Name:- Pranit Arunrao Jagtap **Reg no:-200968214** **Project:- Toxic Comment Classification**

Phase-1

Problem Statement

Online forums and social media platforms have provided individuals with the means to put forward their thoughts and freely express their opinion on various issues and incidents. In some cases, these online comments contain explicit language which may hurt the readers. Comments containing explicit language can be classified into myriad categories such as Toxic, Severe Toxic, Obscene, Threat, Insult, and Identity Hate. The threat of abuse and harassment means that many people stop expressing themselves and give up on seeking different opinions.

To protect users from being exposed to offensive language on online forums or social media sites, it is important to flag comments and block users who are found guilty of using unpleasant language. Several Machine Learning/Deep Learning models can be developed and deployed to filter out the unruly language and protect internet users from becoming victims of online harassment and cyberbullying.

Meta Data of Dataset

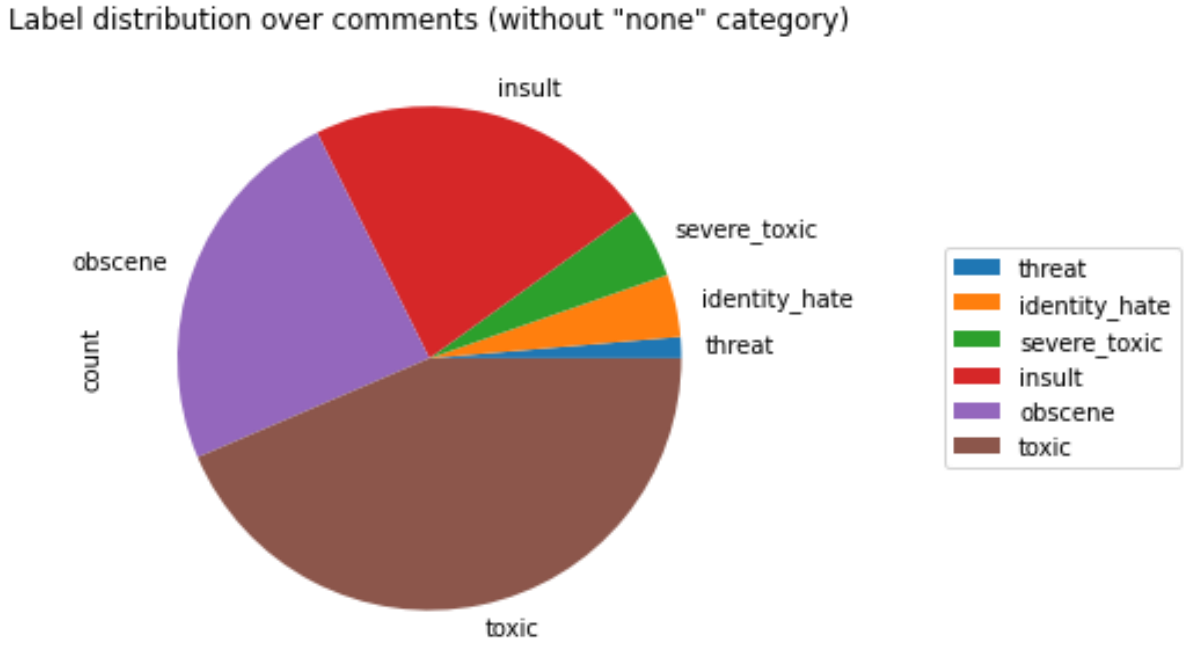
The dataset used was Wikipedia corpus dataset which was rated by human raters for toxicity. The corpus contains comments from discussions relating to use pages and articles dating from 2004-2015. The dataset was hosted on Kaggle by Jigsaw as a challenge. The comments were manually classified into following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

