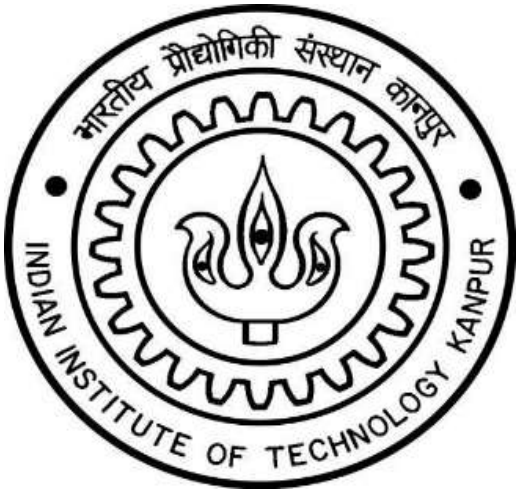
Hate Speech / Toxic Comment Detection



2021-22

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

DATA MINING PROJECT REPORT

INSTRUCTOR:

Prof. Arnab Bhattacharya

COMPLETED BY:

Shwetank Singh – 201422 (shwetanksr20@iitk.ac.in)

Akash Kumar Sharma – 201261 (pawan20@iitk.ac.in)

Nistha Shah – 201358 (nistha20@iitk.ac.in)

Shivani Gupta – 201412 (shivi20@iitk.ac.in)

Contents

Acknowledgments

It is a great pleasure for us to express respect and deep sense of gratitude to our supervisor Dr. Arnab Bhattacharya, Associate, Computer Science Department, Indian Institute of Technology, Kanpur for his vision, expertise, guidance, enthusiastic involvement and persistent encouragement during the planning and development of this work.

We also gratefully acknowledge his painstaking efforts in thoroughly going through and improving the manuscripts without which this work could not have been completed.

We thank our project members for providing all the facilities, help and encouragement for carrying out this project work. We are also obliged to our parents for their moral support, love, encouragement and blessings to complete this task. Finally, we are indebted and grateful to the Almighty for helping us in this endeavour.

Abstract

With the push towards the decline in data rates and growth of telecommunication networks in the last few years, there has been a huge spike in the number of internet users. Penetration of high bandwidth connection into the remotest part of the world has enabled quite a few users to come up online and express their opinions and thoughts. However, another side of coin is that it has led to increase in instances of hate speech and bullying which not only holds back certain users from opening up freely with ingenious ideas but also spreads negativity, depression, communal/racial hatred and at times can trigger riots.

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.

So it becomes imperative for social media ventures to ensure that their platform is free from toxicity to encourage free flow of information and opinions.

To protect users from being exposed to offensive language on online forums or social media sites, companies have started flagging comments and blocking users who are found guilty of using unpleasant language. Several Machine Learning models have been developed and deployed to filter out the unruly language and protect internet users from becoming victims of online harassment and cyberbullying.

1. Introduction

1.1 Overview

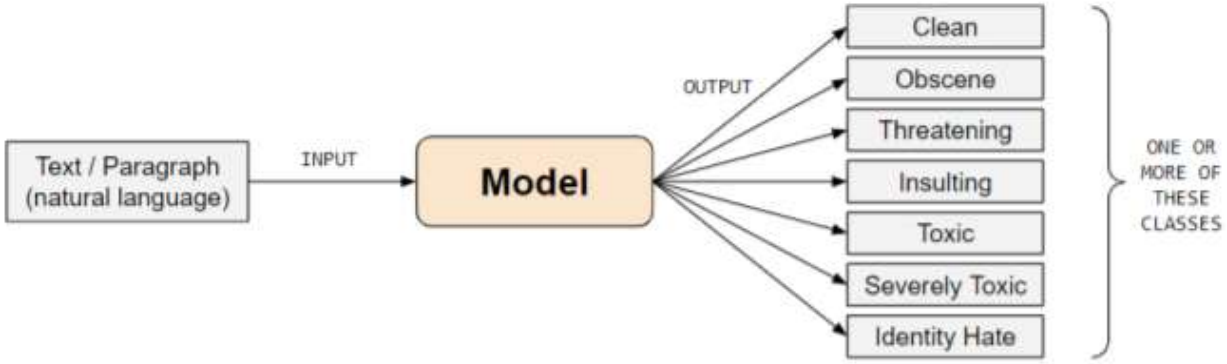


Figure 1: Introduction

Social networking sites have evolved to become an effective medium of communication all over the world. People can freely share their opinions and ideas with anyone. However, many people are found spreading verbal violence in the form of hate speeches and toxic comments. Thus, it has become a serious challenge for the social networking sites to control the spread of toxic comments. The task of hate speech or toxic comment detection can be posed as a multi-class and multi-label classification task. Companies are building models to automatically flag such content on the sites. In this paper, we take a collection of posts obtained from various social media sites and build models to classify those posts into six categories namely, toxic, severe toxic, threat, identity hate, obscene and insult. The advantage of this type of data is that these comments represent a true sample of the content present on the social media sites. We began by performing analysis and visualization of the dataset. Since the dataset contained the real comments posted on social media, the data contains noise. We had to pre-process the data to remove any outliers or noise that were present in the dataset. We initially tested the performance of classical models namely, support vector machines, and logistic regression on this task. We compared the performance of all the models using the mean AUC ROC score as the performance metric.

1.2 Report Structure

The structure of the report is as follows:

- ✓ In Section 1, we describe the problem statement, and give the description of the dataset.
- ✓ In Section 2, we describe the visualization and analysis of the data.
- ✓ In Section 3, we describe the various methods used in cleaning and pre-processing the dataset.
- ✓ In Section 4, we explain the performance of various models on this task.
- ✓ In Section 5, we give a comparative analysis of the results obtained from the models.
- ✓ In Section 6, finally, we give the conclusion of our report.

1.3 Problem Description

We are given a set of tweets/comments/posts which may have been collected from a variety of websites including Facebook, twitter, Instagram, etc. The task involves detecting and flagging

the comments which may involve display of hate or vulgarity. In particular, the model classifies each of the toxic comments into one of the six categories- toxic, severely toxic, threatening, insulting, obscene, identity hate. In practice, if the comment is not toxic, then it is labeled as clean which is added as an additional class label.

The problem involves processing of comments which are written in Natural Language, hence belong to the category of NLP problems. Further the source of data is internet open platforms and social media websites so the comments are from the general public and hence are highly noisy, unclean and unstructured. This necessitates rigorous text cleaning and preprocessing pipeline to obtain useful information and to structure the data.
As we can see the above description this problem is a multiclass classification as well as multilabel classification problem.

• **Multiclass classification problem**: This type of classification involves putting each data point or example into one of several possible categories as opposed to a binary classification problem in which each example can be classified into only one of the two categories.
• **Multilabel classification problem**: This type of classification involves examples such that each example can belong to multiple categories and not necessarily only one category. A multi label classification problem can be viewed as a generalised version of the multiclass classification problem in which there is no restriction over how many classes can a training example belong to.

Figure 2: Multi-class as well as a multi-label classification problem

Now according to our problem description, a comment can belong to any of the seven categories. That's why the problem is a multiclass classification problem. Secondly we note that a comment may be offensive in multiple ways. A comment which is toxic may also be obscene. So a comment can belong to several categories all at the same time and hence, it's a mutlilabel classification problem too. One possible variation of our problem is to classify each comment as just being toxic or not which pretty much reduced it to a binary classification problem. But 4 this is not a permissible reduction as it severely restricts the domain of application of our model. We may, in practice, desire a website in which some degree of insult (say) is allowed but identity hate is not allowed at all. If we restrict our model to binary classifiers, then it will not be usable in these cases. We therefore explore some other problem transformation methods described in forthcoming sections.

