simple differences and results in fight over the social media and often leads to the use of obscene and abusive language. Such conversational toxicity is an issue that can lead people both to stop genuinely expressing themselves and stop seeking others' opinions out of fear of abuse/harassments. The goal of this project will be to use machine learning and deep learning to identify toxicity in text, which could use to help deter users from posting potentially harmful messages, comments, craft more civil arguments when engaging in discourse with others, and gauge the toxicity of other users' comments.

This project aims to implement various machine learning and deep learning algorithms, to tackle the above task. As this is multi-label classification task, so any single comment can have more than one toxic category.

1.2 Problem statement

The input to our algorithms are the comments from online platforms. We than use natural processing techniques with machine learning and deep learning algorithms to detect and classify these comments for different types of toxicity for example identity hate, obscene e.t.c.

1.3 Objectives

With the increasing use of social media platforms, there has been sharp increase in toxic comments in these online platforms. Platforms struggles to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. It is important that toxicity in online platforms is reduced so that the internet is safe place for people no matter their gender, age, race, nationality, likes, dislikes, religious beliefs or political beliefs.

1.4 **Application/Scopes**

Our project have following applications:

1. Filtering posts and comments
2. Social Media Policing
3. User Education

1.5 Feasibility Analysis

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. A system should have reasonable cost and should be technically and operationally feasible to be actually implemented. Our project is studied and analyzed under following feasibility analysis methods:

1.5.1 Technical Feasibility

Technical Feasibility is defined as the feasibility that is concerned with specifying equipment and software that will successfully satisfy the user requirement. It compasses the technical needs of the system.

Toxicity comments detecting and classifying in technical aspect is practical as it can be actually realized to the users. The technology used in the system like machine learning and data mining is robust and efficient technology. Programming language used for scripting and logical task is an efficient and robust language. Thus, the technology used in this system is mature and we currently possessed the necessary technology.

1.5.2 Economic Feasibility

Economic Feasibility is a measure of the cost effectiveness of the project or the solution. This is often called a cost benefit analysis.

The system that is developed should be inside a reasonable cost that can be actually adapted by the users. The software tools used in our project and server are totally free of cost. The cost of production of system capable of detecting toxic comments & classification and eventually filtering the posts and comments can vary depending upon the size of the system and functionalities it supports as well as it's user base. So the production cost and maintenance cost can vary. Currently our system is capable of detecting and classifying toxic comments but it is nowhere near to robustness exhibited by top social networking sites. Thus, Our project is economically feasible at least for now.

1.5.3 Operational Feasibility

Operational feasibility is a measure of how well a specific solution will work in the organization. It is also a measure of how people feel about the system or the project. It also analyze the inside operation on how a deemed process will work, be implemented, and dealing with change resistance and acceptance.

After analyzing the technical and economic feasibility study, next would come the operation analysis. Our system is such that it allows user to check whether the comments or post are toxic or not. A robust system can do a lot then just detecting toxicity but our web system, for now is not robust but doable. Thus, this project does not require the specific training in order to implement. A user can just enter the comment/post and our system will detect toxicity categories/labels. Therefore, the project is operationally feasible.

1.6 Features

This project have following features

1. A comprehensive analysis of online comments for toxicity labels using NLP techniques with aids of machine learning and deep learning algorithms.
2. A web application for classifying entered comments.

1.7 System Requirements

This project requires the following system requirements.
Note: Following system is inefficient for deep learning algorithms and Hyperparameter tuning. We have done everything except web application building in Google Colab.

1.7.1 Hardware Requirements

A well functioning pc with minimum of 2 GB RAM , a processor with minimum 2 GHz and enough Hard drive space.

1.7.2 Software Requirements

For the development, following tools are used:

- Python (various libraries)
- Python Runtime Environment
- Google Colab
- Jupyter Lab
- Spyder/Visual Studio Code/Sublime Text (for structuring and web application development)

Chapter 2

Related Works

Before deep learning (NLP), companies resorted to ineffective methods of identifying hate speech, such as simple keyword searches (bag of word). This method has "high recall but leads to high rates of false positives" [9], mistakenly removing normal conversation.

Sentiment Classification regarding toxicity has been intensively researched in the past few years, largely in the context of social media data where researches have applied various machine learning algorithms to try and tackle the problem of toxicity as well as related, more well-known task of sentiment analysis. Comment abuse classification research initially began with Yin et al's application of combining TF-IDF with sentiment/contextual features. A paper published in August 2019 used multiple-view stacked Support Vector Machine (mSVM) to achieve approximately 80% accuracy with data from various social media companies [11]. . In another study, Badjatiya et al., used extensive experiments with multiple deep learning architectures to learn semantic word embedding to handle toxic comments identification [12].

In addition, many social media companies have invested in methods to eliminate online hate speech. In July 2020, Facebook Canada announced that it is "teaming up with Ontario Tech University's Centre on Hate, Bias and Extremism to create what it calls the Global Network Against Hate" [10], for which Facebook will invest $500,000 to spot online extremism and countering methods.

Chapter 3

Machine Learning Algorithms

This Project employs three machine learning algorithms and one deep learning algorithms. These algorithms are Naïve Bayes, Logistic Regression (later removed), Support Vector Machine (SVM) and Long Short Term Memory (LSTM). We will be explaining each algorithms below excepts for Logistic Regression.

3.1 Naïve Bayes Model

Naïve Bayes model is based on Bayes Theorem and is particularly suited for when the dimensionality of the input is high. It simplifies computational complexity. Naïve Bayesian Classifier assumes that the effect of an attribute value on the given class is independent of the value of other attributes i.e. class conditional independent.

Let D be training set of tuples and each tuple is represented by n-dimensional vector, $X = (x_1, x_2, \ldots, x_n)$ depicting *n* measurements made on the tuple from *n* attributes, respectively, $A_1, A_2, \ldots, A_n$. Suppose that there are *m* classes, $C_1, C_2, C_3, \ldots, C_m$. Given a tuple *X*, the classifier will predict that *X* belongs to the class having higher posterior probability conditioned on *X* i.e. Naïve Bayes classifier predicts tuple *X* belongs to class C_i if and only if

$$P(Ci|X) > P(Cj|X) \qquad for\ i \le j \le m\ ,\ i \ne j$$

$$i.e.\ P(Ci|X) = \frac{P(X|Ci)P(Ci)}{P(X)} \quad \text{must be maximum.}$$

$P(X) = constant$ for all classes , only *P(*X|C_i*)P(C_i) need to be maximized.*

And if the class prior probabilities is not known, then it is commonly assumed that that the classes are equally likely,

$$\text{i.e. } P(Ci) = P(C_1) = P(C_2) = \ldots\ldots\ldots.P(C_m)$$

and we would therefore maximize $P(X|C_i)$. Otherwise we maximize $P(X|C_i)P(C_i)$.

Naïve Bayesian assumes **Class conditional independence** because it would be extremely computationally extensive to calculate $P(X|C_i)$ given that there are many attributes. Class conditional independence presumes that the attribute's value are conditionally independent of one another, given the class label of the tuple. Thus,

$$P(\mathrm{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

$$= P(x_1|C_i) * P(x_2|C_i) * \ldots\ldots.* P(x_n|C_i))$$

We can easily estimate the probabilities $P(x_1|C_i) * P(x_2|C_i) * \ldots\ldots.* P(x_n|C_i))$ from the training tuple. Recall that here x_k refers to the value of attribute A_k for tuple *X.*

To predict the class label for tuple *X*, $P(X|C_i)P(C_i)$ IS evaluated for each class C_l . The classifier predicts that the class label of tuple X is class C_l if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \text{ for } 1 \leq J \leq m, j \neq m$$

In theory, Bayesian classifier have minimum error rate compared to all other classifiers.

However, in practice this is not always the case, owing inaccuracies in the assumptions made for it’s use, such as class-conditional independence, and lack of available probability data.

3.2 Support Vector Machine

Support Vector Machine (SVMs) is method for the classification of both linear and non-linear data. SVM can be used for both classification as well as regression tasks. They belongs to the family of generalized linear models. It uses nonlinear mapping to transform the original training data into higher dimension and within this new dimension, it searches for linear optimal separating hyperplane (i.e., a ‘decision boundary’ separating the tuples of one classes from another). With appropriate nonlinear mapping to sufficiently high dimension, data from two classes can always be separated by hyperplanes. Two parallel hyperplanes are constructed on each side of the hyperplane that separate the data. The separating hyperplane is the hyperplane that maximize the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be.

Let the dataset D be given as (X_1, y_1), (X_2, y_2),…………, $(X_{|D|}, y_{|D|})$, where X_i is the set of training tuples with associated class labels, y_i . Each of the y_i can take one of the two values, either +1 or -1 corresponding to the two classes.

A separating hyperplane can be written as

$$W^T.X + b = 0 \qquad \text{…………(1)}$$

Where W is the weight vector, namely $W = \{w_1, w_2, w_3, w_4, \ldots, w_n\}$; n is the number of attributes and b is a scaler, often refer to as bias. This bias allows us to increase the margin. In the absent of bias term, the hyperplane is forced to pass through the origin, restricting the solution.

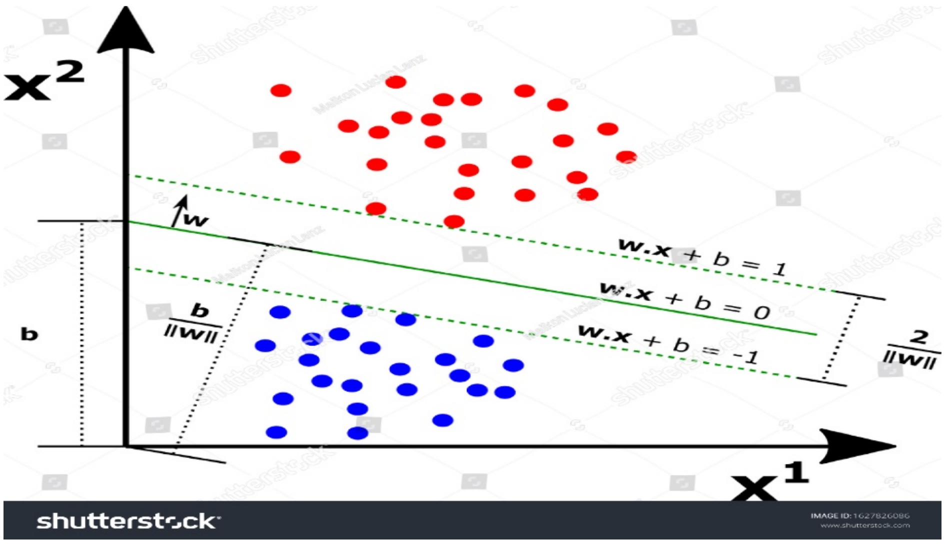


Figure 1 Maximum Marginal Hyperplanes for SVM

As we can see in the above figure, there are two dotted lines parallel to the hyperplane and passing through some points. These dotted lines are called marginal plane or marginal hyperplane and points through which it passes are called support vectors. We are interested in maximizing the marginal distance, the distance between two marginal plane, because the hyperplane with maximum margin seems to classify the unseen data more accurately hence reducing the classification error as compared to that with lower margin.