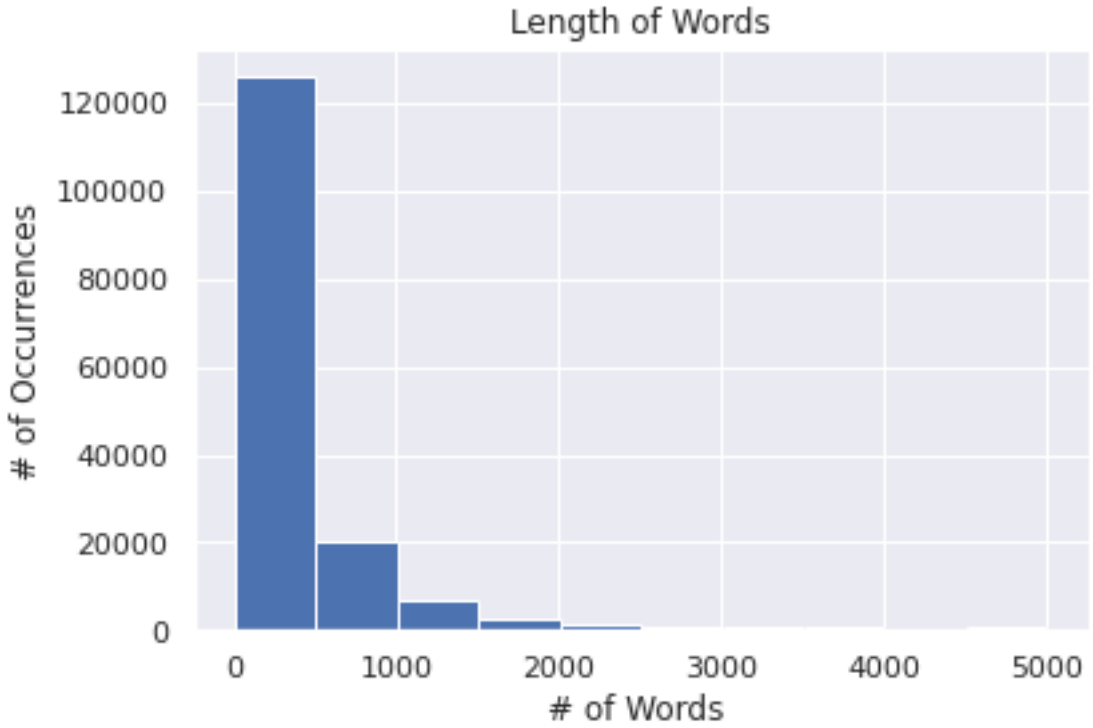
1. Toxic
2. Severe toxic
3. Obscene
4. Threat
5. Insult
6. Identity hate