1.4 Dataset Description

```
train.head(10)
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Figure 3: Dataset

The dataset for the problem has been made publicly available by Conversational AI. In the snapshot of the dataset we can see it has approximately 160k training examples. Corresponding to each example we have a set of labels and each label can take two values 0 or 1 depending on whether that particular comment should belong to the class label or not. The preparation of the dataset has been done using crowdsourced annotation platforms. This means different examples have been labeled by different users across the internet and hence there is an element of subjectivity in labeling. What may be offensive for me may not be offensive for someone else! This makes the dataset particularly vulnerable to judgement bias. On further exploring the dataset we realise that toxicity is influenced by some other factors too. One such example is the way a comment is rendered. If rendered in a particular way on a particular screen width, the comment takes the form of offensive symbols which are obviously toxic. But if the annotator had been using 5 some other screen size then it won't be detected by him. In such cases he may think the comment is spam, but it will be classified as clean nonetheless. As another example, we have some comments which are very long and detailed, possibly expressing someone's opinion on some topic. Some of these comments have been classified as toxic without any reason. Thus it is impossible to classify 100% examples correctly as some labels may not be correct themselves.

1.5 Applications

1. Social Networking: With increase in user base of the popular sites it has become practically infeasible for humans to read every post and comment to decide what is toxic and what not and take down the offensive posts. Machine learning models can be put to use by social networking websites to flag the offensive posts. The process of taking down may happen in a two-step process in which the model first flags the toxic comment and a human only analyses the flagged content and decides if the post should be removed from the website. This certainly makes the task a lot easier than to analyse every post by human only.

2. Online meetings / webinars: The COVID era has seen a dramatic increase in the meetings and classes being conducted over the internet platforms. Unlike physical meetings where it is highly

unlikely that someone might abuse the speaker, it is quite common to see people abusing on the online meeting platform as it gives them assurance of being 'hidden' by using anonymous accounts.

3. Chatbot Training: Chatbots like Google Assistant, Alexa and Bixby can be put to use in a wide range of applications. These chatbots collect the user data which are then again used to train them and customise to user specific data. But this also opens the possibility of abusing the system by training them on toxic data and in turn making them learn and replicate the insulting remarks. Toxic comment analysis can be used to filter out any toxic data input by the user thereby preventing the model from learning from toxic data.

4. Threatening messages: If the model perfects the art of detecting offensive comments, then it is possible to directly report threatening messages to 6 concerned authorities taking quick and timely action. As an example, if a kidnapper demands a ransom from someone, this message can be automatically reported to the Police triggering corrective measures.

2. Data Visualization