The given two parallel marginal planes or marginal hyperplanes can be described by the equations

$H_1{:}W^T.X + b = 1$ …………(2) (upper margin plane)

$H_2{:}W^T.X + b = -1$ …………(3) (lower margin plane)

That is, any tuples that falls on or above H_1 belongs to class +1, and any tuples that fall on or below H_2 belongs to class -1. They above equations can be combined together and let's write them in inequalities form, we get

$y_i(W^T.X + b) \geq 1, \forall_I$ …………(4)

Any training tuples that fall on marginal hyperplanes H_1 and H_2 satisfy the Eq. (5) and are called the ***support vectors***. If the training data are linearly separable, we can select these hyperplanes so that there is no points between them and then try to maximize their distance. By geometry, we find that the distance between these two marginal hyperplanes is $2/||W||$. So the distance between separating hyperplane and each marginal hyperplane is $1/||W||$. In order to find the optimal separating hyperplane having a maximal margin, a learning machine should minimize $||W||^2$ subject to the above inequality (5). (Note: This optimization problem is solved by the saddle points of Lagrange's Function.)

3.3 Long Short Term Memory

.One of the problems of LSTM is the problem of Long-Term Dependencies. In theory. RNN are absolutely capable of handling such ''long-term dependencies''. Sadly, in practice, they don't seems to be able to learn them. Long Short Term Memory is a special kind of RNN, capable of learning long-term dependencies [3]. LSTM architecture have gates that regulate the flow of information inside cell. Usually these gates are input, output and forget gate- but there are variations to this. The input gate control the flow of input activations, while the output gate controls the flow of output activations. The forget gate scale the cell internal state, and therefore forgetting and resetting the cell's memory.

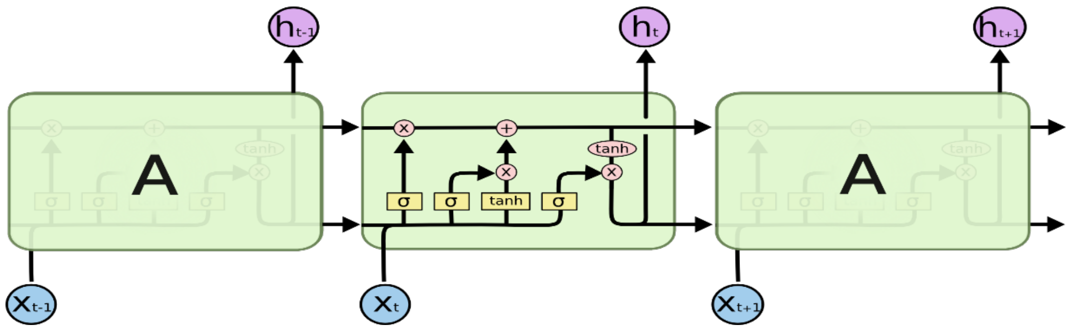


Figure. The LSTM model

Chapter 4

Methodology

4.1 Process Workflow

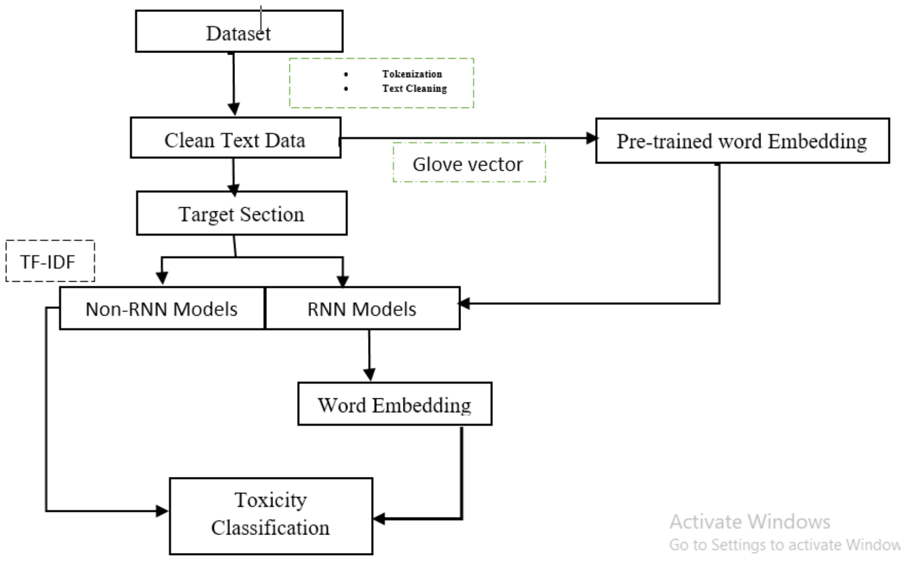


Figure. Project Process Workflow

Let us just explain above workflow.

- We first collected the dataset for our project.

- After we have our required data, the first thing we did was data preprocessing. It involves cleaning and preprocessing text data (comments, in our case).
- Than we separated dependent and independent data. Here dependent data is toxic labels and independent data are column containing online comments.
- After that we have two choices, either to use RNN Model or non-RNN models. For non-RNN model, we performed word embedding on our data using TF-IDF technique. Similarly, for RNN model we used pre-trained Glove vector for word embedding
- Eventually, we trained our model and performed classification of online comments for toxic categories. A comment can be entirely non-toxic as well. For such non-toxic comment, our model does not predict any toxic label indicating comment is not toxic.
- After building model, we evaluated our models and tested for unseen data/comments.

4.2 Datasets

We trained and evaluated our models on Toxic Comment data available in Kaggle. The data contains 6 group of toxic categories. This is a multi-label classification problem, so each comment can belong to zero or more toxic categories. The 6 toxic categories given are

- toxic
- severe toxic
- obscene
- threat
- insult
- identity hate

Our dataset contains almost 160k rows of online comments, all labeled. The distribution of comments of six toxic categories are shown in the table below.

As mentioned, this is multi-label classification problem so same comment can belong to zero or multiple categories.

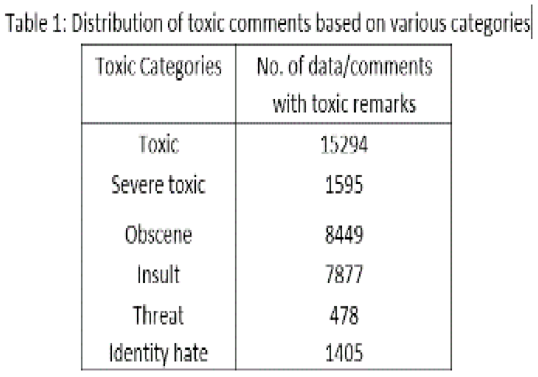
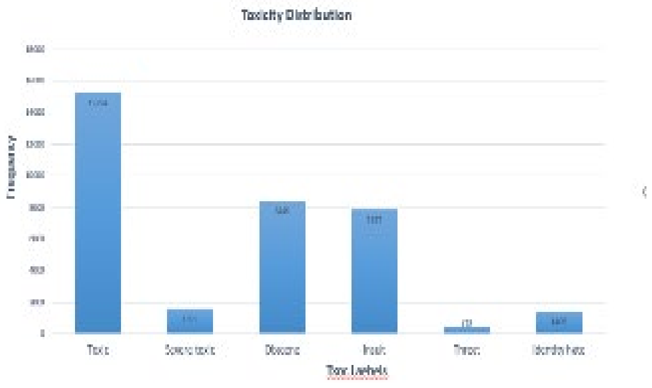


Table 1: Distribution of toxic comments based on various categories

Toxic Categories	No. of data/comments with toxic remarks
Toxic	15294
Severe toxic	1595
Obscene	8449
Insult	7877
Threat	478
Identity hate	1405



Figure, Table. Toxicity Distribution

4.3 Imbalanced Data and Handling Imbalanced Data

Out of ~ 160K comments over 130K comments are non-toxic leaving behind only about 35K toxic comments. This implies that our dataset is highly imbalance. Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 examples in the minority class to the majority class. Let us look at the class distribution of each toxic labels,

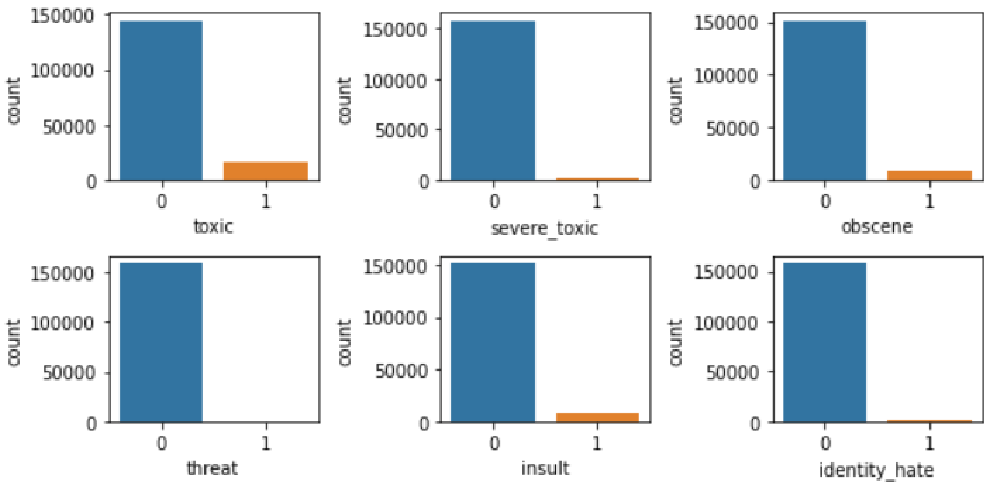


Figure. Class distribution for toxic categories labels (imbalanced data)

We can see that for each toxic label, the frequency of non-toxic comments (label 0) is extremely high compared to frequency distribution of toxicity labels (label 1). So there is severe skewness in the class distribution.

This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. This is a problem as it is typically the minority class on which predictions are most important (like our where minority class (toxic labels) are important).

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called *undersampling*, and to duplicate examples from the minority class, called oversampling.

At first we tried oversampling method in order to avoid loss of information but it was very memory intensive to train the models with our system. So we resorted to *undersampling* method.