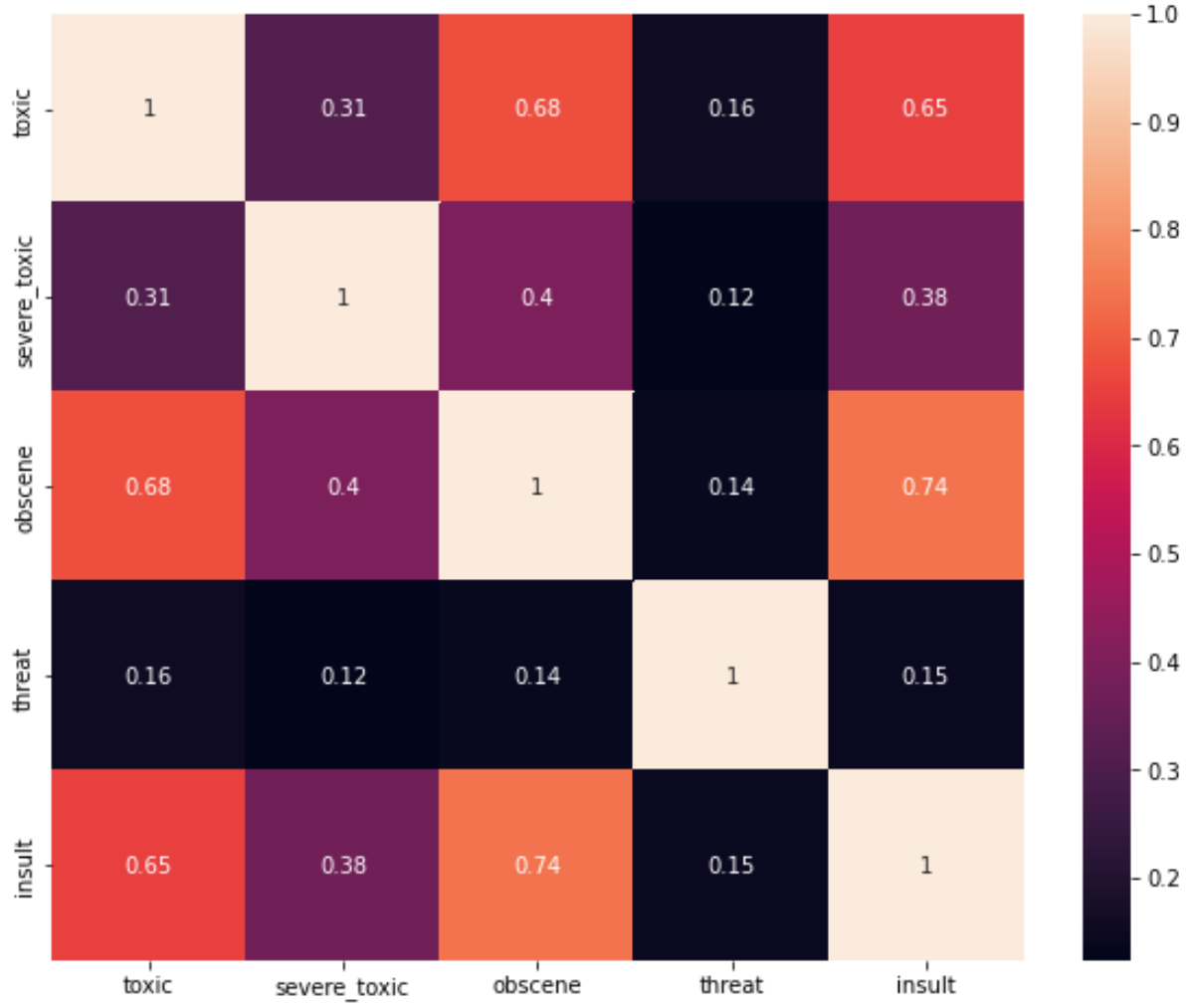
train.csv - the training set, contains comments with their binary labels

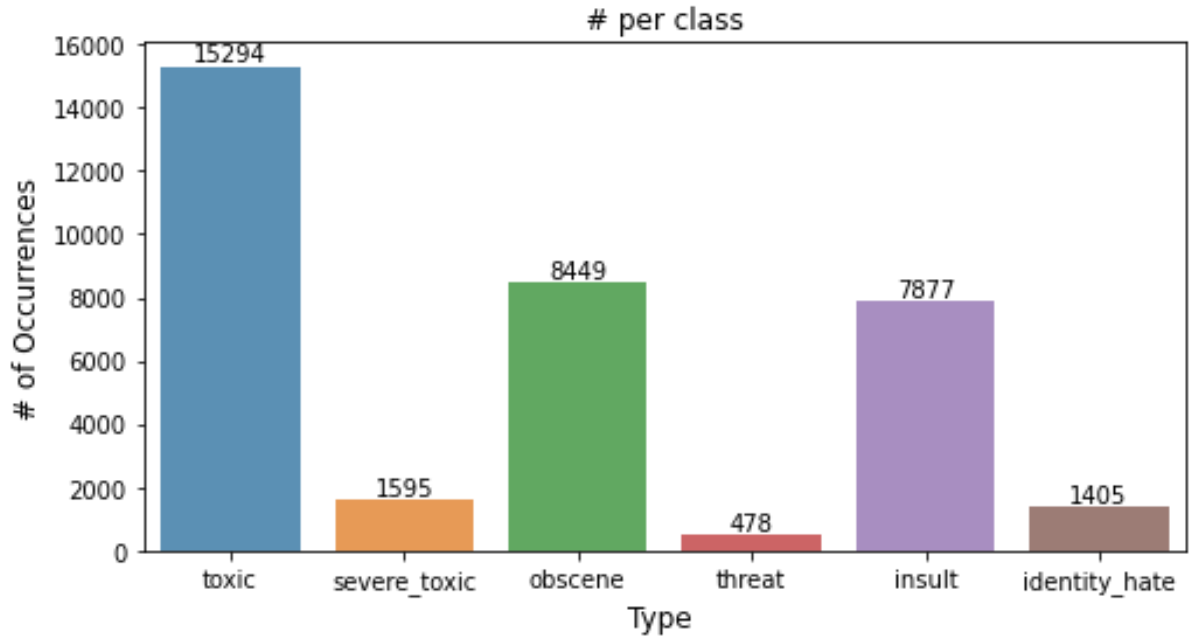
test.csv - the test set, you must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.