Data Visualization is an important part of Data Mining. It helps us get visual insights about the dataset, such as some striking features or patterns in the dataset, which can help us choose the appropriate machine learning algorithms to apply. We first plotted a pie chart, showing the relative amount of hate tags present in the dataset, and found a massive class imbalance. “Toxic” tags were the highest, whereas tags labeled with “threat” had the lowest count. Also, plotting a bar graph after adding a column for “clean” tags, it was found that a significantly large amount of comments was clean.

toxic	15294
severe_toxic	1595
obscene	8449
threat	478
insult	7877
identity_hate	1405

Figure 4: hate_tag_count

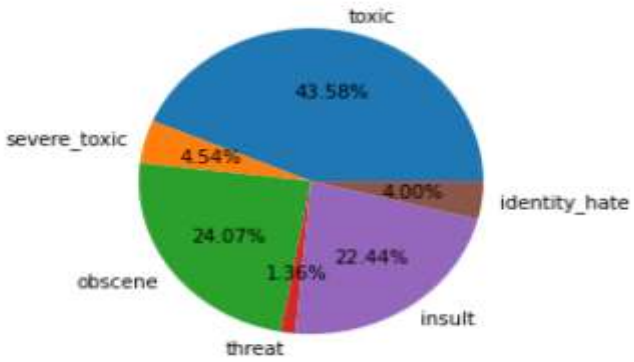


Figure 5: Relative amount of hate tags present in the dataset

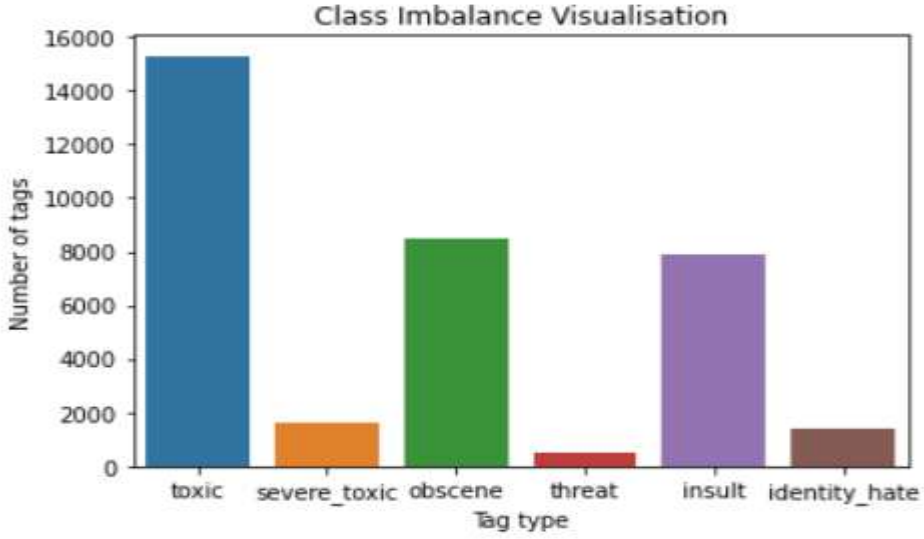


Figure 6: Class imbalance visualisation

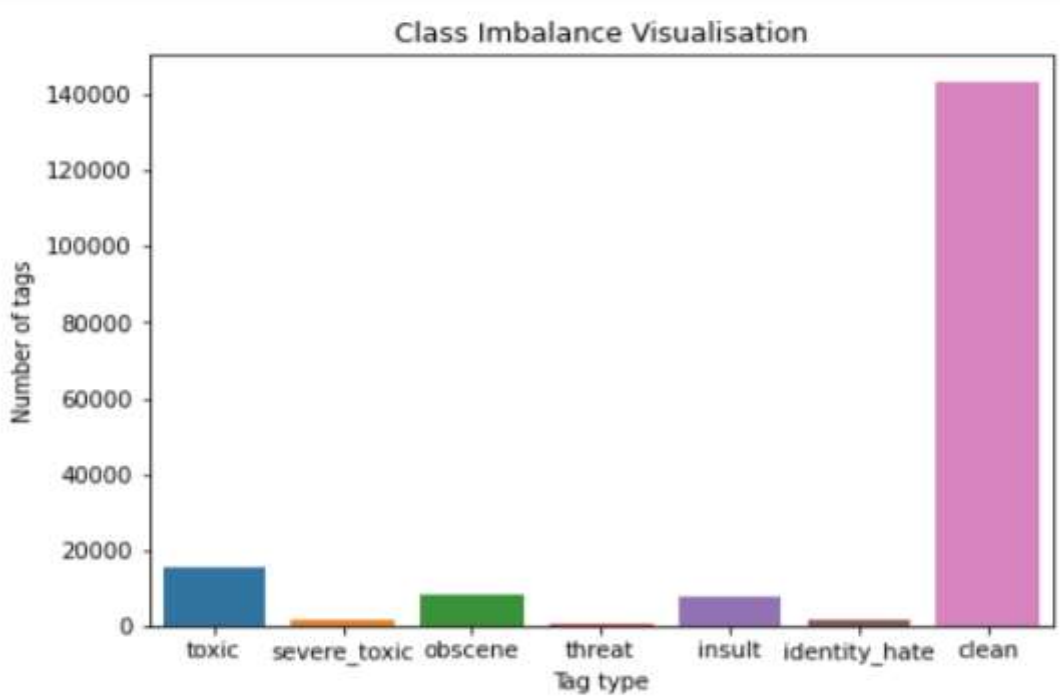


Figure 7: Class imbalance visualisation with clean data

2.1 Visualization of the length of comments

	count_sent	count_word	count_unique_word	count_letters
count	159571.00000	159571.000000	159571.000000	159571.000000
mean	3.52074	67.273527	48.097323	394.073221
std	5.96225	99.230702	54.436443	590.720282
min	1.00000	1.000000	1.000000	6.000000
25%	1.00000	17.000000	16.000000	96.000000
50%	2.00000	36.000000	31.000000	205.000000
75%	3.00000	75.000000	59.000000	435.000000
max	313.00000	1411.000000	816.000000	5000.000000

Figure 8: Count

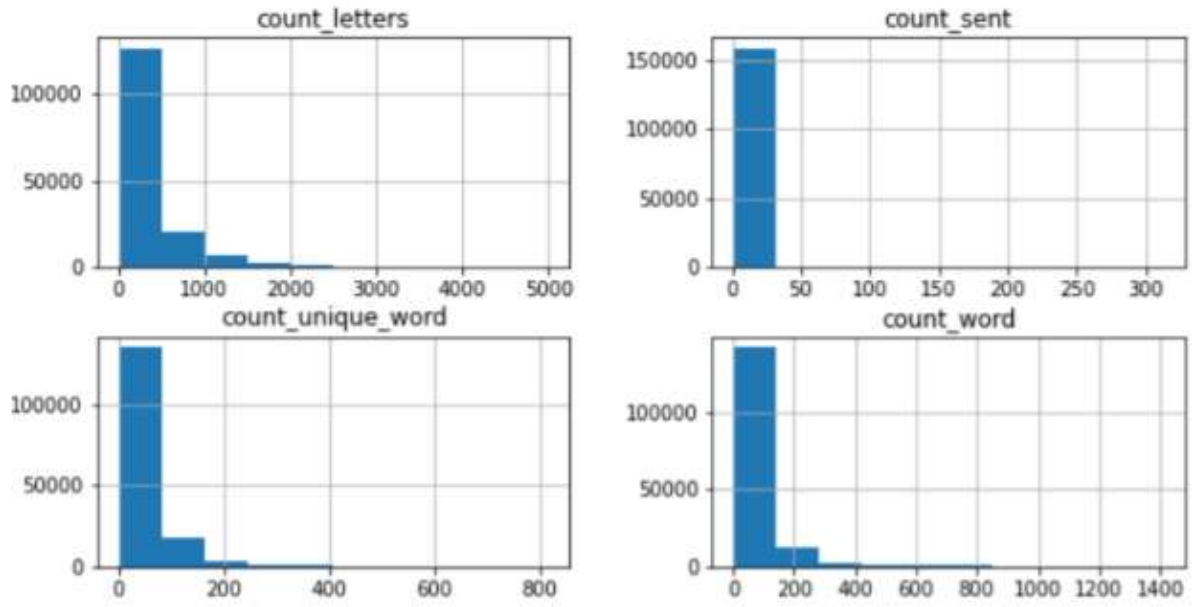
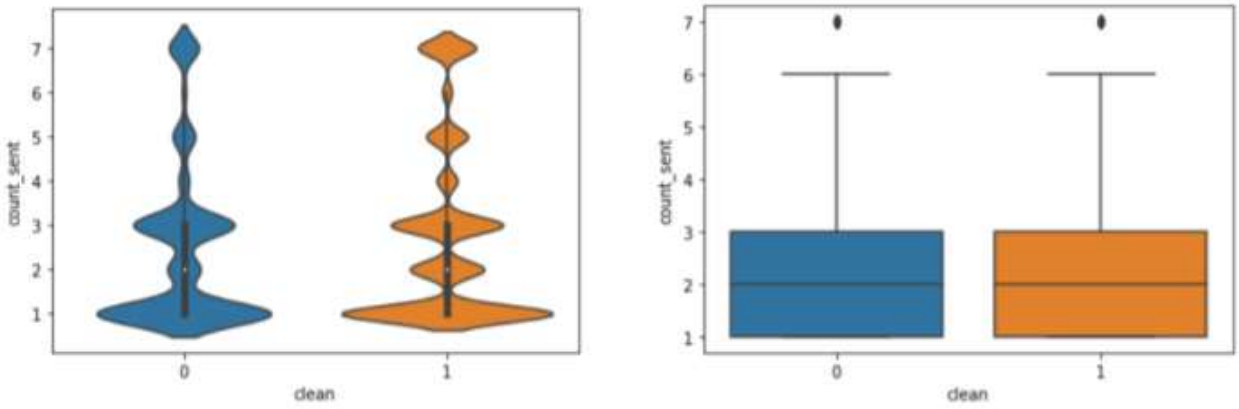


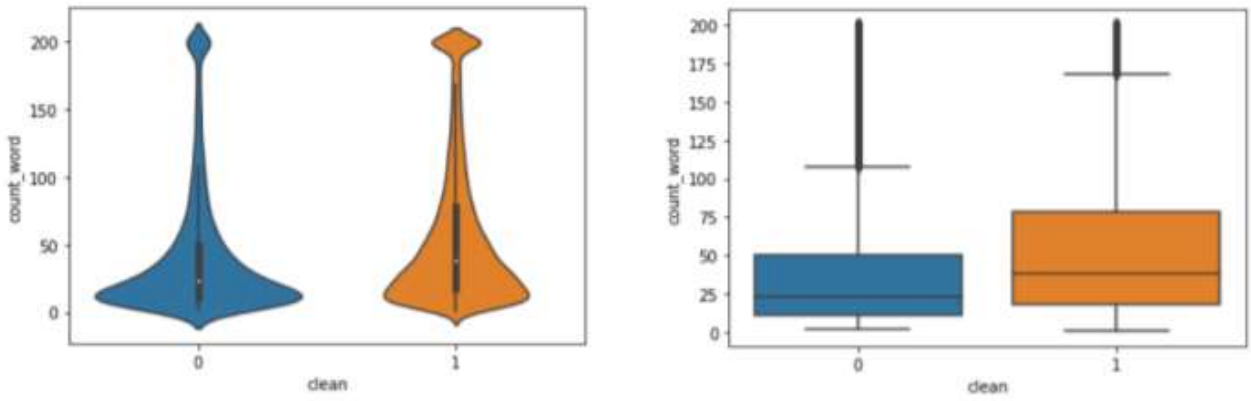
Figure 9: A lot of comments have very less count of sentences, words, or letters

2.2 Correlation between length of comments and toxicity



(a) Violin plot (b) Box plot

Figure 10: There is not much correlation between the count of sentences and the comment's toxicity



(a) Violin plot (b) Box plot

Figure 11: There is a slight correlation between the count of words and toxicity - less is the count of the word, the more is the comment's toxicity.

2.3 Correlation between spam comments and toxicity

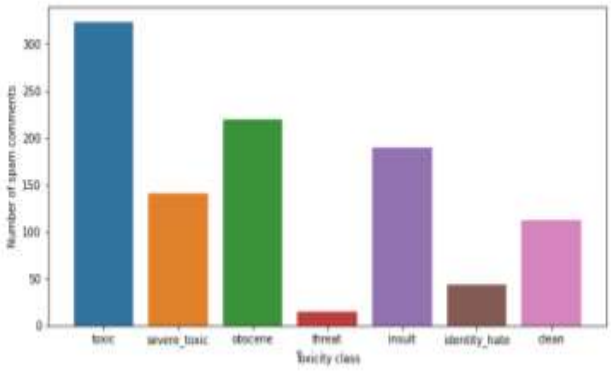


Figure 12: Bar plot - The spam comments (i.e. the comments with less percentage of unique words) increase the chance of toxicity.

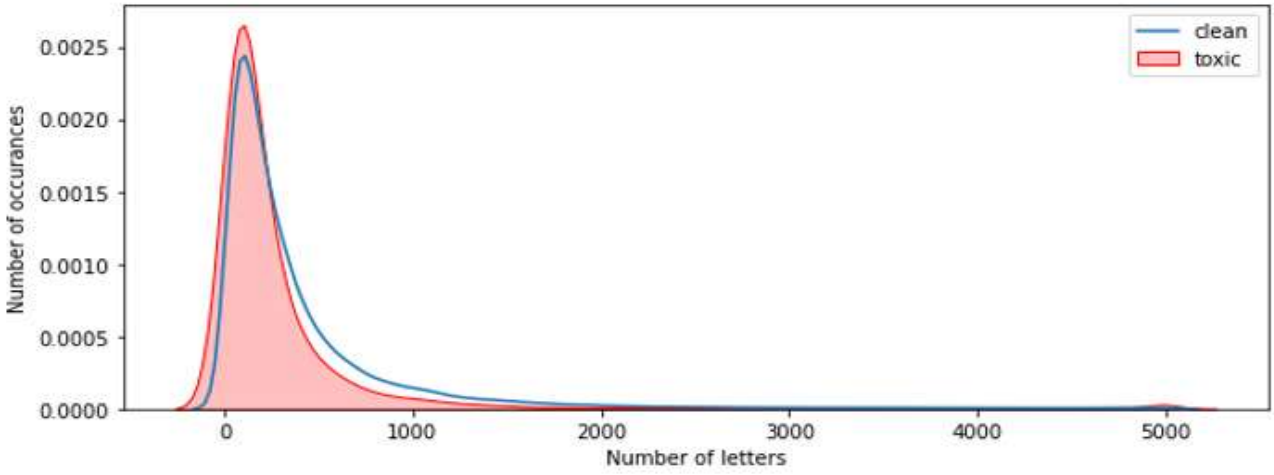


Figure 13: The plot shows that less the no. of letters, more is toxicity.

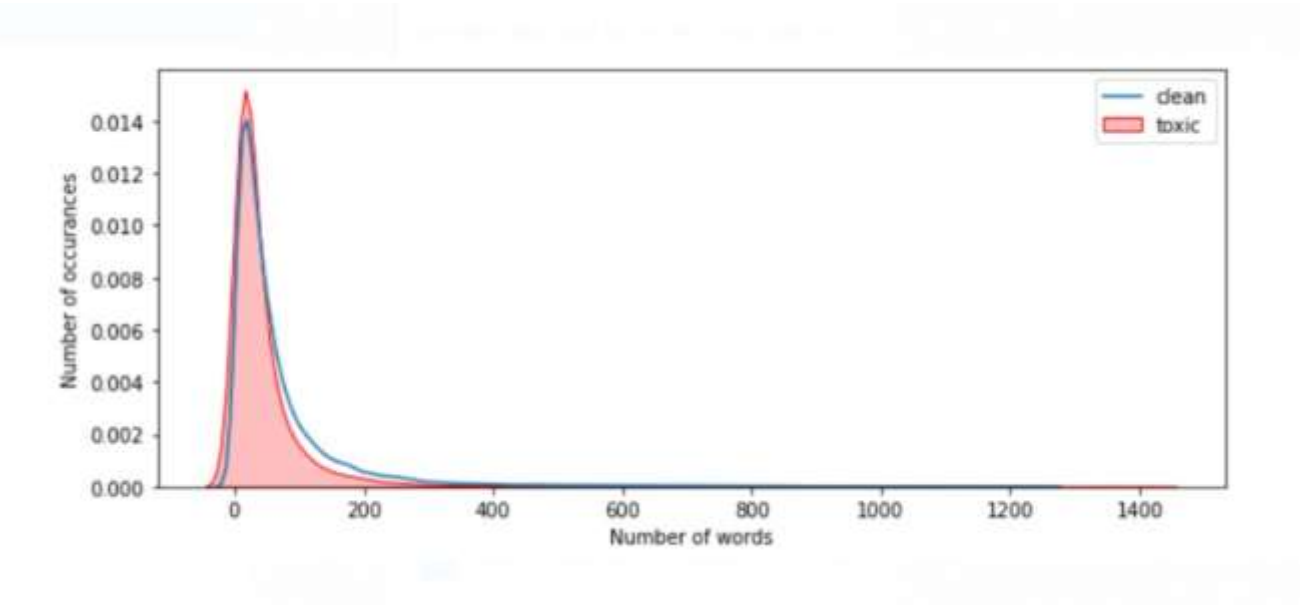


Figure14: Less the no. of words, the more the toxicity. A lot of comments had only single word and yet classified as toxic.

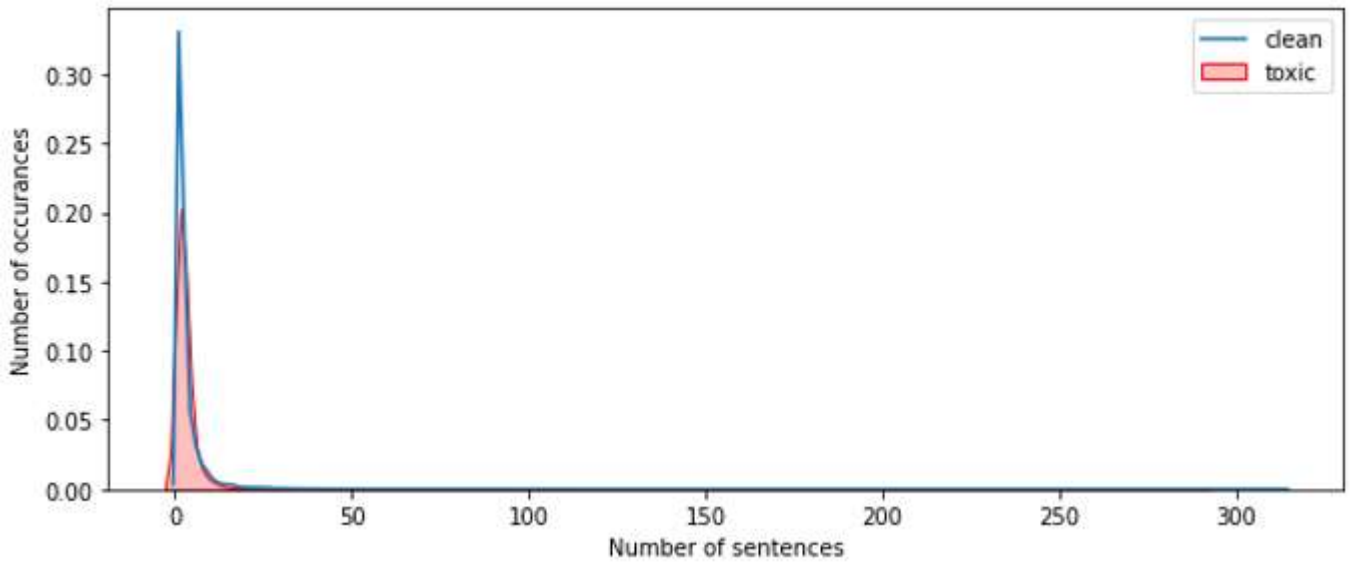


Figure15: Comments do not need to be long in order to be toxic. Less the no. of sentences, more the toxicity.

2.4 Visualization of the Multi-class classification

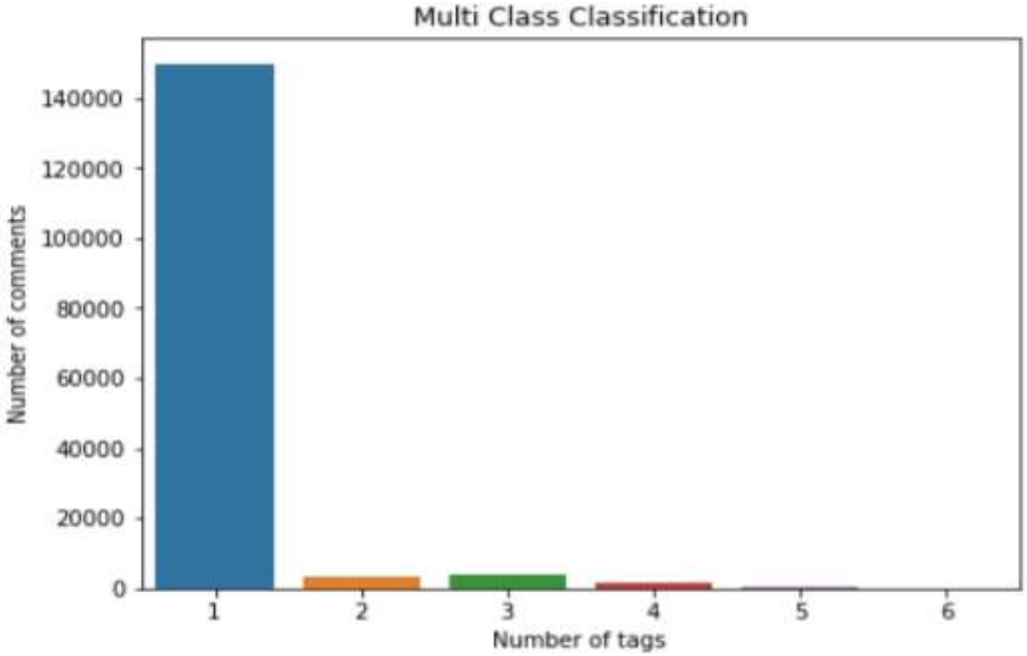


Figure 16: The bar plot shows that a large amount of comments has only a single tag. However, there do exist the comments which have 5 or 6 hate tags.

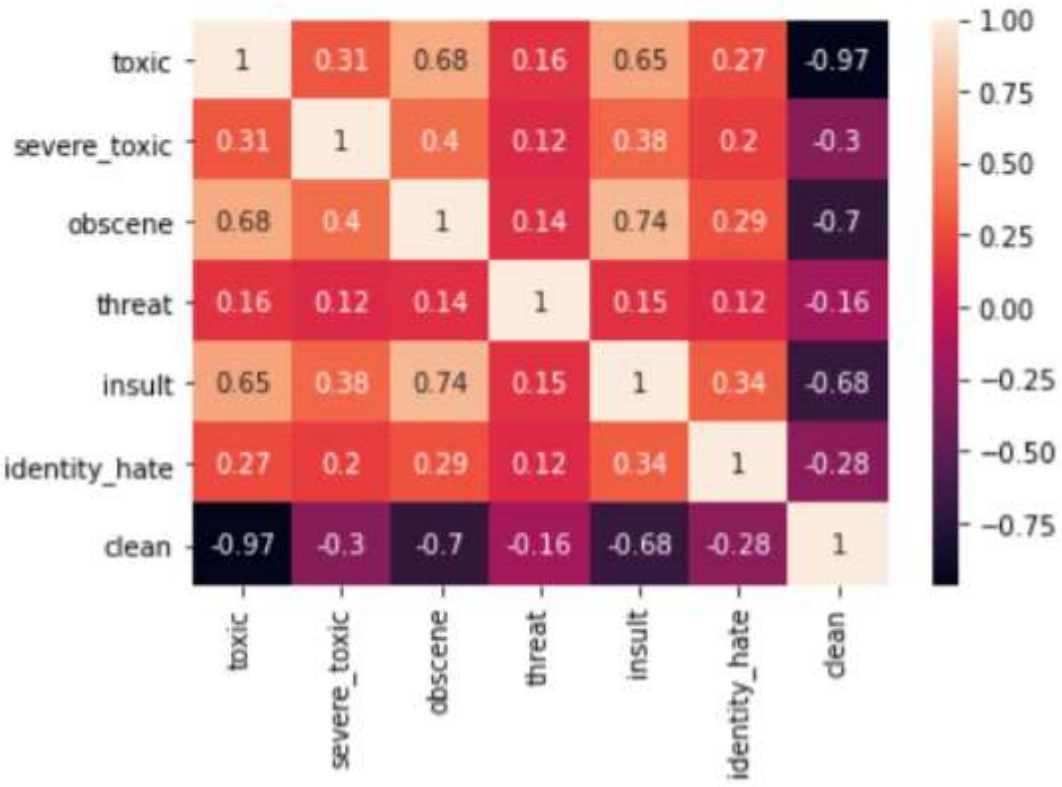


Figure 17: The heat map of the correlation between the tags show a negative correlation between the clean comments and the toxic comments. Also, a strong correlation can be observed between certain tags, such as toxic-obscene, and toxic-insult.

3. Data Cleaning and Preprocessing

During this stage of the Data Mining task, we clean the data, so as to remove any outliers or noise that may be present in the dataset. If any data is missing, then either that data needs to be filled (manually or automatically) or that tuple needs to be ignored completely. Fortunately, we checked for missing values in the dataset and found that none of the values were missing.

Since the dataset contains the comment text from the internet, it may also consist of the URLs/hyperlinks, IP Addresses, user handles, trailing newline marks, etc. which aren't of any use in classifying the text. Therefore, we also removed these from the dataset. We then reduced all the letters to the lowercase after that we decontracted the words. Then we remove words with numbers and lastly we removed the special characters from the reduced sentences.