Random *undersampling* involves randomly selecting examples from the majority class to delete from the training dataset. This has the effect of reducing the number of examples in the majority class in the transformed version of the training dataset. This process can be repeated until the desired class distribution is achieved, such as an equal number of examples for each class. A limitation of *undersampling* is that examples from the majority class are deleted that may be useful, important, or perhaps critical to fitting a robust decision boundary. Given that examples are deleted randomly, there is no way to detect or preserve "*good*" or more information-rich examples from the majority class.

Now let's look at the class distribution of various toxic labels. We can see that the class labels for each toxic category is balanced. Our dataset size now is comparatively much lower than original. We created the dataframe for each toxic category and trained the model using *OneVsRest* strategy.

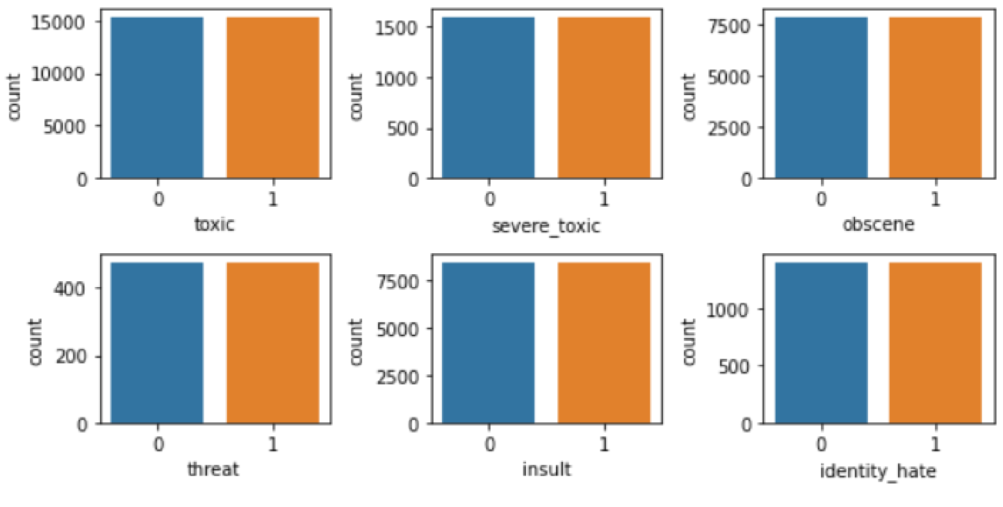


Figure. Class distribution for toxic categories labels after undersampling

Now here is a tricky thing, most of the time when we train the dataset with class imbalance, we will get sufficiently high accuracy and low AUC scores. But in such case our high model accuracy means nothing because our model is biased toward majority class label data ignoring the minority class label data. We shall evaluate results for both cases: with imbalanced data (our original data) and with handled class imbalanced data (undersampled data in our case).

4.4 Should we handle Class Imbalance for our dataset?

Handling class imbalance is pretty much important. Why? We explained the reasons above already. But question is should we really handle class imbalance for our project? Now it depend on the approach we employ. In our project, we evaluated result for both imbalanced data and undersampled data. One thing we should consider seriously is our project is based on multi-label classification. In multi-label classification, an observation can belong to multiple categories. Our toxicity categories are distributed very differently (refer to Figure. Toxicity Distribution). After handling class imbalance for each toxicity categories, we will get six dataset of different sizes for each categories because toxic categories are distributed very differently. Then we have to train model for each datasets. Training model on such datasets might give decent accuracy for particular toxic category, but our problem is based on multi-label classification, and in such cases our model fails to predict multiple categories for each observation miserably.

But we have different way to cope with this as well. We will not be handling class imbalance and will be using ***OneVsRest*** strategy to train model. In this strategy, we train model for each toxic categories but our dataset will remain same for each training.

We shall evaluate results for both the ways mentioned above.

4.4 Data Preprocessing

Data pre-processing is transforming the data into a basic form that make it easy to work. This project is based on Natural Language Processing. Natural Language Processing (NLP) is a branch of Data Science which deals with Text data. Apart from numerical data, Text data is available to a great extent which is used to analyze and solve business problems. But before using the data for analysis or prediction, processing the data is important.

To prepare the text data for the model building, we perform text preprocessing. It is the very first step of NLP projects. Some of the preprocessing steps are:

- Removing punctuations like . , ! $( ) * % @
- Removing URLs
- Removing Stop words: words commonly found in text eg. ‘is’, ‘am’, ‘the’, ‘a’ etc. Such words need to be removed so that we can focus on the words/text that matter more for our analysis or prediction.
- Lower casing: converting all texts to lowercase so that we don’t get different word vectors for same word.
- Tokenization: It allows efficient filtering of the words before vectorising text. The tokenization of “I am Engineering student.” is [“I”. “am”, “Engineering”, ”student.”]. After this we can easily filter out stop words , short words(sometimes) and vectorize words/text.
- Stemming
- Lemmatization

At first we tried stemming and Lemmatization too but the result was not good as expected so we removed these two from our project.

4.5 Text Representation and Feature Engineering

After the text preprocessing, we need to convert that preprocessed text into proper representation. By representation we mean, we need to convert text into numeric format. Our computer does not understand such high level linguistics so we need to represent text in numeric format. We can achieve this by process called Word Embedding. Word embedding is a popular techniques to convert text into word vectors (numeric vector). The basic requirement of word embedding or vectorized text is:

- It should not result into sparse, since it cost more computational costs.
- It should return most of the linguistic information.

For our models, we have used two word embedding techniques; TF-DF for non-RNN models models and Glove vector for RNN model.

4.5.1 TF-IDF

- It denotes Term Frequency and Inverse Document Frequency.
- TF-IDF is a numerical statistics that is intended to reflect how important a word is to a document in a collection of corpus.
- Common words or Stop words like ‘is’, ‘the’, ‘a’ etc. tends to appear quite frequently in every documents. Such stop words does not carry much information. TF-IDF penalizes these common words by assigning them lower weights and giving more weights to the words that actually matters. For our project, for e.g. we give more weights to words like nigga, fuck, goddamn, morons etc. which represents toxic words.

- TF(x) = (Number of times word x appears in a document) / (Total number of words in the document)
- IDF(x) = log(Total number of documents) / (Number of documents with word x in it)
- Finally, TF(x) * IDF(x)

4.5.2 Glove Embedding

- For our deep learning model, Glove was chosen to embed the text in the data.
- Glove is a pre-trained word vectors which were trained on words from Wikipedia 2014– and Gigaword 5 corpus.
- The Glove package contains four different versions of dimensions – 50, 100, 200 and 300. The dimension refers to the length of vectors and larger vectors can store more information. The word vectors were trained on four hundred uncased words and were contained in a text file 989 megabytes.
- The text file was parsed and each line are converted into numpy array and imputed into dictionary.
- Next, the dictionary was mapped to the words that were acquired from the tokenization process.

4.5.3 Padding

The algorithms requires that all the input text have same length. Many text data have varying degree of lengths. A padding length was decided on – and any text shorter than this was padded with zeros. The zeroes are appended at the end of each sentence. Any text longer than the decided length are discarded and removed from the text data. In this experiment padding is especially used in deep learning model (LSTM) and decided padded length is significantly high enough to include all data lengths.

4.6 Model building

There are several supervised ML approaches for NLP multi-label classification tasks. There is no universal model to apply. We had to test different models and tune them but at the end tuning wasn't necessary because model accuracy was already very high enough to not opt for hyperparameter tuning but it is recommended to perform hyperparameter tuning always.

WE used ***OnevsRest*** strategy to train our every classifier. The ***OnevsRes****t* strategy is a strategy where you train N binary classifiers with one class at a time and leaving rest out. In other words, it consists on a multi-label algorithm that accepts a binary mask over multiple labels. The result for each prediction will be an array of 0s and 1s marking which class labels apply to each row input sample. In document classification One-Vs-Rest treats the classes as mutually exclusive and trains 6 different classifiers in our case, corresponding to each toxic category and converting this problem into a binary classification problem. We applied this method for Multinomial Naïve Bayes and Support Vector Machine. Similarly, for deep learning model, I used 300 dimensional

Glove word vectors. I also used max sentence length of 1000

4.7 Software/Web App Development

Till now what we did was all about analysis and prediction which are for now completely based on static notebooks. Every data science projects are done aiming to build software or business products that puts positive values to the business and customers. Without building software system our project is incomplete and it does not put any positive values. With that in mind, we have built rather a simple web based application for classification of user comments. We used

Streamlit library of python for building this web app. Streamlit is an library for rapid prototyping and building web application for data science. It is especially designed for data science. It does not require knowledge of front-end web technology tools to build web application.

Chapter 5

Result and Evaluation

In this chapter we shall perform quantitative evaluation of two cases: one with balanced data (undersampled data in our case) and other with imbalanced data (our original data) and see how these two cases affect our evaluation metrics.

5.1 Evaluation Metrics.

We will be using following accuracy scores for evaluating the model performance.

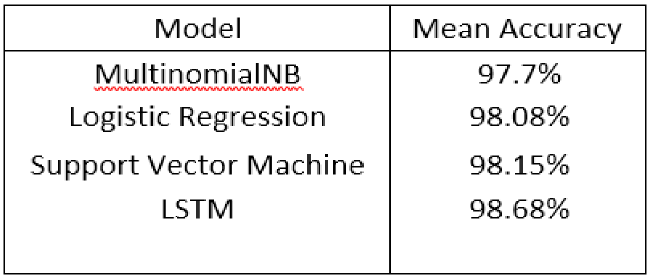
We used the following metrics for evaluating our model performance.

- Accuracy score, classification report, confusion matrix
- F1-score, Recall, precision
- AUC score, ROC_AUC curve

5.2 Quantitative Evaluation (Imbalanced data)

We shall first evaluate imbalanced data. We trained data on three machine learning models and one deep learning model. The mean accuracy obtained after training these models are given below in the table.