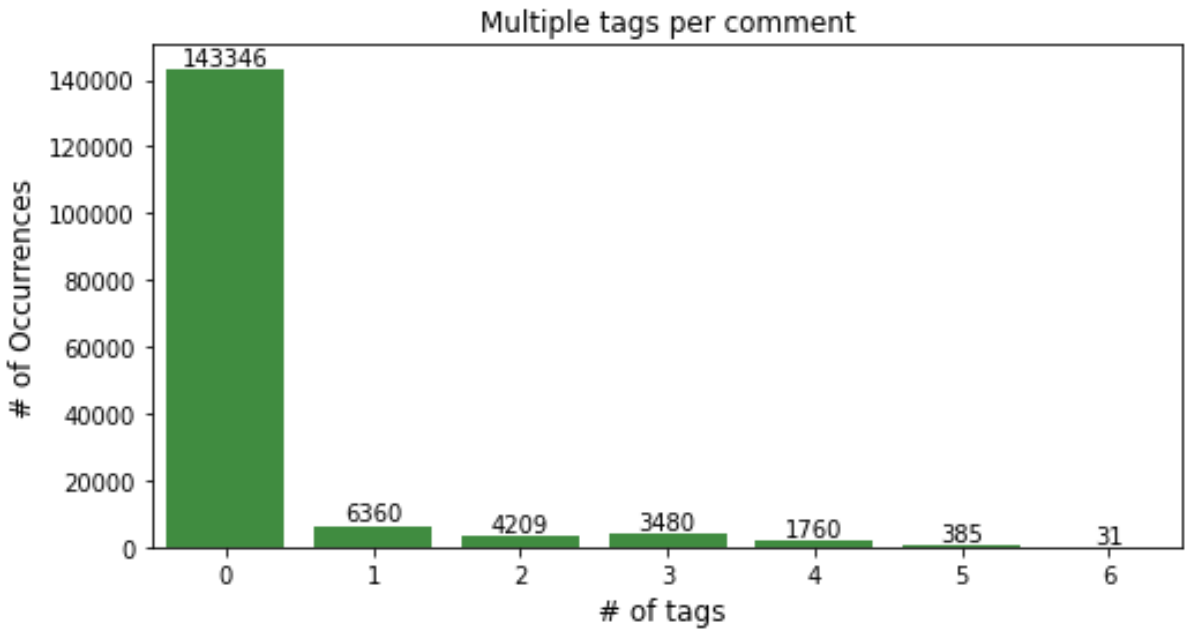
Exploratory Analysis






toxic
severe_toxic
obscene
threat
insult
1
0.31
0.68
0.16
0.65
0.31
1
0.4
0.12
0.38
0.68
0.4
1
0.14
0.74
0.16
0.12
0.14
1
0.15
0.65
0.38
0.74
0.15
1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


per class
of Occurrences
Type
15294
1595
8449
478
7877
1405
toxic
severe_toxic
obscene
threat
insult
identity_hate
16000
14000
12000
10000
8000
6000
4000
2000
0



The dataset has large number of tuples with obscene, toxic and insult category when compared to other labels

Data set has very high number of normal texts which are not offensive

There is high correlation between insult & obscene, obscene & toxic telling us that they generally occur together in an comment

Pre-processing Pipeline

1. Removing stop words
2. Lowercase conversion
3. Removing Special Characters excluding a few
4. Tokenization

Project Objectives

To build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate.