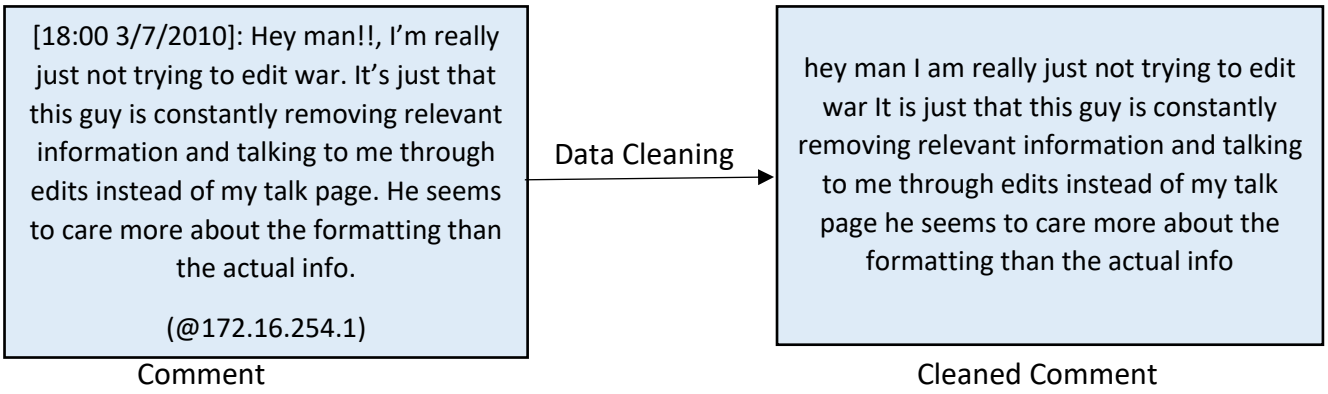


Figure 18: Data Cleaning

Then, the next step involves preprocessing the cleaned data, so that the data in the natural language form can be converted into machine readable form, on which the machine learning algorithms can be applied to get the result. Data preprocessing consists of following steps:

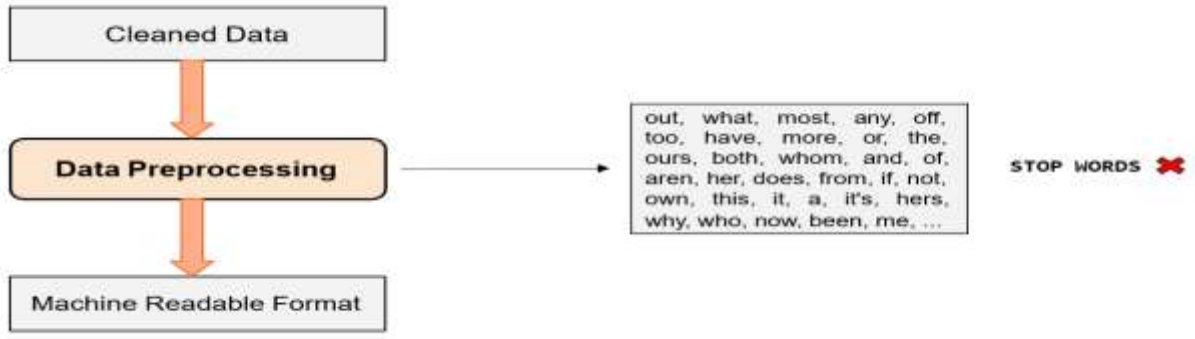


Figure 19: Data Preprocessing Flowchart

3.1 Removing the stop words

The stop words are those words in the comment text, which don't add any physical meaning to the comment, but are generally used as connectors in the natural language in order to make the text grammatically correct. They generally comprise of the auxiliary verbs like is, am, are, was, etc., interrogative pronouns like who, was, were, etc., or any other commonly used words used in the English language. They also have punctuations, that may not be useful. To remove the stop words, we made a list of stop words then we removed those stop words using a for loop.

3.2 Word Encoding Methodology

The preprocessed data is still in natural language form, which is not understandable by the machine. So, it needs to be broken down into tokens or words, and these tokens need to be encoded in the form of integer or floating point numbers, to be used by machine learning algorithms for prediction. This complete process of converting the preprocessed text in natural language form into encoded form is called vectorization or feature extraction.
The following methodologies are typically used for text feature extraction:

1. Tokenizing: Splitting text in the form of tokens and assigning integer or floating-point value to each token.

2. Counting: counting the frequency or the occurrence of token in the document.

3. Normalizing: Penalizing the tokens or reducing the impact of those tokens that occur in most of the documents.

One popular techniques used for vectorization is TF-IDF Vectorizer.

3.2.1 TF-IDF Vectorizer

This vectorizer also normalizes the text, i.e. reduces the impact of the tokens occurring in most of the documents, apart from tokenizing and counting. The value of the TF-IDF term in the vector increases if the frequency of the term is more in the comment text. However, the TF-IDF term decreases if the token appears in most of the comments.TF-IDF stands for Term Frequency Inverse Document Frequency. It consists of two parameters - term frequency (TF) and inverse document frequency (IDF), which decide the final value of the TF-IDF term

• **Term Frequency - TF (t, d):** This is a local parameter which calculates the frequency of a term 't' in a document 'd'. This is similar to the Count Vectorizer method of encoding text.

• **Inverse Document Frequency - IDF (t):** The IDF gives the inverse of the document frequency, i.e. it is a global parameter. Here, Document Frequency DF (t) is the count of documents (i.e. comments) that contain a tern t 't'. Therefore, it calculates the number of documents in which a term appears and it takes the inverse and log of that. In order to avoid a zero-division error, an extra 1 is added to the denominator.

$$IDF(t) = log\frac{1+|D|}{1+df(t)} + 1$$

where df(t) denotes the number of documents containing the term 't' and |D| denotes the total number of documents.
The final TF-IDF term is calculated as follows:

$$TFIDF = TF(t,d) \times IDF(t)$$

In this formula, the first term increases if the word is frequent in the document.
However, if the word appears in most of the documents, then the second term decreases.