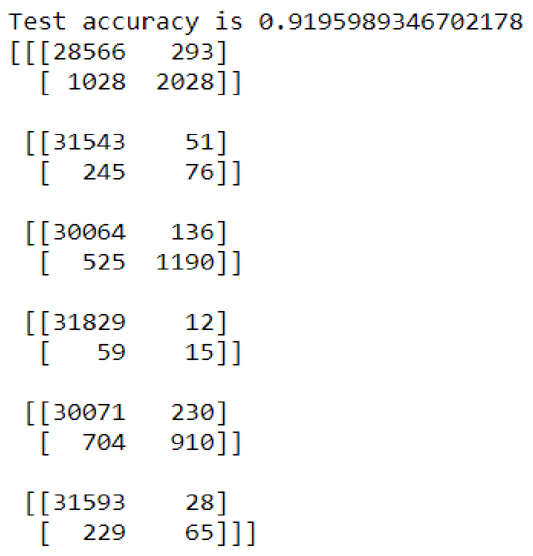
Table: Mean Accuracy Scores (imbalanced data)



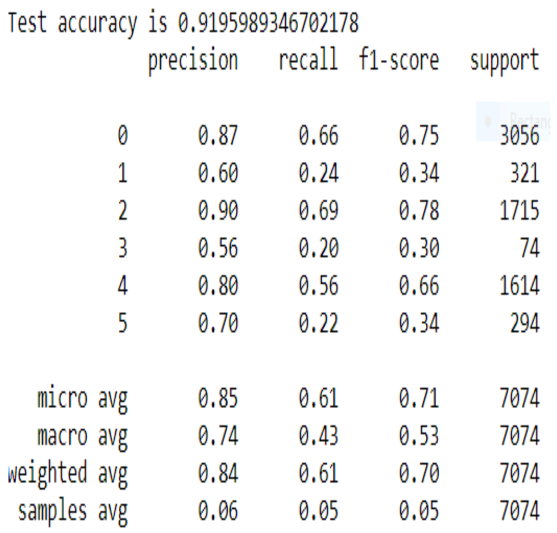
Model	Mean Accuracy
MultinomialNB	97.7%
Logistic Regression	98.08%
Support Vector Machine	98.15%
LSTM	98.68%

It is seen that there is no significant difference between the models we trained our data on though Support Vector Machine and LSTM have slightly highest accuracies. Let us see confusion matrix and classification report as well as AUC scores before concluding the effectiveness of the models.

Since SVM is our best performing model here, we shall only show **Confusion Matrix** and **Classification Report** for SVM along with its test accuracy. Confusion Matrix shows how well our observations are classified. Similarly Classification report shows scores for different evaluation metrics. There are six confusion matrices in ndarray form for six toxic categories.



```
Test accuracy is 0.9195989346702178
[[[28566   293]
  [ 1028  2028]]

 [[31543    51]
  [  245    76]]

 [[30064   136]
  [  525  1190]]

 [[31829    12]
  [   59    15]]

 [[30071   230]
  [  704   910]]

 [[31593    28]
  [  229    65]]]
```



```
Test accuracy is 0.9195989346702178
              precision    recall  f1-score   support

           0       0.87      0.66      0.75      3056
           1       0.60      0.24      0.34       321
           2       0.90      0.69      0.78      1715
           3       0.56      0.20      0.30        74
           4       0.80      0.56      0.66      1614
           5       0.70      0.22      0.34       294

   micro avg       0.85      0.61      0.71      7074
   macro avg       0.74      0.43      0.53      7074
weighted avg       0.84      0.61      0.70      7074
 samples avg       0.06      0.05      0.05      7074
```

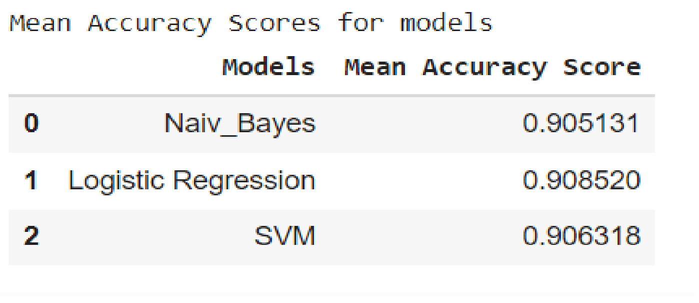
Figure. Confusion Matrix (left) and Classification Report (right) for svm

At a glance we may feel like our models are performing extremely well due high accuracy scores, but if we look into *recall* and *f1-score* scores in the classification report above, it make us think twice. *AUC scores* for all our models were between 0.50 – 0.65 which are considered poor. We are not showing *AUC scores* and *ROC_AUC* curves for imbalanced data. We shall show them for handled imbalanced data (our randomly undersampled data).

In conclusion, our models performance were good enough based on accuracy scores for our imbalanced data. We are using OneVsRest for multilabel classification. SO it it can be considered good enough performance. But there is still room for improvement for AUC scores.

5.3 Quantitative Evaluation (undersampled data)

Now in this section we shall perform quantitative evaluation of random undersampled data. The mean accuracy for our three machine learning models for random undersampled data is given below.



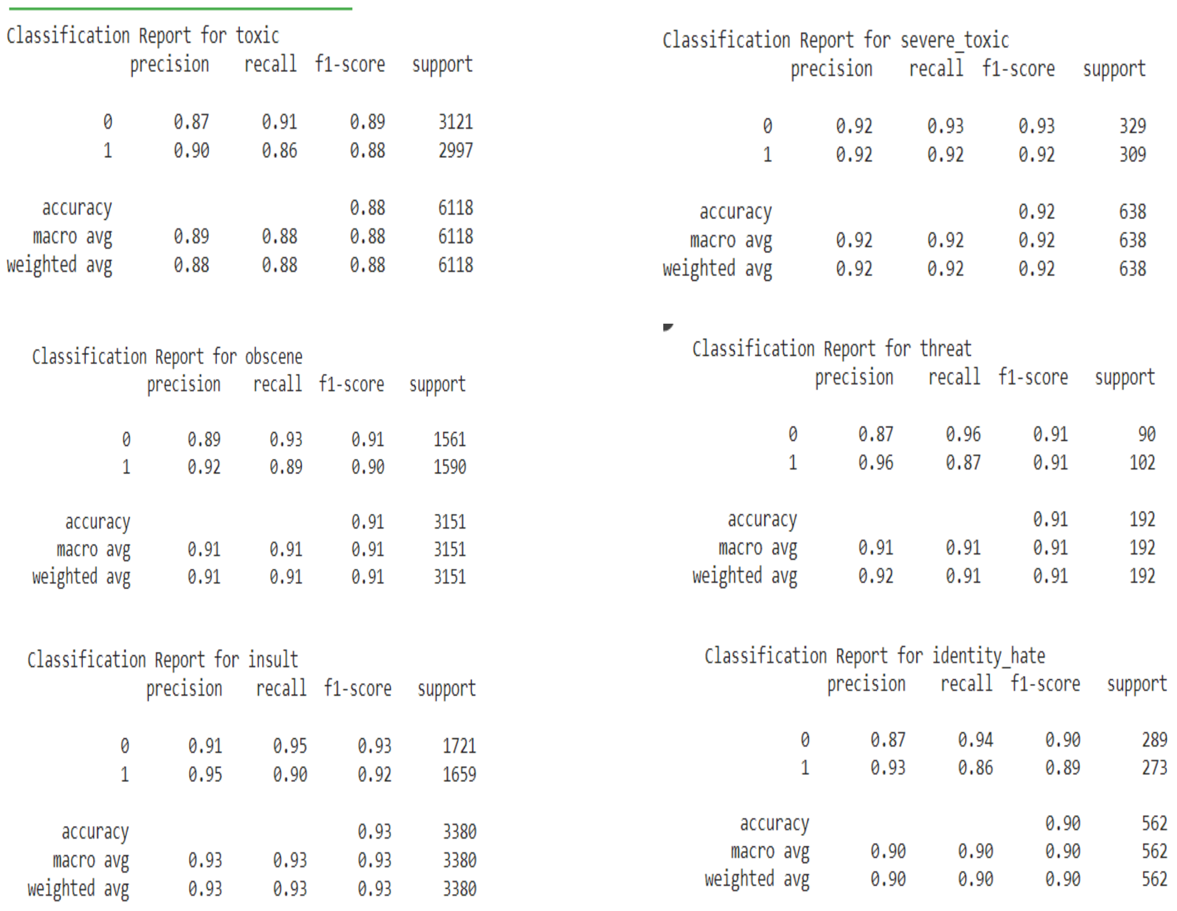
```
Mean Accuracy Scores for models
```

	Models	Mean Accuracy Score
0	Naiv_Bayes	0.905131
1	Logistic Regression	0.908520
2	SVM	0.906318

Figure. Mean Accuracy Scores (undersampled data)

Here we almost similar mean accuracy sores for the test data from undersampled data. All three models have ~90% average accuracy. Let's first look at other evaluation metrics before concluding the performance of our models.

Let's look at the confusion matrix and classification report. We will use that of SVM since we also showed It's for imbalanced data.



```
Classification Report for toxic
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      3121
           1       0.90      0.86      0.88      2997

    accuracy                           0.88      6118
   macro avg       0.89      0.88      0.88      6118
weighted avg       0.88      0.88      0.88      6118
```

```
Classification Report for severe_toxic
              precision    recall  f1-score   support

           0       0.92      0.93      0.93       329
           1       0.92      0.92      0.92       309

    accuracy                           0.92       638
   macro avg       0.92      0.92      0.92       638
weighted avg       0.92      0.92      0.92       638
```

```
Classification Report for obscene
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      1561
           1       0.92      0.89      0.90      1590

    accuracy                           0.91      3151
   macro avg       0.91      0.91      0.91      3151
weighted avg       0.91      0.91      0.91      3151
```

```
Classification Report for threat
              precision    recall  f1-score   support

           0       0.87      0.96      0.91        90
           1       0.96      0.87      0.91       102

    accuracy                           0.91       192
   macro avg       0.91      0.91      0.91       192
weighted avg       0.92      0.91      0.91       192
```

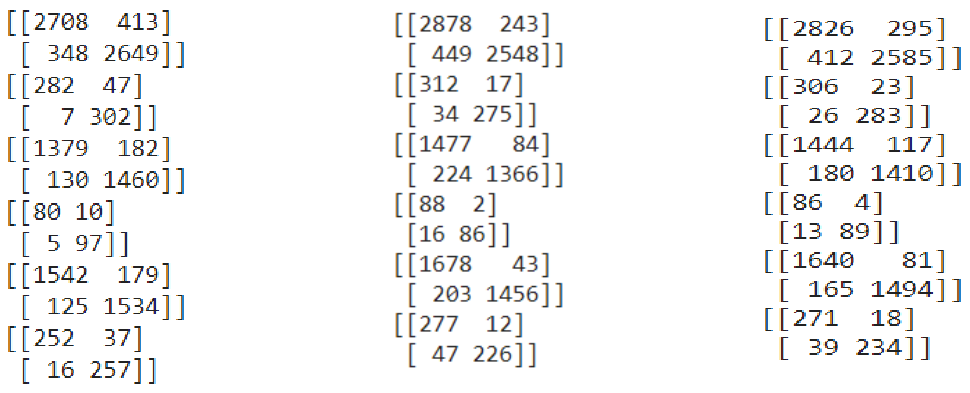
```
Classification Report for insult
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      1721
           1       0.95      0.90      0.92      1659

    accuracy                           0.93      3380
   macro avg       0.93      0.93      0.93      3380
weighted avg       0.93      0.93      0.93      3380
```

```
Classification Report for identity_hate
              precision    recall  f1-score   support

           0       0.87      0.94      0.90       289
           1       0.93      0.86      0.89       273

    accuracy                           0.90       562
   macro avg       0.90      0.90      0.90       562
weighted avg       0.90      0.90      0.90       562
```

Figure. Classification report for each individual toxic categories

In the above figure, classification report for each individual toxic categories is given. We also see that all metrics like accuracy, precision, recall and f1-score have significantly high scores. This is because our model is not biased toward one class label than other since we handled imbalance of data by randomly undersampling. We can find the average scores for all these metrics from above classification reports by simple statistics.

