

**MALIGNANT COMMENT CLASSIFIER**

Submitted by:

NIKHIL SINGH RANA

**ACKNOWLEDGMENT**

The satiation that accompanies the successful completion of the project would be incomplete without the mention of the people who made it possible

I would like to take the opportunity to thank and express my deep sense of gratitude to my data trained academy mentors for providing their valuable guidance at all stages of the study of my data scientist course, their advice and constructive suggestion through which I have gained this much skills that I can complete this project.

I have taken the help of my previous projects which I had done in my training phase with data trained academy and also refered google for some line of codes .

**INTRODUCTION**

* Business Problem Framing

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Humans have built extensive models of expressing their thoughts via numerous means. The internet has not only become a credible method for expressing one's thoughts, but is also rapidly becoming the single largest means of doing so. In this context, the increasing levels of threat and abuse from certain sections of the online community renders a fair degree of inadequacy to facilitate proper conversations. Quite naturally, one area that requires attention is the identification of negative online behavior and consequentially, restricting their scope. This can be done by developing a classification module for the various types of toxic comments that we may be interested in finding. More recently, Deep Learning methods have been used as a potential method for abusive and toxic comment detection. The following work explores the usage of a very basic Logistic Regression classifier and then moves on to explore Deep Learning based approaches based primarily on Sequential Models

* Conceptual Background of the Domain Problem

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but “u are an idiot” is clearly offensive.

* Review of Literature

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 Malignant , Highly Malignant, rude, Threat, Abuse, and Loathe. 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, 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.

* Motivation for the Problem Undertaken

Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

Detection and classification of abusive language online is a difficult NLP challenge as there are endless number of online comments being published every second, all with different contexts and with each consisting of words from different known and unknown set of words. The goal of this work was to explore and implement some of the baseline approaches available online as well as to try and implement advanced concepts that would help model the data in a better fashion. Through a comprehensive Exploratory Data Analysis and visualization setting, we can also aim to better understand the underlying patterns and behaviors that exist amongst these comments

**Analytical Problem Framing**

* Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

**Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.

**Highly Malignant:** It denotes comments that are highly malignant and hurtful.

**Rude:** It denotes comments that are very rude and offensive.

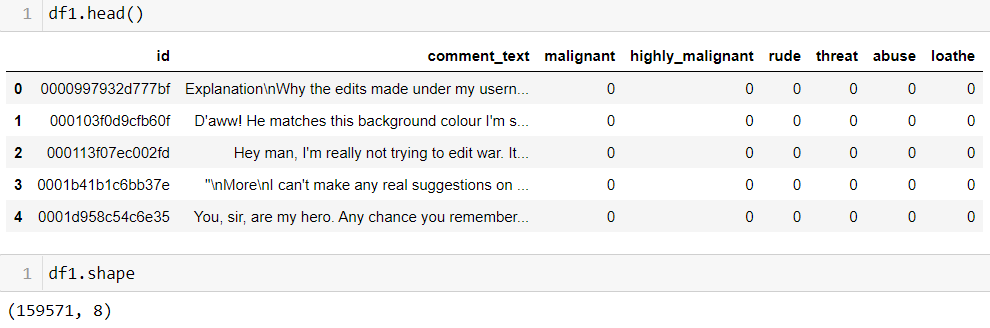
**Threat:** It contains indication of the comments that are giving any threat to someone.

**Abuse:** It is for comments that are abusive in nature.

**Loathe:** It describes the comments which are hateful and loathing in nature.

**ID:** It includes unique Ids associated with each comment text given.

**Comment text:** This column contains the comments extracted from various social media platforms.



* Data Preprocessing Done

Step 1: Checking for missing values.

First and foremost, after importing the training and test data into the pandas dataframe, I decided to check for missing values in the downloaded data. Using the “isnull” function on both the training and test data, I discovered that there were no missing records and therefore, I moved on to the next step of my project.

Step 2: Text Normalization.