Therefore, the TF-IDF term considers both local and global impact of a word in a comment, and it penalizes those words that have a high document frequency and are not meaningful in making predictions. Hence, TF-IDF vectorizer performs better than Count Vectorizer for feature extraction.

3.3 Word Embeddings

Word embeddings are used to represent words in structured format so that machine learning algorithms can be applied on it. Since, the real world data is not structured, word embeddings techniques are a useful tool to transform data into more structured format so that useful information can be extracted.
The Bag of words models are effective methods for extracting features from text, but it fails to capture semantic and context information from unstructured texts. One-hot encoded vectors may lead to a highly sparse structure which causes the model to overfit. To overcome these shortcomings of the above approaches, word embeddings are used.
Word embeddings represent words in the form of vectors in pre-defined dense vector space. These vectors contain meaningful semantic information about the words. The core idea behind the approach is that similar words have similar semantic information. Hence, the similar words will be in proximity within the high dimensional vector space. Thus, we can significantly reduce the vector size in contrast to the one-encoding technique.

3.4 Pretrained Word Embeddings

Pretrained word embeddings are obtained by unsupervised training of a model on a large corpora. As they are trained on a large corpus, they capture the semantic information of the words. Many pretrained-embeddings are made available by different organizations. One of them is:

3.4.1 Word2Vec

Word2Vec is one of the earliest pre trained embedding models. It is classified in two approaches:

1. Continuous Bag of Words (CBOW) model: In this approach, the algorithm tries to predict the word if a context is given. In this way, the word embeddings vectors are generated.

2. Skip-Gram Model: In this approach, the algorithm tries to predict the context or surrounding words in which the word would have been used. It learns by predicting the surrounding words given a current word.

3.5 TF-IDF weighted Word2Vec

In text processing, different features have different influences on the text. For example, the features related to the theme and structure of the text are more important than the general features. Therefore, it is necessary to weight the feature items of the text. In the Word2vec word vector method, the usual method for processing features to obtain the vector representation method of text is weighted average or TF-IDF method, as shown in Formula below.

$$word2vec\_doc(i) = \frac{\sum_{k=0}^{word\_num(i)} word2vec(w_k)}{word\_num(i)} \quad (1)$$

$$word2vec\_TF - IDF_{doc(i)} = \sum_{k=0}^{word_{num(i)}} word2vec(w_k) . TF\_IDF(w_k) \quad (2)$$

Among them, $word2vec\_doc(i)$ represents the vector representation of the text i, $word\_num(i)$ represents the number of words in the text i, $word2vec(w_k)$ represents the number of words corresponding to words w_k, and $TF\_IDF(w_k)$ represents the TF-IDF value of word w_k.

4. Prediction Algorithms

4.1 Problem Transformation

Since this problem is a multi-class classification problem as well as a multi-label classification problem, we need to transform it into either a separate single-class classification problem, or separate single-label classification problem, so that further machine learning algorithms can be applied to the problem. Following two methods are used for this:

4.1.1 Binary Relevance