Similarly, let’s now look at the confusion matrixes for each models.



```
[[2708  413]
 [ 348 2649]]
[[282  47]
 [  7 302]]
[[1379  182]
 [ 130 1460]]
[[80 10]
 [ 5 97]]
[[1542  179]
 [ 125 1534]]
[[252  37]
 [ 16 257]]
```

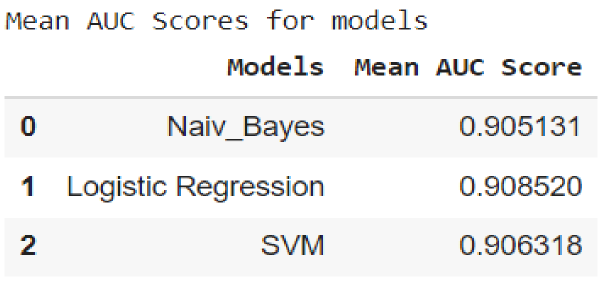
```
[[2878  243]
 [ 449 2548]]
[[312  17]
 [ 34 275]]
[[1477   84]
 [ 224 1366]]
[[88  2]
 [16 86]]
[[1678   43]
 [ 203 1456]]
[[277  12]
 [ 47 226]]
```

```
[[2826  295]
 [ 412 2585]]
[[306  23]
 [ 26 283]]
[[1444  117]
 [ 180 1410]]
[[86  4]
 [13 89]]
[[1640   81]
 [ 165 1494]]
[[271  18]
 [ 39 234]]
```

Figure. Confusion matrix from left to right: Naïve Bayes, SVM, Logistic Regression

The above figure shows confusion matrix for each individual toxic categories for three models from left to right: Naïve Bayes, SVM, Logistic Regression. Each confusion matrix is formed of 6 sub confusion matrix which indicates six different toxic categories. We can see that there are more correctly classified observation than misclassified observation for test data. Diagonal (top left, bottom right) represent True Negative and True Positive, in other words, correctly classified observation. Similarly, Diagonal (top right, bottom left) represent False Positive and False Negative, in other words, incorrectly classified observation.

Similarly, now let's look at the AUC scores and ROC_AUC curve. This will be our final evaluation metrics. The AUC scores and ROC_AUC curve for each individual toxic categories for each models are given below.



Mean AUC Scores for models

	Models	Mean AUC Score
0	Naiv_Bayes	0.905131
1	Logistic Regression	0.908520
2	SVM	0.906318

Figure: Mean AUC Scores (undersampled data)

The above table shows the AUC scores for three models. we have in average `90% accuracy for all three models. This is about 8% lower compared to the best model on the Kaggle LeaderBoard that obtains an AUC scores of 0.98856 (98.856%).

```
AUC scores of toxic: 0.8757772511377125
AUC scores of severe_toxic: 0.9172445677380048
AUC scores of obscene: 0.9008235327297853
AUC scores of threat: 0.9199346405228757
AUC scores of insult: 0.9103228543728343
AUC scores of identity_hate: 0.9066021290655209

mean auc_score :  0.9066821298655209
acu_scores: [0.9066821298655209]
```

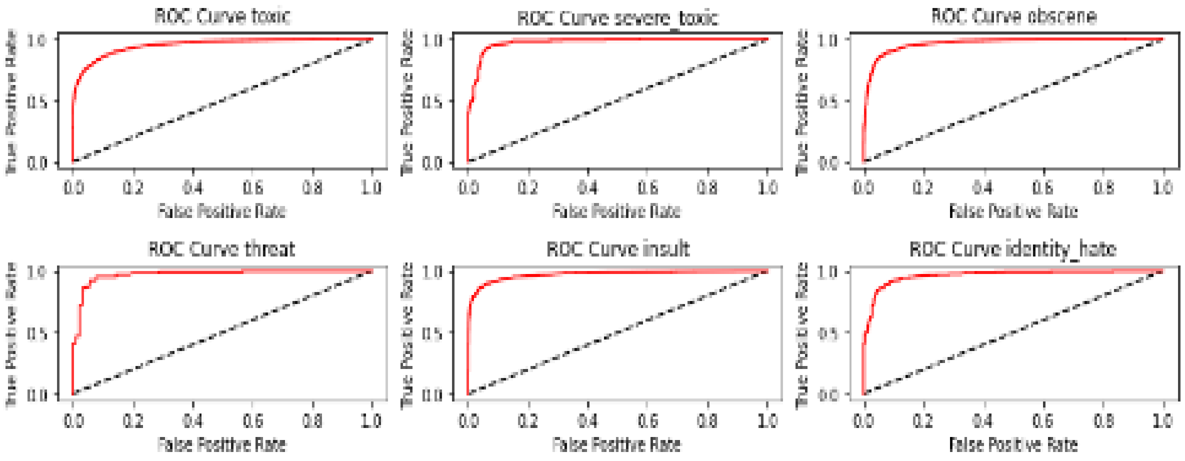


Figure: AUC scores and ROC_AUC curve (Naïve Bayes) for each individual toxic categories

In the above figure, we showed the AUC scores and ROC_AUC curves for each individual toxic labels. This is for Naïve Bayes model. We can see that AUC scores of each individual toxic categories are at the range of 0.89 to 0.92. We shall look for that of Logistic Regression below.

```
AUC scores of toxic: 0.886161928242458
AUC scores of severe_toxic: 0.9191479525088283
AUC scores of obscene: 0.9026530188312604
AUC scores of threat: 0.9104575163398693
AUC scores of insult: 0.9263258286190025
AUC scores of identity_hate: 0.8931581682446734

mean auc_score :  0.8931581682446734
acu_scores: [0.8931581682446734]
```

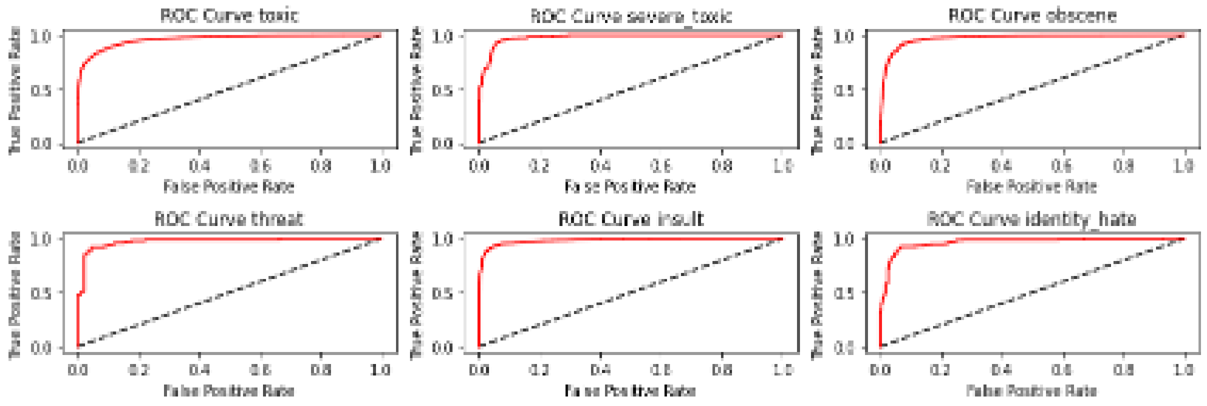


Figure: AUC scores and ROC_AUC curve (Logistic Regression) for each individual toxic categories

In the above figure, we showed the AUC scores and ROC_AUC curves for each individual toxic labels. This is for Logistic Regression model. We can see that AUC scores of each individual toxic categories are at the range of 0.89 to 0.92

So we evaluated quantitatively for all three different models in randomly undersampled data. The scores of different evaluating metrics like accuracy score, confusion matrix, classification report, AUC scores and ROC_AUC curves suggests that our model is performing well enough. But we train model for each categories separately with their own unsersampled data. So we have six dataset of different sizes for each categories. This lead to decent performance on individual category labels but somewhat poor on multi-label prediction. More analysis and use of robust deep learning algorithms can solve this problem.

Similarly, for our LSTM recurrent neural network, the validation loss and validation accuracies are obtained as below

```
Epoch 1/5
852/852 [==============================] - 237s 240ms/step - loss: 0.1106 - accuracy: 0.7926 - val_loss: 0.0582
- val_accuracy: 0.9941
Epoch 2/5
852/852 [==============================] - 203s 239ms/step - loss: 0.0551 - accuracy: 0.9677 - val_loss: 0.0543
- val_accuracy: 0.9938
Epoch 3/5
852/852 [==============================] - 202s 238ms/step - loss: 0.0504 - accuracy: 0.9766 - val_loss: 0.0548
- val_accuracy: 0.9928
Epoch 4/5
852/852 [==============================] - 202s 237ms/step - loss: 0.0490 - accuracy: 0.9684 - val_loss: 0.0546
- val_accuracy: 0.9934
Epoch 5/5
852/852 [==============================] - 201s 236ms/step - loss: 0.0461 - accuracy: 0.9716 - val_loss: 0.0566
- val_accuracy: 0.9935
```

