Introduction :-

In today's world, Social networking sites are the biggest platform. People spend hours on it, where they meet with known and unknown people, and make conversation and comment on varying topics. Topics include politics, religion, economy, crimes etc. These are very good platform to share idea and information in nano second. For the developing and developed nation, they play a very good role in boosting economy on one side. But another side, Many time these conversations become arguing that take a face of assault, molest, threat etc. With the increase of network, there is need to build a system that can classify these comments and conversations and take suitable action before the conversation become more aggressive and harmful.

Natural language processing is used to deal with such problem. It is used to extract information from unstructured data in the form of text which is available at emails, chats, web page, social media and more. Here, we will try to predict the intention of person by their comment with the help of NLP and Machine learning.The idea of my project and the dataset has been taken from kaggle.

Under the name of Toxic comment classification challenge, it aims to identify and classify online toxic comments. Data-set is provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are: toxic, severe_toxic, obscene, threat, insult, identity_hate.

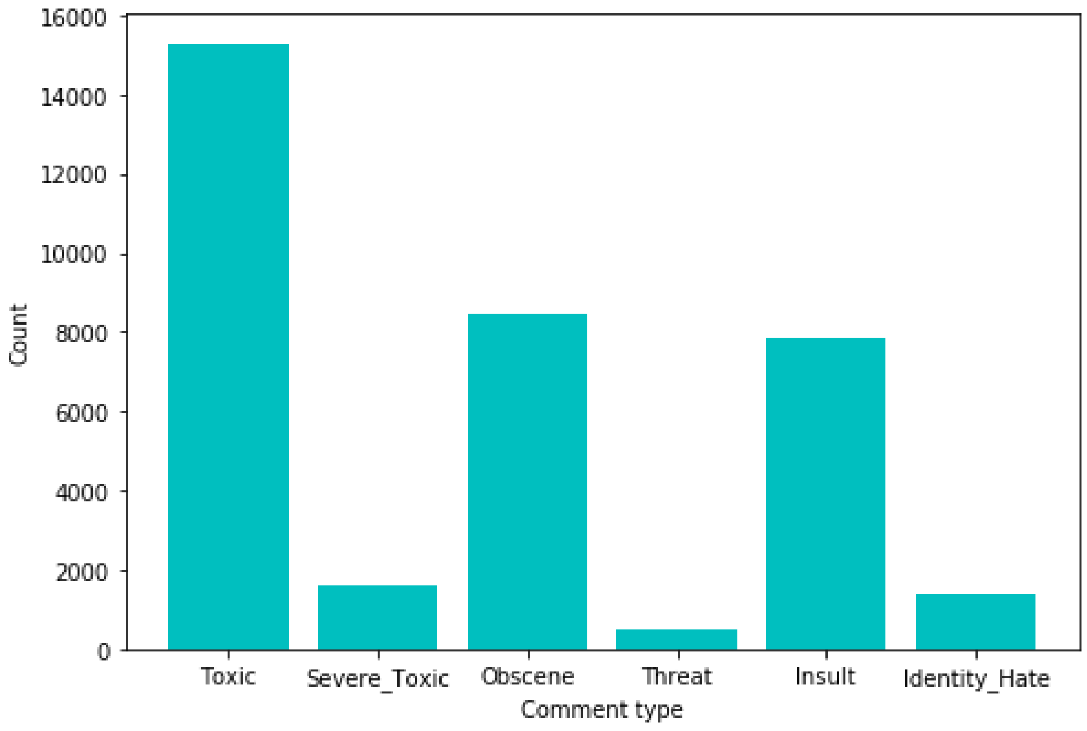
Methodology:-

In the data-set, each comment belong to one or more of the following categories - toxic, severe-toxic, obscene, threat, insult or identity-hate with approximate probabilities or discrete values (0/1). This is clearly a Multi-label classification problem.