In this method, we transform the problem into separate single-class classification problems, each of the problems having a single label. We then apply the machine learning algorithm on each problem separately to get the result. As an example, if the problem is to classify a comment with six possible toxic labels, then the problem would be broken into separate six problems, such that each problem has a single label, hence each problem is a single-class classification problem. Thereafter, the results of the six problems can be combined to get all the labels for a comment. Though this is a simple method to solve such multi-label classification problems, it has drawbacks. Since all the problems are treated as independent problems, this method neglects the correlation between the labels. Hence, it gives poor result if any correlation is observed between the labels.

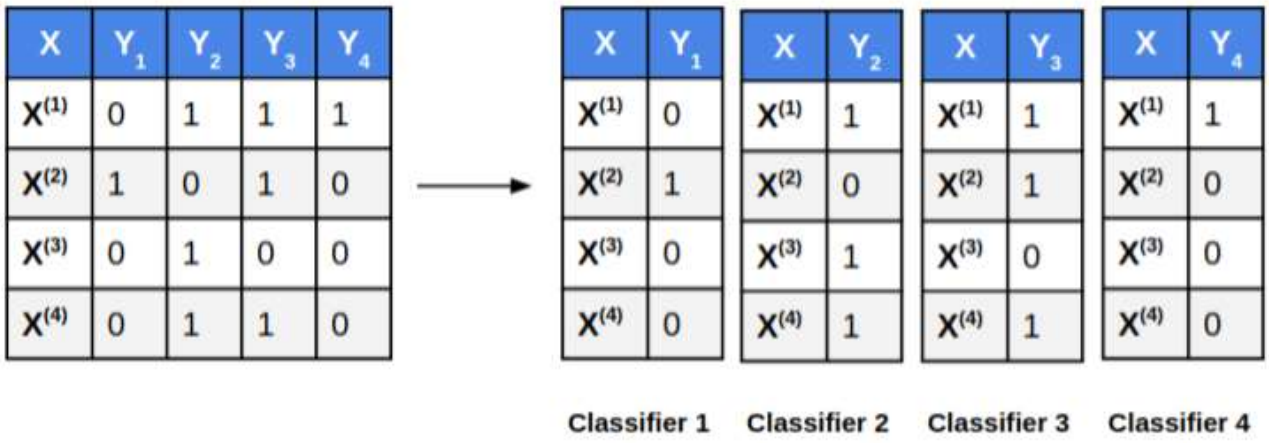


Figure 20: Binary Relevance

4.1.2 Classifier Chains

In this method, we transform the problem into separate single-label classification problems, such that if i^{th} classifier is trained on input variable(s) X, then $(i+1)^{th}$ classifier is trained on input variable X as well as the output produced by i^{th} classifier. Hence, in this method, the first classifier will be the same as the first classifier of Binary Relevance method, but the subsequent classifiers will also include the predictions of the previous classifiers as an input variable. Thus, this technique also considers the correlation between the labels, since for every new classifier, the predictions of the previous classifiers are taken into account, i.e. for a given target variable, it also considers the correlation between previous target variables. Hence, this method is efficient than the Binary Relevance method when the labels are correlated with each other. However, it may perform poorly when all the labels are independent with each other. For the task of hate speech detection, we applied the problem transformation methods using Binary Relevance and Classifier Chains, on Logistic Regression and Support Vector Machines. As there was a strong correlation between the tags, such

as toxic-obscene, toxic-insult, etc, hence the Classifier Chains technique performed better than the Binary Relevance technique.

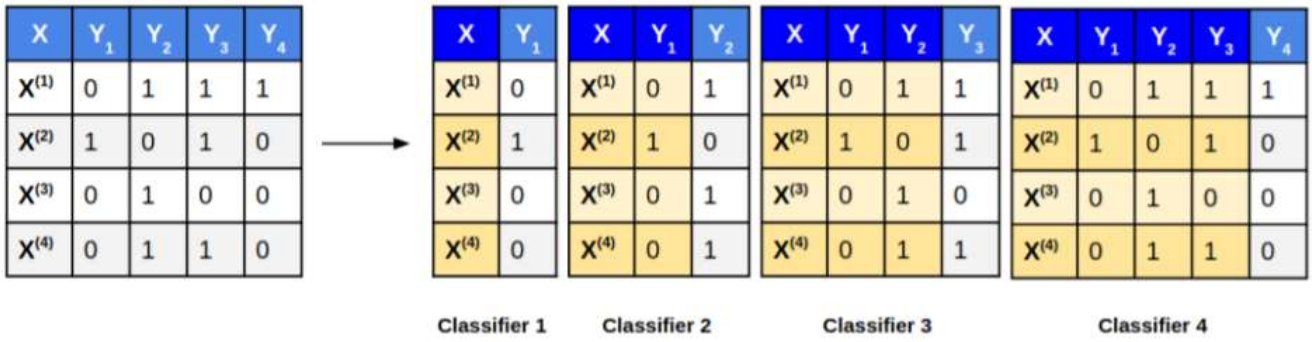


Figure 21: Classifier Chains

4.2 Logistic Regression

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. It is one of the most important analytic tools in the social and natural sciences. In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also has a very close relationship with neural networks. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. The binary logistic model classifies specimen into two classes, whereas the multinomial logistic model extends this to an arbitrary number of classes without ordering them. The mathematics of logistic regression rely on the concept of the "odds" of the event, which is a probability of an event occurring divided by the probability of an event not occurring. Just as in linear regression, logistic regression has weights associated with dimensions of input data. In contrary to linear regression, the relationship between the weights and the output of the model (the "odds") is exponential, not linear.

4.3 Support Vector Machines (SVM)

Support Vector Machines use the concept of support vectors and hyperplanes for classification tasks. They fall under the domain of supervised machine learning algorithms and can be used for both classification and regression tasks, particularly for the former one. They can be used for both linear and non-linear data. The main objective of the support vector machine is to construct an optimal hyperplane in an N-dimensional space i.e. a boundary separating the data points, such that the margins between the support vectors is maximized.

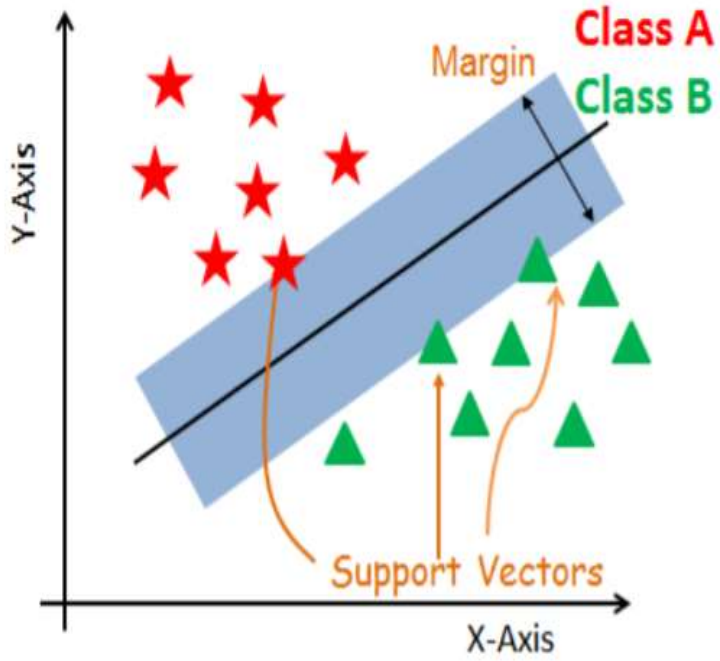


Figure 22: Support Vector Machine