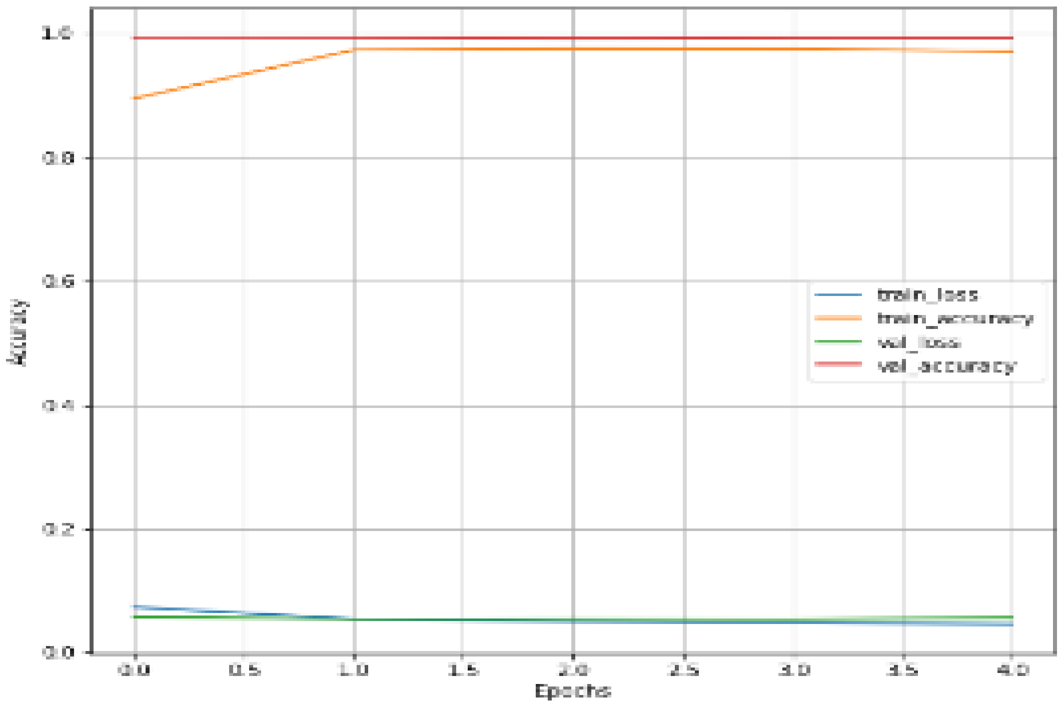


Figure: Train and Validation learning curve (LSTM)

The *val_loss* and *val_acc,* obtained are **0.056** and **0.9935** respectively. The *train_loss* and *train_acc,* obtained are **0.0461** and **0.9716** respectively. This result is already pretty impressive but turning the hyperparameters might still rise accuracy slightly.

5.4 Model comparison

Now we shall compare models based on accuracy score, AUC scores for all three models for each individual toxic categories in both undersampled and imbalanced data.

First we shall compare accuracy scores for each toxic categories for imbalanced data.. This is shown in the figure/table below.

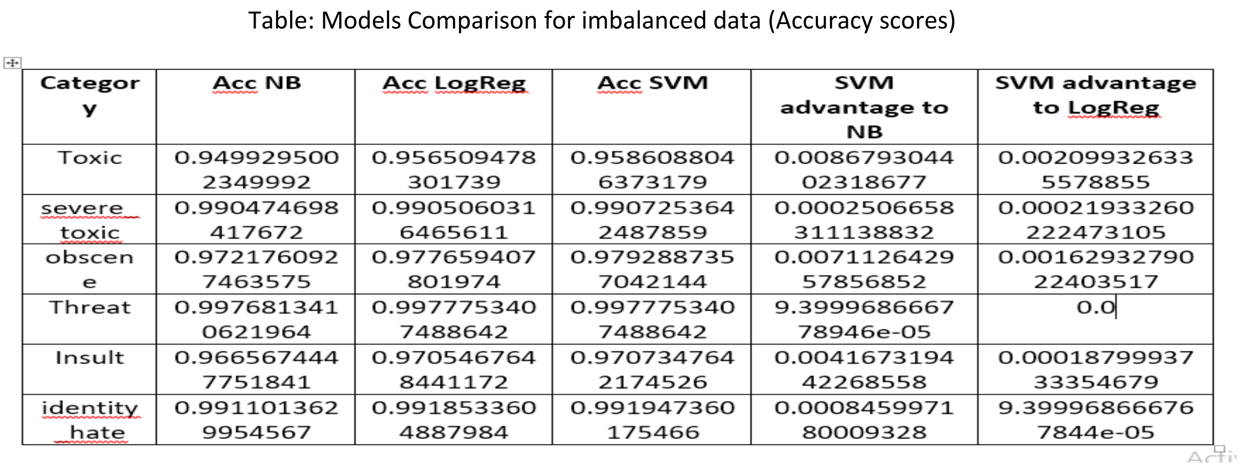
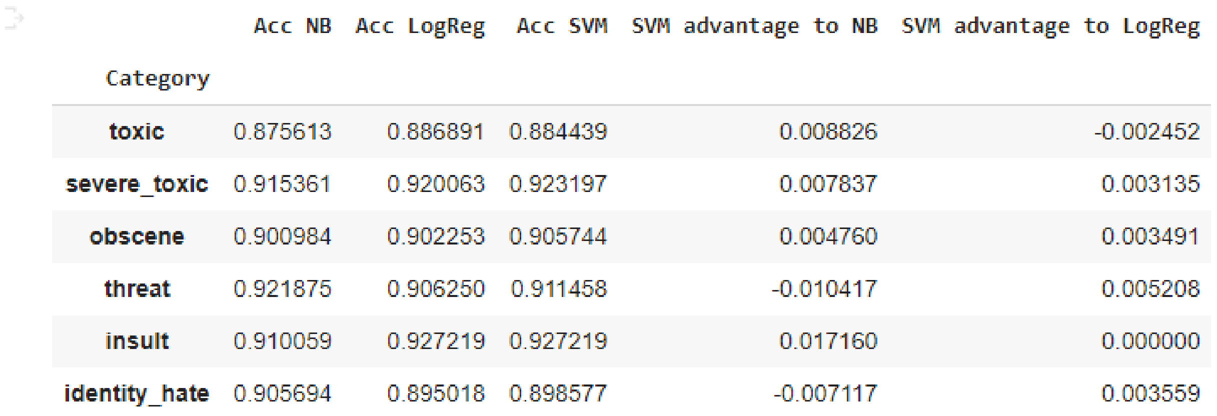


Table: Models Comparison for imbalanced data (Accuracy scores)

Category	Acc NB	Acc LogReg	Acc SVM	SVM advantage to NB	SVM advantage to LogReg
Toxic	0.9499295002349992	0.956509478301739	0.9586088046373179	0.008679304402318677	0.002099326335578855
severe toxic	0.990474698417672	0.9905060316465611	0.9907253642487859	0.0002506658311138832	0.00021933260222473105
obscene	0.9721760927463575	0.977659407801974	0.9792887357042144	0.007112642957856852	0.0016293279022403517
Threat	0.9976813410621964	0.9977753407488642	0.9977753407488642	9.399968666778946e-05	0.0
Insult	0.9665674447751841	0.9705467648441172	0.9707347642174526	0.004167319442268558	0.0001879993733354679
identity hate	0.9911013629954567	0.9918533604887984	0.991947360175466	0.000845997180009328	9.399968666767844e-05

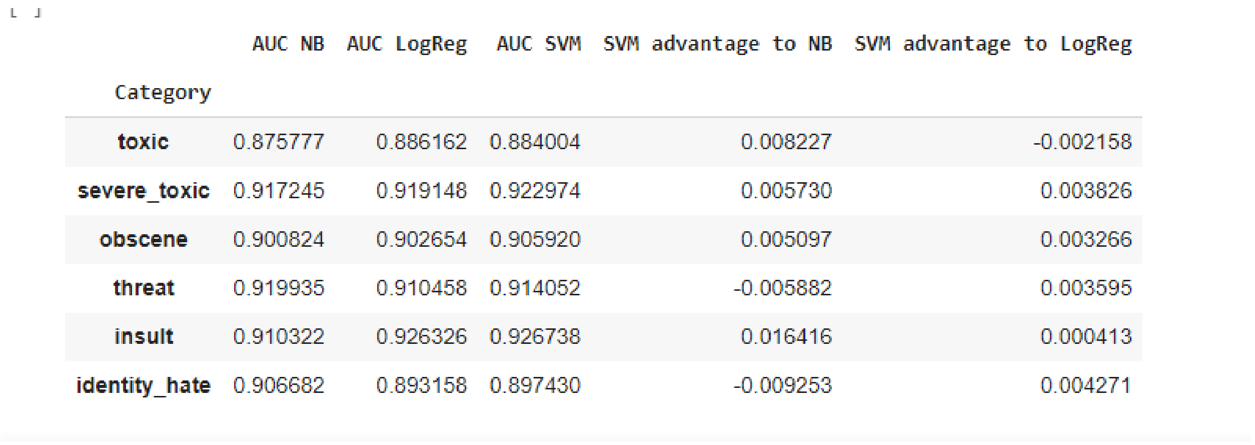
Similarly, we shall compare accuracy scores for each toxic categories for randomly undersampled data.. This is shown in the figure/table below.



Category	Acc NB	Acc LogReg	Acc SVM	SVM advantage to NB	SVM advantage to LogReg
toxic	0.875613	0.886891	0.884439	0.008826	-0.002452
severe_toxic	0.915361	0.920063	0.923197	0.007837	0.003135
obscene	0.900984	0.902253	0.905744	0.004760	0.003491
threat	0.921875	0.906250	0.911458	-0.010417	0.005208
insult	0.910059	0.927219	0.927219	0.017160	0.000000
identity_hate	0.905694	0.895018	0.898577	-0.007117	0.003559

Figure. Model comparison for randomly undersampled data (accuracy scores)

Similarly, we shall again show the model comparison based on AUC scores for randomly undersampled data. This is shown in the table/figure below.



Category	AUC NB	AUC LogReg	AUC SVM	SVM advantage to NB	SVM advantage to LogReg
toxic	0.875777	0.886162	0.884004	0.008227	-0.002158
severe_toxic	0.917245	0.919148	0.922974	0.005730	0.003826
obscene	0.900824	0.902654	0.905920	0.005097	0.003266
threat	0.919935	0.910458	0.914052	-0.005882	0.003595
insult	0.910322	0.926326	0.926738	0.016416	0.000413
identity_hate	0.906682	0.893158	0.897430	-0.009253	0.004271

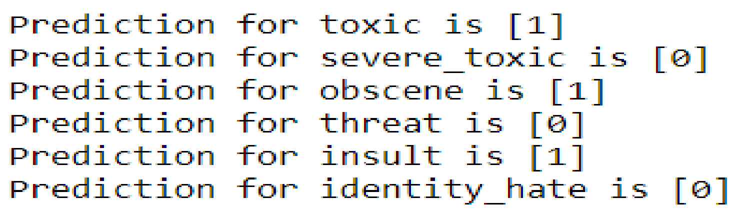
Figure. Model comparison for randomly undersampled data (AUC scores)_

5.5 Testing Model

Now that the model is trained and we have obtained a decent accuracy on average, let’s try out toxic online comment which is not part of our dataset, and compare its output label to the category that a human would give it. I am going to test our model on following toxic test.