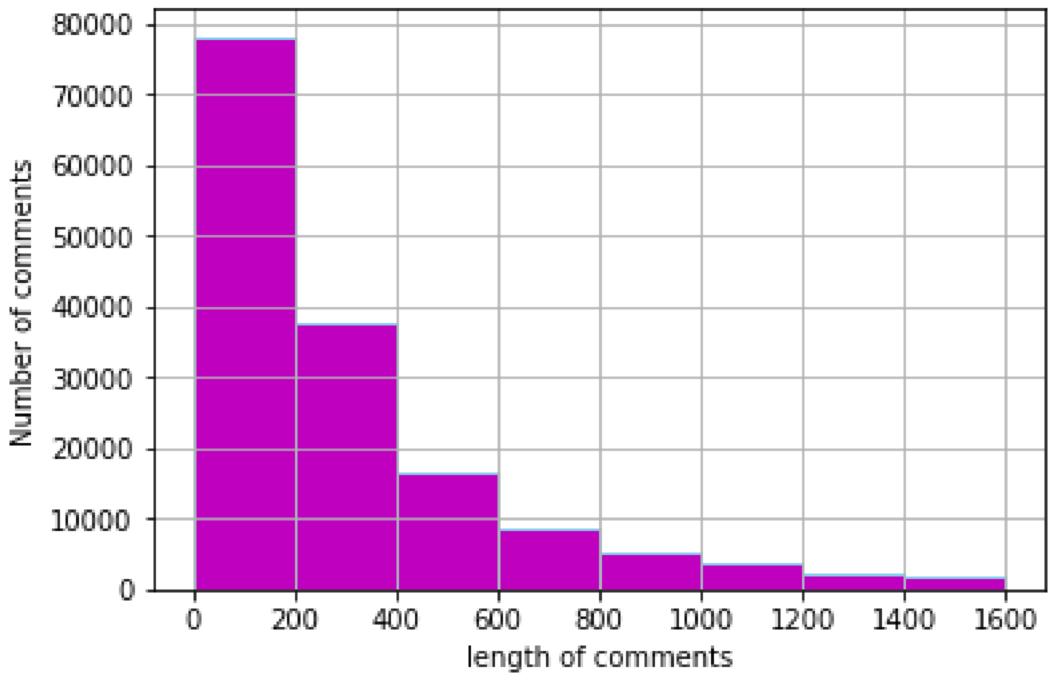
A multi-label classification problem differs from a multi-class classification problem (in which each sample can only be assigned to one of the many-labels). Hence in our problem each comment can belong to more than one label for eg, a comment can be obscene, toxic and insult, all at the same time.

Data Exploration and Visualization:-

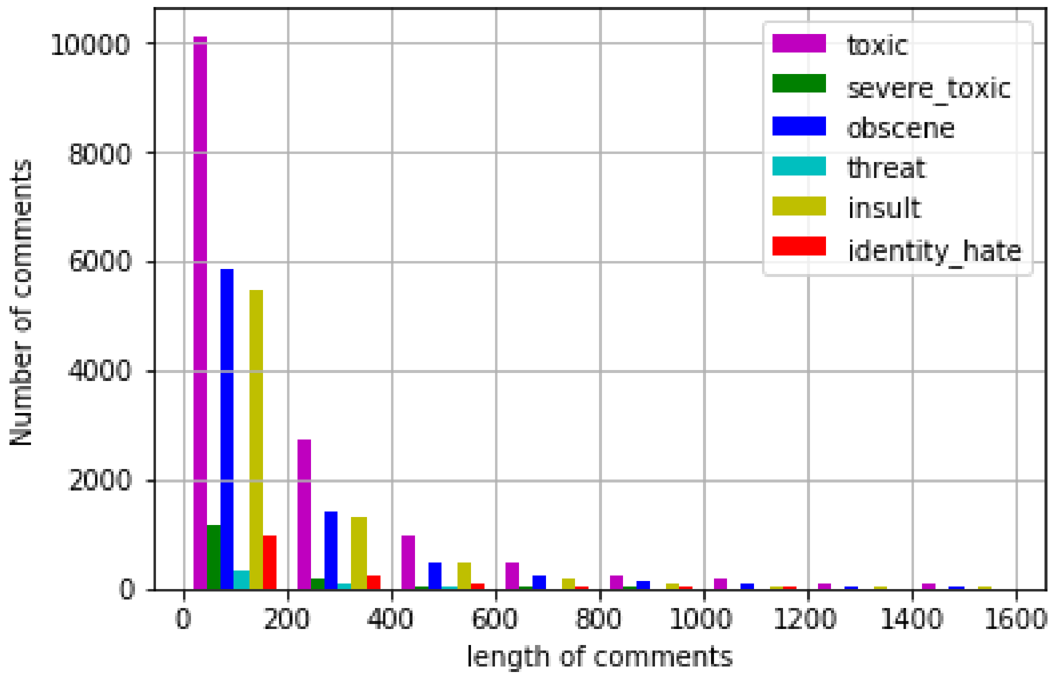
Before moving to multi-label classification, we have to done some exploration to convert unstructured data in structured form. We had a data-set of 1,59,571 samples of comments along with their labels. There is no null comment and label in the given data. I observed that every 1 in 10 samples was toxic, every 1 in 20 samples was obscene and insulting, every 1 in 100 samples was severe-toxic and identity-hate but threat samples was very rare, every 1 in 300 sample was threat. Below show a bar representation of count of each label:-



From the below visualisation we can observe that comments have varying lengths from within 200 upto 1600. The majority of comments have length upto 200, and as we move towards greater lengths, the number of comments keeps on falling. Since including very long length comments for training will increase the number of words manifold, it is important to set a threshold value for optimum results.