Here, the support vectors denote the data points which are closest to the hyperplane and are useful for training tasks, and the margin denotes the distance between the two parallel lines passing through the closest support vectors on either side of the hyperplane. For non-linearly separable data, it transforms the original data into higher dimensions for the classification task. This is called the Kernel Trick. As an example, for the data points shown in the figure, the data points plotted in the left figure in x-y plane are not linearly separable, therefore the same data points are plotted in higher dimensional plane, say x-z plane in the right figure, such that $z = x^2 + y^2$. Thus, these data points can be linearly separable using support vector machines.

Support Vector Machines are very efficient for classifying higher dimensional data. They are very effective when the dimension of data is greater than the number of samples. This is because the complexity of the SVM classifier does not depend on the dimensionality of data, rather on the number of support vectors. Since only the support vectors are used for training, they are memory-efficient too. However, they perform poorly in the case of noisy data, e.g. when there is an overlap between the classes. Also, they are not much efficient for large datasets, as the time required to train the model is higher when the dataset is large.

	Unnamed: 0	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	vector
0	0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	[0.00593334, 0.07003178, -0.2012667, 0.3775345...
1	1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	[-0.2508106, 0.01673899, 0.31538136, -0.121969...
2	2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	[0.41935966, -0.12742983, -0.48621123, 0.24537...
3	3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	[0.32448252, -0.40889367, -0.00564199, 0.13656...
4	4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	[-0.04981409, 0.68221519, -0.62567751, 0.10110...

Figure 23: Vector

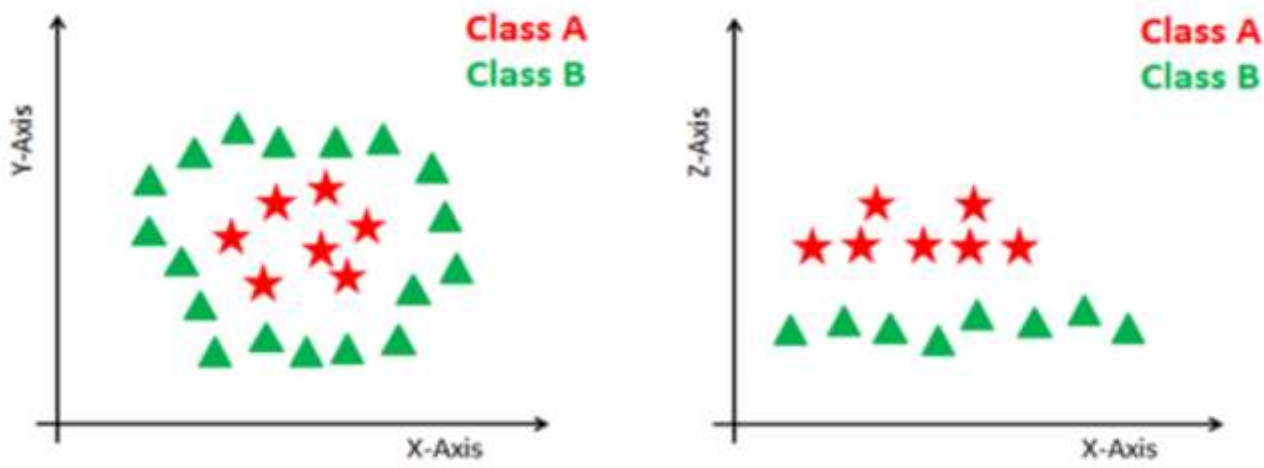


Figure 24: SVM on non-linearly separable data

4.4 Naive Bayes

Naïve Bayes classifier is a probabilistic classifier based on Bayes' theorem, which assumes that each feature makes an independent and equal contribution to the target class. NB classifier assumes that each feature is independent and does not interact with each other, such that each feature independently and equally contributes to the probability of a sample to belong to a specific class. NB classifier is simple to implement and computationally fast and performs well on large datasets having high dimensionality. NB classifier is conducive for real-time applications and is not sensitive to noise. NB classifier processes the training dataset to calculate the class probabilities $P(y_i)$ and the conditional probabilities, which define the frequency of each feature value for a given class value divided by the frequency of instances with that class value. NB classifier best performs when correlated features are removed because correlated features will be voted twice in the model leading to the overemphasis of the importance of the correlated features. Naïve Bayes algorithms are often used in sentiment analysis, spam filtering, recommendation systems, etc. They are quick and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent.

In our model, after performing Naïve Bayes algorithm we achieved:

- Mean training accuracy of approximately 93.25%
- Mean validation accuracy of approximately 93.32%
- Mean validation F1 score of 0.3728

5. Results and Analysis

5.1 Performance Measures