"You little piece of shit do you really want me to smack on your fucking face goddamn moron"

Our best model, SVM predicted the following class labels for above test data:



```
Prediction for toxic is [1]
Prediction for severe_toxic is [0]
Prediction for obscene is [1]
Prediction for threat is [0]
Prediction for insult is [1]
Prediction for identity_hate is [0]
```

For above test data,

- SVM predicted toxic classes : *toxic, obscene and insult*
- LSTM predicted toxicity classes : *toxic, obscene and insult*
- SVM predicted toxic classes : *toxic, obscene and insult*

Since performance of all of our models are comparable, for above test data, we got same prediction

5.6 Qualitative Analysis

- Direct insults and comments are tagged toxic with respective labels with a very high confidence. For example, whenever, a sentence contains words such as "f*ck, shit, shut up, idiot".
- Toxic comments that had spelling errors were also labelled as toxic with a fairly high confidence. Examples: ". F*ck ing trollreasons", "if ytou think shesgreek your a morooon."
- Certain words were strong indicators of severe toxicity. For example - "d*ck", and "assh*le". This was problematic because a sentence about "Moby Dick" was classified as severely toxic when in reality it had no element of toxicity.
- The model fails to identify a statement as non toxic when the language used is strong. For example: Have you tried to fathom why reactions to you harsh?" The presence of strong words such as fathom, and harsh confused the model into incorrectly classifying the sentence as toxic.
- Statements that contain insults in foreign languages or unfamiliar slangs are not labelled accurately. For example: “chamars” in Hindi means “untouchables”, and the model could not pick on this as identity hate.

Chapter 6

Conclusion

6.1 Conclusion

Our best performing models were LSTM and Support Vector Machine with mean accuracy of 98.68% and 98.15% respectively for imbalanced data. The more complex and sophisticated model can be built by tuning hyperparameters but the results are good enough so we excluded them for now.

6.4 Future Works

- Tuning the hyperparameters
- Integrating BERT and ELMO contextual word embedding with our model.
- Recall and fi-score are somewhat not optimistic. It may be due to very low frequency distribution of toxic comment compared to non-toxic. We will be working on this in near future.
- More in depth error analysis

References

[1] Jiawei Han, Micheline kamber, Jian Pei. ‘’ Data Mining concepts and Techniques’’. Third Edition , Morgan kaufmann Publisher, 2012. (page no. 350 to 355 and 408 to 415)

[2] Durgesh Srivastava, Lekha Bhambhu. “Data Classification using Support Vector Machine”. Article in Journal of Theoretical and Applied Information Technology · February 2010.

[3] Christopher Olah ‘’ Understanding LSTM Networks’’. Colah’s blog, Posted on August 27, 2015.

[4] Sebastian Laverde, ‘’Automatic Classification of Sexual Harassment Cases’’. A NLP based project by Omdena.

[5] Chu, T & Jue, K. ‘’Comment Abuse Classification with Deep Learning’’.

[6] Kevin Khieu and Neha Narwal. ‘’Detecting and Classifying Toxic Comments’’

[7] Stanford CS244N lectures series on Natural Language Processing with Deep Learning. (lecture 5 and 6)

[8] Analytics Vidhya’s article on ‘’An Intuitive Understanding of Word Embedding: From Count Vector to Word2Vec’’.

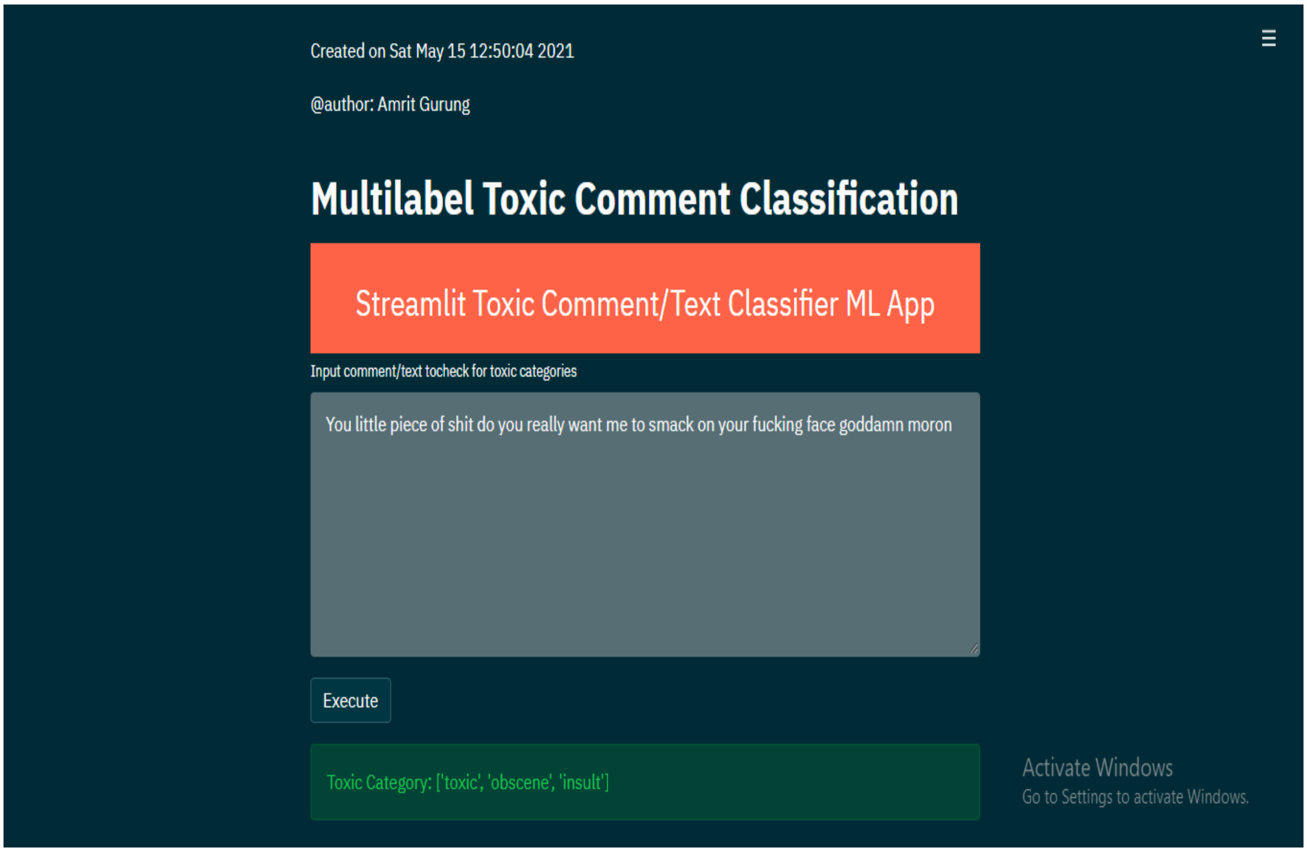
[9] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated Hate Speech Detection and the Problem of Offensive Language,” dissertation, 2017.

[10] E. Thompson, “Facebook partners with Ontario university on 'global network' to counter rise in online hate | CBC News,”.

[11] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, “Hate speech detection: Challenges and solutions,”

[12] Pinkesh Badjatiya, Shashank Gupta, ManishGupta, and Vasudeva Varma.Deep learning for hate speech detection int weets. In Proceedings of the 26th International Conference on World Wide Web Companion, pages (759, 760),2017.

Appendix


Created on Sat May 15 12:50:04 2021

@author: Amrit Gurung

Multilabel Toxic Comment Classification

Streamlit Toxic Comment/Text Classifier ML App

Input comment/text tocheck for toxic categories

You little piece of shit do you really want me to smack on your fucking face goddamn moron

Execute

Toxic Category: ['toxic', 'obscene', 'insult']

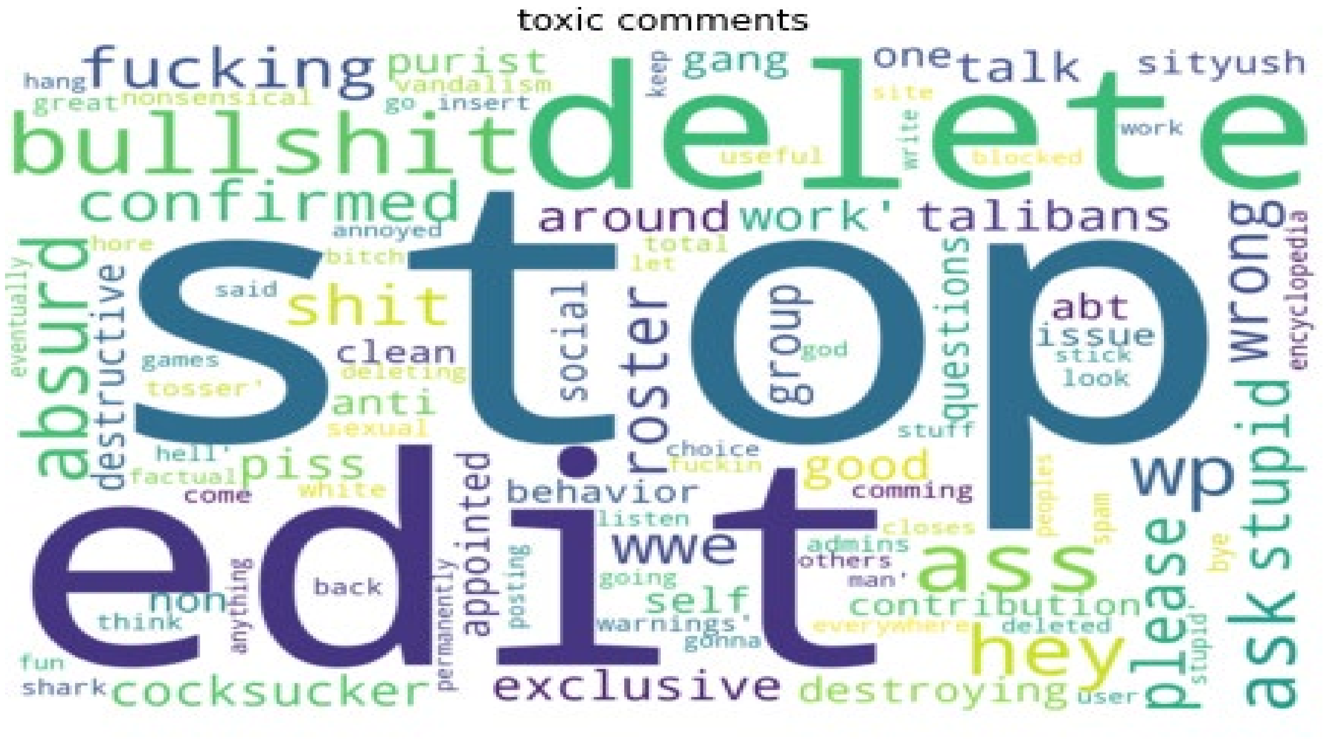
Activate Windows
Go to Settings to activate Windows.