From the second visualization, we can observe the number of words falling under the six different outcome labels toxic, severe_toxic, obscene, etc along with their lengths. Here also similar to the first plot, we observe that most of the abusive comments have lengths under 200, and this number falls with the length of comments.



Based on these plots, taking comments having lengths upto 400 for training is a good estimation of our data and can be expected to give acceptable results on testing later. Hence before preprocessing, we will be removing all comments with length more than 400 which will serve as our threshold.

Now, we have 12,873 samples of toxic, 1,357 samples of severe toxic, 7,230 samples of obscene, 413 samples of threat, 6,782 samples of insult and 1,181 samples of identity hate comments out of 1,15,910 comments.

Data Preprocessing and Feature Extraction:-

In previous phase, we already understood the structure and distribution of data. In this phase, we will focus on cleaning and extracting feature from data.

For data cleaning, a function is created that first convert the comment in lower case, then remove all punctuation from the comment and then we apply tokenization. After this we remove all stopwords and then we apply lemmatization and stemming.

Let discuss each operation in detail and their importance:-

1. Lowercase:- Let there are two words - “slap” and “Slap”, both words have same meaning but computer consider them as two different word, which increase the features unnecessarily.

2. Remove Punctuation:- there are punctuation like “#$%&()+
“ in the comment that people use many time unnecessarily which will not contribute any thing to predict the type of comment.

3. Tokenization :- It means conversion of sentence into words. It is necessary because we make the list of unique words in the text corpus called vocabulary. Then we can represent each sentence or document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary.

4. Removing Stopword :- For tasks like text classification, where the text is to be classified into different categories, stopwords are removed or excluded

from the given text so that more focus can be given to those words which define the meaning of the text.

5. Lemmatization:- It is a process by which inflected forms of words are grouped together to be analyzed as a single aspect. It is a way of using the intended meaning of a word to determine the “lemma”. Examples of Lemmatization are that “run” is a base form for words like “running” or “ran” or that the word “better” and “good” are in the same lemma so they are considered the same.

6. Stemming :- It is quite similar to Lemmatization in that it groups words together, but unlike lemmatization it takes a word and refers it back to its base or root form. In fact, on the best examples I’ve come across to describe it involves refereeing Stemming back to its base form. “Stems”, “Stemming”, “Stemmed”, “and Stemtization” are all based on the single word “stem”.

After completing the cleaning part of data, it’s time to extract feature from data. In text processing, words of the text represent discrete, categorical feature. Here, we have to use any encoding technique that can map textual data to real valued vectors. One of the simplest technique to numerically represent text is bag of words. Under bag of words, we are going to extract feature in two different features:- countvectorizer and Term Frequency-Inverse Document Frequency.

- CounterVectorization is a SciKitLearn library tool takes any mass of text and returns each unique word as a feature with the count of number of times that word occurs. While this can generate a lot of features that are some extremely useful parameters that help avoid that including stop_words, n_grams, and max_features.

- TF-IDF reveals what words are the most discriminating between different bodies of text. It is particularly, helpful if you are trying to see the difference between words that occur a lot in one document, but fail to appear in others allowing you interpret something special about that document.

Term Frequency (TF) = (Number of times term t appears in a document)/(Number of terms in the document)

Inverse Document Frequency (IDF) = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.

TF-IDF = TF*IDF

Model Implementation :- Our problem is a multi label classification problem. For such type of problem, problem transformation methods are used. Problem transformation method include binary relevance, classifier chain and more. Here we use Binary relevance method.

In this case an ensemble of single-label binary classifiers is trained, one for each class. Each classifier predicts either the membership or the non-membership of one class. The union of all classes that were predicted is taken as the multi-label output. This approach is popular because it is easy to implement, however it also ignores the possible correlations between class labels. This is a simple approach but does not work well when there's dependencies between the labels.

I will be using support vector machine - SVC with radial basis kernel, Naive bayes - MultinomialNB and Ensemble - Random forest classifier with 20 estimators (number of trees in forest) algorithms with binary relevance method.