Phase-2

Literature Review

Serial No	Author	Type of Paper	Published	Models implemented
1	Pallam Ravi, Hari Narayana Batta, Greeshma S, Shaik Yaseen	Article	Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456- 6470, Volume-3 | Issue-4, June 2019, pp.24-27 7 Toxic Comment Classification (ijtsrd.com)	Logistic Regression, Random Forrest
2	Kevin Khieu, Neha Narwal	Report	Published in Stanford 6837517.pdf (stanford.edu)	Long Short Term Memory(LSTM), Convolution Neural Network (CNN)
3	P.Vidyullatha , Satya Narayan Padhy ,Javvaji Geetha Priya , Kakarlapudi Srija ,Sri Satyanjani Koppisetti	Research Article	Published in Turkish Journal of Computer and Mathematics Education Identification of Toxic Comment	Support Vector Machine (SVM)

Pros and Cons of the Models:-

SVM:-

Pros:-

- SVM works relatively well when there is a clear margin of separation between classes.
- More effective in high dimensional spaces.
- Effective in cases where the number of dimensions is greater than the number of samples.
- Memory efficient

Cons:-

- SVM algorithm is not suitable for large data sets.
- Does not perform very well when the data set has more noise i.e. target classes are overlapping.

- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

Logistic Regression

Pros:

- Simple to implement
- Feature scaling not needed: Does not require input features to be scaled (can work with scaled features too, but doesn't require scaling
- Tuning of hyperparameters not needed.

Cons:

- Poor performance on non-linear data
- Poor performance with irrelevant and highly correlated features
- High reliance on proper presentation of data. All the important variables / features should be identified for it to work well.

Random Forrest

Pros:

- Good Performance on Imbalanced datasets : It can also handle errors in imbalanced data
- Handling of huge amount of data: It can handle huge amount of data with higher dimensionality of variables.
- Good handling of missing data: It can handle missing data very well. So if there is large amount of missing data in your model, it will give good results.
- Little impact of outliers: As the final outcome is taken by consulting many decision trees so certain data points which are outliers will not have a very big impact on Random Forest.
- No problem of overfitting: In Random forest considers only a subset of features, and the final outcome depends on all the trees. So there is more generalization and less overfitting.
- Useful to extract feature importance (we can use it for feature selection)

Cons:

- Features need to have some predictive power else they won't work.
- Predictions of the trees need to be uncorrelated.
- Appears as Black Box: It is tough to know what is happening. You can at best try different parameters and random seeds to change the outcomes and performance.

LSTM

Pros:

- Constant error backpropagation within memory cells results in LSTM's ability to bridge very long time lags
- For long time lag problems such as those discussed in this paper, LSTM can handle noise, distributed representations, and continuous values. In contrast to finite state automata or

hidden Markov models L$ TM does not require an a priori choice of a finite number of states.In principle it can deal with unlimited state numbers.

- LSTM generalizes well - even if the positions of widely separated, relevant inputs in the input sequence do not matter, quickly learns to distinguish between two or more widely separated occurrences of a particular element in an input sequence, without depending on appropriate short time lag training exemplars.
- There appears to be no need for parameter fine tuning. LSTM works well over a broad range of parameters such as learning rate, input gate bias and output gate bias. However, a large learning rate pushes the output gates towards zero, thus automatically countermanding its own negative effects.

Cons:

- LSTMs became popular because they could solve the problem of vanishing gradients. But it turns out, they fail to remove it completely.
- They require a lot of resources and time to get trained and become ready for real-world applications.
- LSTMs get affected by different random weight initialization and hence behave quite similar to that of a feed-forward neural net. They prefer small weight initialization instead.
- LSTMs are prone to overfitting and it is difficult to apply the dropout algorithm to curb this issue. Dropout is a regularization method where input and recurrent connections to
- LSTM units are probabilistically excluded from activation and weight updates while training a network.

CNN

Pros:

- Takes into account local structure - since a convolution is usually taken over adjacent word embeddings, information contained in adjacent words is learned effectively.
- They perform well on classification tasks, like sentiment analysis, spam detection or topic categorization.
- CNNs are very fast.

Cons:

- In images, local structure matters – useful semantic information is contained in adjacent pixels. However, in sentences, words often don't need to be adjacent to be related. This makes CNNs perform badly for part of speech tagging and entity extraction.