The problem involves highly unbalanced dataset. So accuracy is not a well suited performance measure. With only 10% of the training data belonging to the positive class (hate tags), it is trivial to achieve 90% accuracy by a naive model which simply labels every input as clean. Indeed, we verified this by fitting a very simple Naive Bayes model which achieved a validation accuracy of whopping 93% which looks good on paper only as long as one doesn't analyse the model predictions manually or by some other performance measures. As quite expected, the recall was merely 65% which means the model is simply unable to recognise 35% of the hate comments. The precision scores were still poorer just 35% indicating even among the comments predicted as hate tags, 65% are misclassified. Needless to say, such a model is not of any use which highlights the essence of a good evaluation metric. Precision-Recall or F1 score seem like the next obvious choice however they have their own share of limitations including selection of threshold value and relative importance to be given to precision vs recall. Hence we finally settled on the ROC curve and AUC score which give a very accurate picture of the performance of a discriminative model.

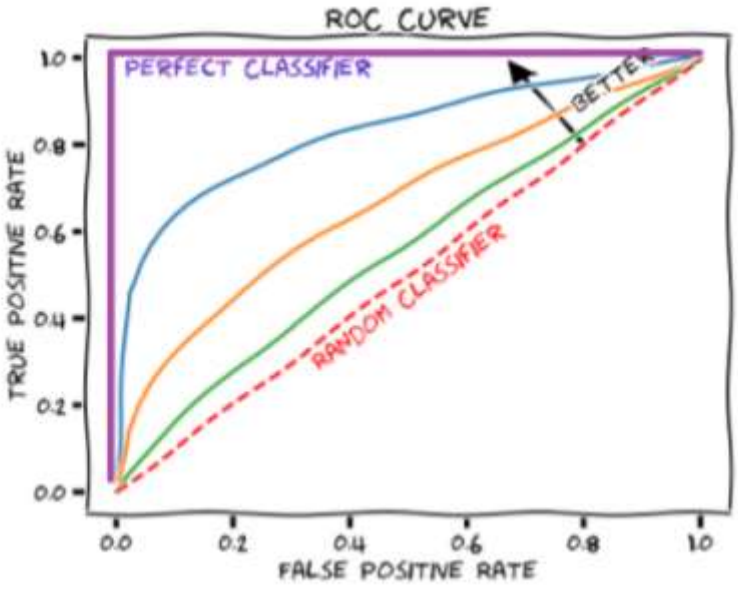


Figure 25: ROC

5.1.1 Receiver Operating Characteristic

A Receiver Operating Characteristic is a curve which plots True Positive Rate vs True Negative Rate. These two factors are an indicator of the model performance.

1. True Positive Rate (TPR): $TRP = \frac{TP}{TP+FN}$

2. False Positive Rate (FPR): $FPR = \frac{FP}{FP+TN}$

The aim of any discriminative model is to maximize the TPR while at the same time keeping the FPR as low as possible. The adjoining figure shows a perfect classifier and a random classifier.

5.1.2 AUC

AUC denotes the complete Area Under the ROC curve for the given domain. AUC values can range from 0 to 1. The idea of AUC stems from the observation that a better model will have more area under the ROC curve with a perfect model having AUC=1 and a model which always predicts incorrectly having AUC score=0. In this sense AUC can be understood as the average of performance measures of the classifier across all thresholds. Another interesting interpretation is that it expresses the probability that the model gives a higher positive score to a hate comment in our case as compared to a clean comment for any threshold value chosen.

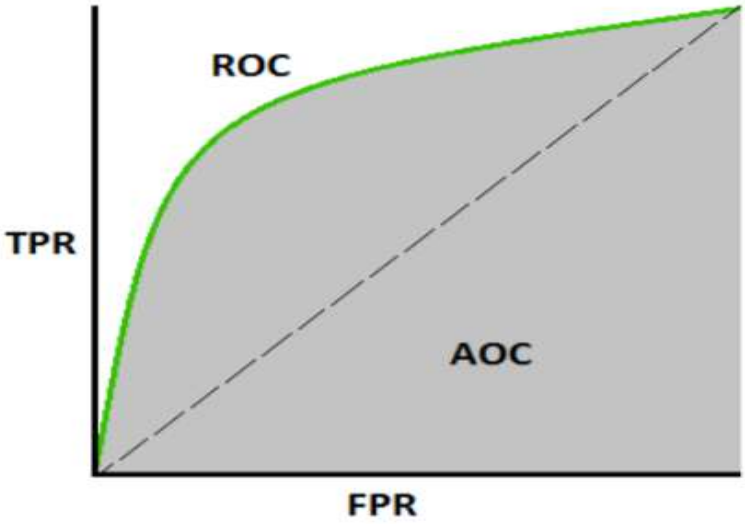


Figure 26: AOC ROC Curve

Advantages

1. Scale Invariant: AUC is independent of the scale and number of the predictions. It is based on fractions which always lie between 0 and 1. So it is a versatile model which can be used to compare the performance of several models alike.

2. Threshold Invariant: Unlike F-measure, it doesn't depend on the threshold set for classifying an example as positive. Despite these, AUC does have some shortcomings which can be alleviated by combining it with some other metrics. We however limit ourselves to AUC for the purpose of this project. The mean AUC-ROC scores obtained from different models are mentioned in the table:

Model	Mean AOC ROC Score
Support Vector Machines (Binary Relevance)	0.6375
Support Vector Machines (Classifier Chains)	0.6529
Logistic Regression (Binary Relevance)	0.6446
Logistic Regression (Classifier Chains)	0.6446

Figure 27: Mean AUC ROC scores obtained from different models

6. Conclusion

We compared the performance of the model based on the mean AUC ROC scores. We observed that the classical models like Support Vector Machines and Logistic Regression failed to achieve high AUC ROC scores. We also found out that the SVM method performed slightly better than logistic regression in this task. Further Improvements can be achieved by experimenting with complicated neural networks.

7. References

[1] https://stackoverflow.com/

[2] https://www.wikipedia.org/

[3] https://iopscience.iop.org/article/10.1088/1742-6596/1486/7/072032/pdf