Evaluation metrics:- In a multi-label classification problem, we evaluate on the basis of accuracy and hamming loss.But, we can't simply use our normal metrics to calculate the accuracy of our predictions. For that purpose, we will use accuracy score metric. This function calculates subset accuracy meaning the predicted set of labels should exactly match with the true set of labels. A misclassification is no longer a hard wrong or right. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them, i.e., predicting two of the three labels correctly this is better than predicting no labels at all.

In the same way, In multilabel classification, the Hamming loss is different from the subset zero-one loss of multiclass classification. The zero-one loss considers the entire set of labels for a given sample incorrect if it does not entirely match the true set of labels. Hamming loss is more forgiving in that it penalizes only the individual labels. Hamming-Loss is the fraction of labels that are incorrectly predicted, i.e., the fraction of the wrong labels to the total number of labels.

Out of the total dataset(1,15,910 samples) used, 70% (80,000 samples) was used for training and 30% (35,910 samples) for testing. Each testing dataset was labelled and hence for each algorithm using the predictions and labels, calculation of metrics such as accuracy and hamming-loss was done. The final results have been compiled on the basis of values obtained by algorithmic models in accuracy and hamming loss combined.

Tools used:-

Programming Setup:- The models are trained on Anaconda which provide a Jupyter notebook environment that is a free web application for interactive computing. With it, users can create and share documents with a live code, develop and execute code, as well as present and discuss task results.

Programming Language:- Python is a high level language which is interpreted in other words bytecode-compiled and dynamic language.

Modules and Dependencies:-

1. Numpy - For multi-dimensional arrays and matrices operation
2. Pandas - For data manipulation and analysis.
3. Matplotlib - For creating static, animated, and interactive visualizations
4. NLTK - The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
5. Random - For generating random numbers and for doing random file access.
6. Re - It provides full support for Perl-like regular expressions.
7. Scikit-learn - It contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

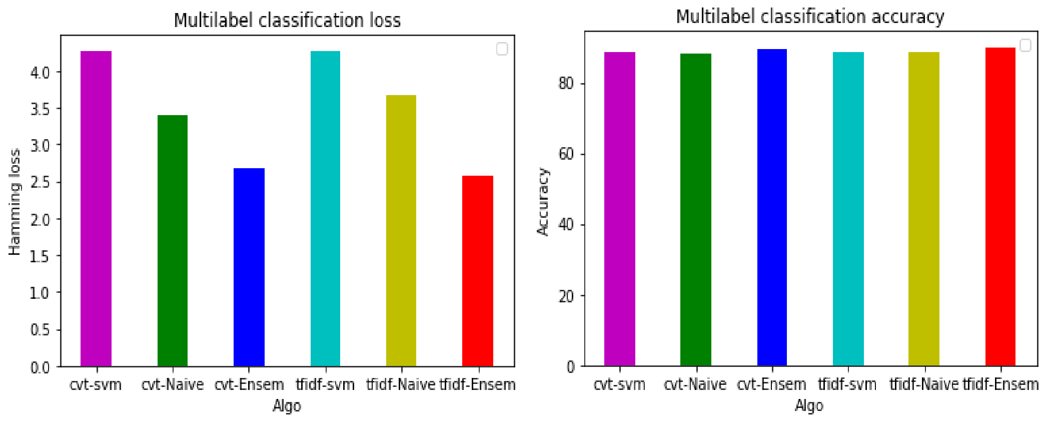
Result:-

As, we have two different feature, and we apply binary relevance model using all three algorithms on both features separately. Here, we create a table:-

Feature	Algorithm	Accuracy	Hamming Loss
Count vectorizer	SVM-SVC	88.42	4.27
Count vectorizer	Naive bayes - MultinomialNB	88.05	3.39
Count vectorizer	Ensemble-Random Forest Classifier	89.33	2.67
Tf-Idf	SVM-SVC	88.42	4.27
Tf-Idf	Naive bayes - MultinomialNB	88.68	3.67
Tf-Idf	Ensemble-Random Forest Classifier	89.90	2.58

Conclusion:-

The hamming-loss and accuracy of different models used, can be plotted in two bar plots as below :



From the above figure, we found that there is not any huge difference in accuracy and hamming loss. But, it can observe that binary relevance model using random forest classifier on TF-IDF feature give higher accuracy and less hamming loss as compare to other.

Although we have tried quite a number of parameters in refining my model, there can definitely exist a better model which gives greater accuracy. In our project, we have used binary relevance method. One can use classifier chain method, label powerset method and more-problem transformation method.