Models Shortlisted:-

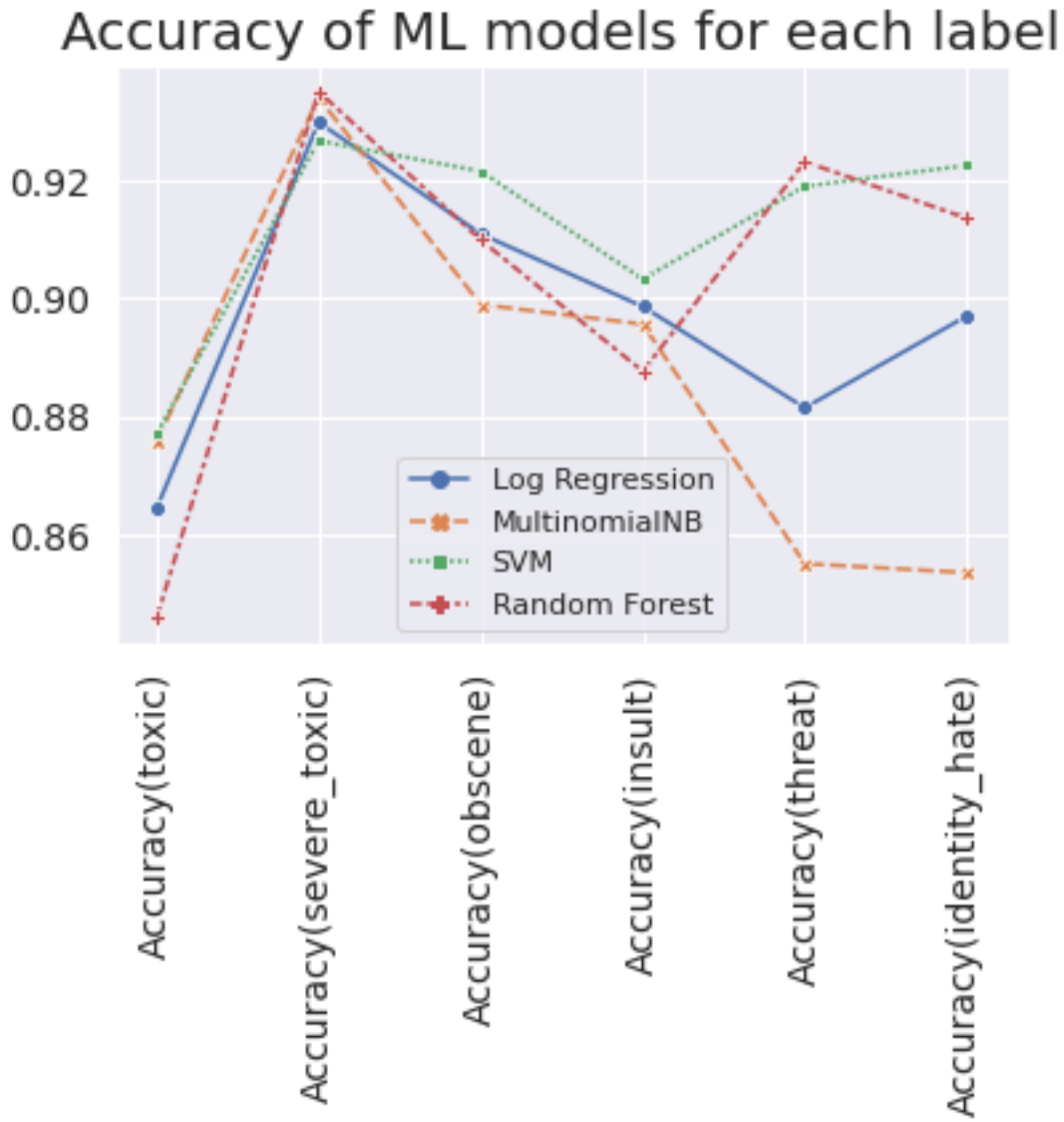
1. **SVM using Naïve Bayes**
2. **LSTM**