Figure. Web Application

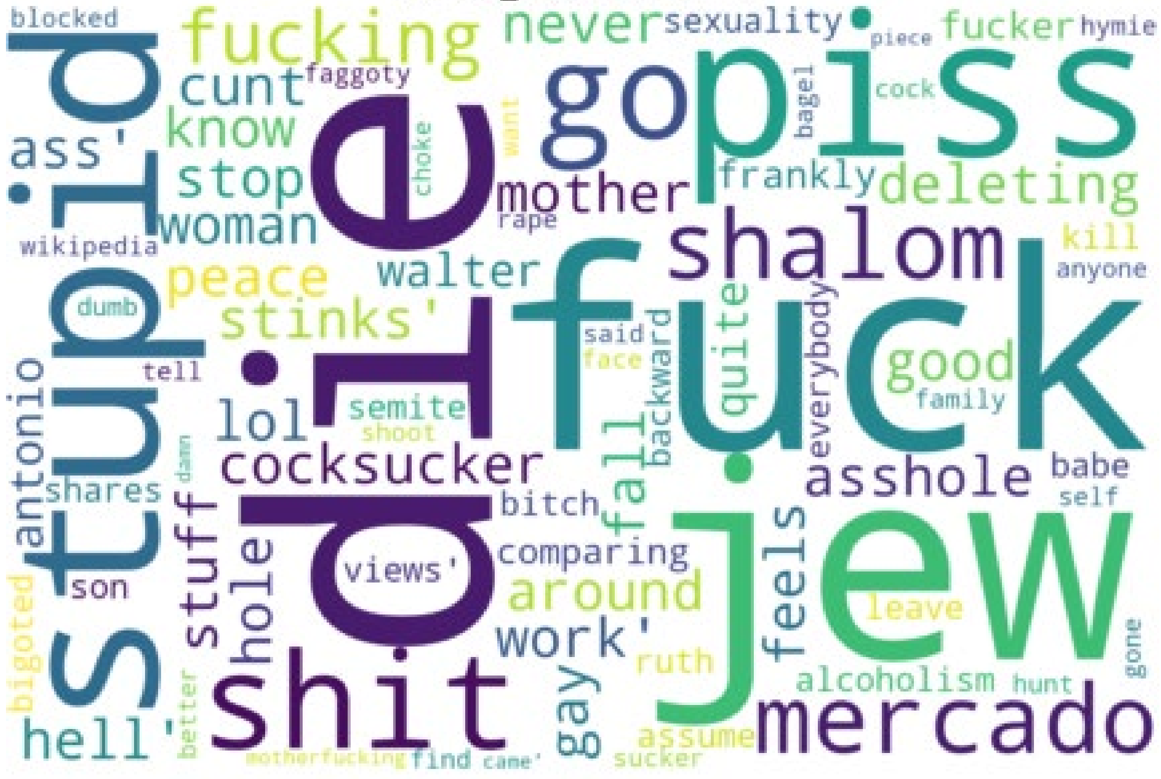
Word Cloud

Word Cloud is the novelty visual representation of text data, typically used to depictkeyword metadata on websites, or to visualize free form text. The importance and frequecny of important words areshown with font size or color.

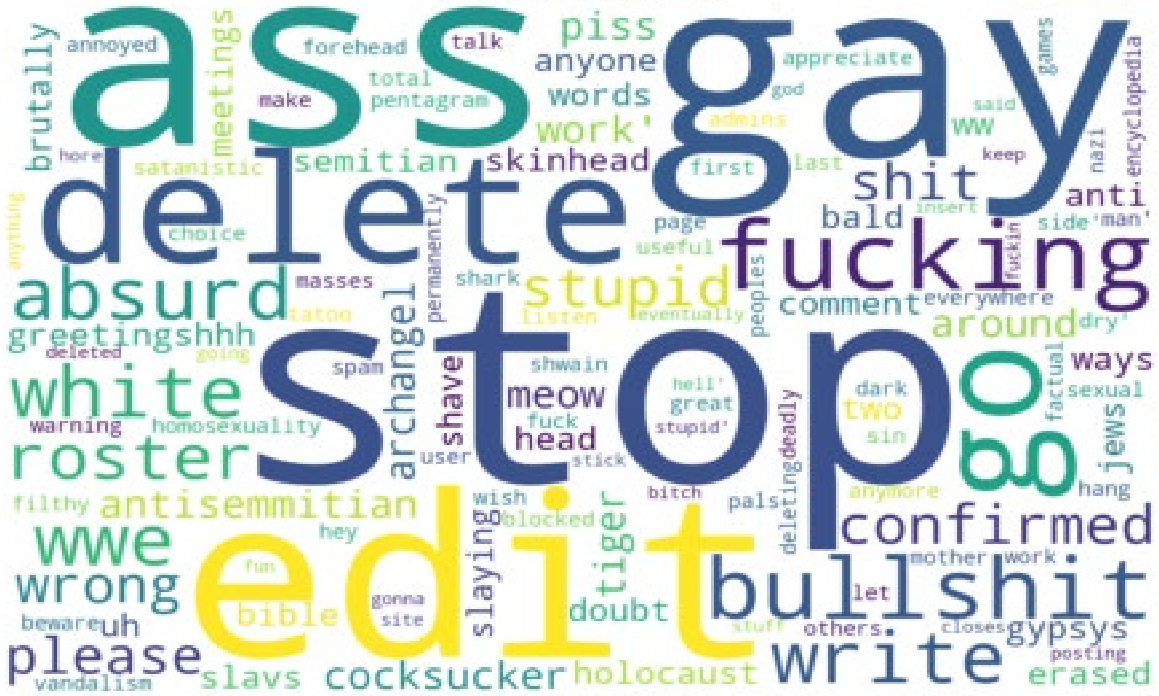
We are presenting word cloud for each toxic categories. The size of word in the below word clouds represent importance and frequency of that words in the particular toxic category.



severe_toxic comments

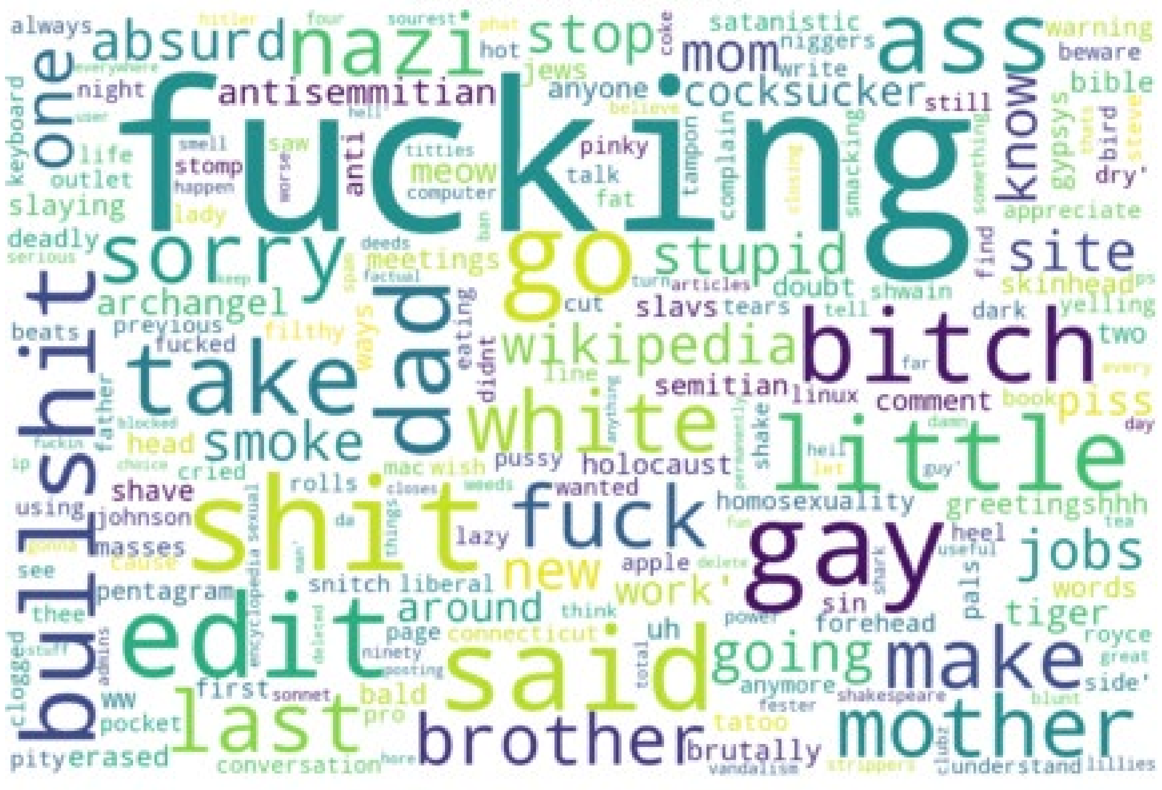


insult comments



ass
ass
die
die
must die
jim wales
wales must
die jim
kill
kill
fucking
talk page
going kill
kill
going
going murder
die wikipedia
di edie
supertr live
live forever
die di
forever respect
wikipedia die
pathetic fool
en wikipedia
lifetime ban
respect supertr
blank talk
supertr die
murder going
fool supertr
edie die
real stephen
die pathetic
wikipedia org
org wiki
stephen hawkinghttp
hawkinghttp en
fuck
wiki real
steal password
password take
dust filter
kill youcaltlas
block blank
page steal
kill block
block ban
fuckin kill
beavis butthead
simpson niggaz
niggaz moonshine
jessica simpson
ban lifetime
rvv fuckin
page block
butthead jessica
take block
youcaltlas continue
continue rvv
unblock lifetime
block dust
piece shit
bail beavis
caltlas rvv
fuck unblock
filter rvv
filter caltlas
wanta fuckin
bastard
hope
asshole
cunt deserve
fuckin lifetime
ban kill
kill gonna
gonna kill
faggot
stupid
bitch
death
life
know
shit
mother
hate
find
stop
never
face
dead
hell
house
raped
little
rape
suck
please
hello
edit
think
give
right
threat comments

obscene comments



identity_hate comments

fucking
jew
gay
shit
said
nazi
wikipedia
think
bitch
take
sorry
little
anyone
fuck