Baseline Model:- SVM Using Naïve Bayes

Result and Analysis:-

For ML models for each label

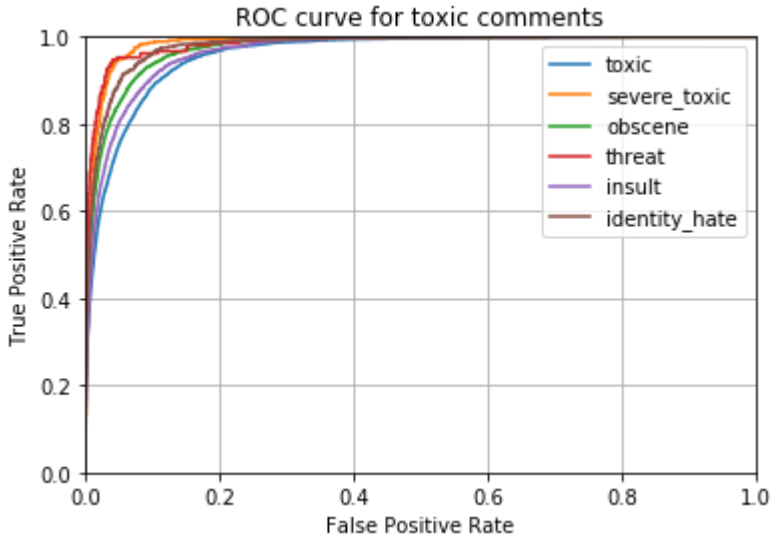
	Accuracy(toxic)	Accuracy(severe_toxic)	Accuracy(obscene)	Accuracy(insult)	Accuracy(threat)	Accuracy(identity_hate)
Log Regression	0.864333	0.929990	0.911000	0.898667	0.881450	0.897059
MultinomialNB	0.875667	0.934169	0.899000	0.895667	0.854951	0.853416
SVM	0.877000	0.926855	0.921667	0.903333	0.919107	0.922676
Random Forest	0.845667	0.935214	0.910000	0.887667	0.923291	0.913662



Comparing ML models with LSTM on Overall Accuracy

	Accuracy
Log Regression	0.897083
MultinomialNB	0.885478
SVM	0.911773
Random Forest	0.902584
LSTM	0.994047

LSTM ROC curve



We can Observe that LSTM gives a much better Accuracy when compared to Baseline model like SVM and other ML models.

Although Training time for LSTM is relatively larger than other models

Parameters Reasoning:-

Max length was set to 200 because from EDA we concluded that maximum sentences were of length around 50 and 200

We begin our defining an Input layer that accepts a list of sentences that has a dimension of 200. Next, we passed it to Embedding layer, where we project the words to a defined vector space depending on the distance of the surrounding words in a sentence.

We can make use of the output from the previous embedding layer which outputs a 3-D tensor of (None, 200, 128) into the LSTM layer. We will receive a Tensor shape of (None, 200, 60), where 60 is the output dimension we have defined.

Before we could pass the output to a normal layer, we needed to reshape the 3D tensor into a 2D one. We reshape carefully to avoid throwing away data that is important to us, and ideally we want the resulting data to be a good representative of the original data.

Therefore, we used a Global Max Pooling layer which is traditionally used in CNN problems to reduce the dimensionality of image data. In simple terms, we go through each patch of data, and we take the maximum values of each patch. These collection of maximum values will be a new set of down-sized data we can use.

With a 2D Tensor in our hands, we pass it to a Dropout layer which indiscriminately "disable" some nodes so that the nodes in the next layer is forced to handle the representation of the missing data and the whole network could result in better

generalization.

We set the dropout layer to drop out 10%(0.1) of the nodes.

we feed the output into a Sigmoid layer. The reason why sigmoid is used is because we are trying to achieve a binary classification(1,0) for each of the 6 labels, and the sigmoid function will fit the output between the bounds of 0 and 1.

We have set our model to optimize our loss function using Adam optimizer, define the loss function to be "binary_crossentropy" since we are tackling a binary classification.

Conclusion:-

In order to get a better idea of the performance of each model, we mainly look at the test accuracy, but also to the other metrics (Training accuracy, validation accuracy). We notice that the LSTM has the best results for all categories. Although Training LSTM takes a longer time comparatively. Using the model for a larger data set might give a better idea on the generalization of the model . Mainly on dataset extracted from Twitter and other social media platforms where people tend to use symbols to mask their abusive words and use a lot of short forms to convey messages.