As I was now certain that there are no missing records in my data, I decided to start with data pre-processing. Firstly, I decided to normalize the text data since comments from online forums usually contain inconsistent language, use of special characters in place of letters (e.g. @rgument), as well as the use of numbers to represent letters (e.g. n0t). To tackle such inconsistencies in data, I decided to use **Regex.** The text normalization steps that I performed are listed below:-

Removing Characters in between Text.

Removing Repeated Characters.

Converting data to lower-case.

Removing Punctuation.

Removing unnecessary white spaces in between words.

Removing “\n”.

Removing Non-English characters.

Removing Hashtags

Removing Usernames Etc.

Step 3: Stopwords Removal.

Stopwords Removal, as we all know, is one of the most critical steps in text pre-processing for use-cases that involve text classification. Removing stopwords ensures that more focus is on those words that define the meaning of the text.

To remove stopwords from my data, I took the help of the “spacy” library. Spacy has a list of common stopwords, “STOP\_WORDS” that can be used to remove stopwords from any textual data.

Step 4: Text Cleaning

Next step is to remove unnecessary elements from text.

It is important to note that this "unnecessary text" elimination process could be more successful when followed in a particular sequence.

For instance, elimination of special characters before removal of hashtags, usernames defeats our purpose as the text following a '#' could be considered a token (ex: #kaggledays -> kaggledays).

Here is the sequence we will implement:

HTML codes (if present)

URLs/ email addresses

Hashtags/Usernames

Emojis

Stopwords

Expanding Abbreviations

Punctuations

Special characters/ Numbers





Step 5: Lemmatization.

Since the data is now clean and consistent, it is the right time to perform **Lemmatization**. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. For example, we do not want the Machine Learning algorithm to treat studying, studies, and study as three separate words because, in truth, they are not. Lemmatization helps reduce the words “studying” and “studies” to their root form, i.e. study. To implement Lemmatization, I imported “WordNetLemmatizer” from the “nltk” library, created a function “text\_transformation” to perform Lemmatization, and applied it to the clean data that I procured from Step 4.

Step 6 : Vectorization

Bag of words techniques do not take into consideration the order of words in a given text. These techniques are primarily concerned with number of occurences of words in the text.

There are three ways in which we could vectorize text:

Count Vectorizer

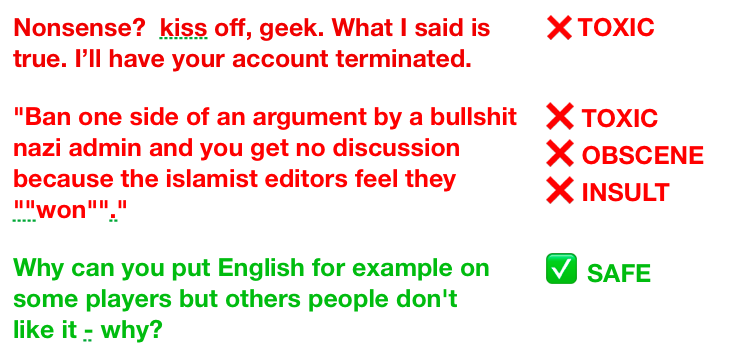
Tfidf Vectorizer

Hashing Vectorizer

I have use Tfidf Vectorizer

* Data Inputs- Logic- Output Relationships

The background for the problem originates from the multitude of online forums, where-in people participate actively and make comments. As the comments some times may be abusive, insulting or even hate-based, it becomes the responsibility of the hosting organizations to ensure that these conversations are not of negative type. The task was thus to build a model which could make prediction to classify the comments into various categories. Consider the following example:



Example showing 2 toxic and 1 non-toxic comment.

**The exact problem statement was thus as below:**

Given a group of sentences or paragraphs, used as a comment by a user in an online platform, classify it to belong to one or more of the following categories — **Malignant, Highly Malignant, Rude, threat, Abuse or Loathe**  with either approximate probabilities or discrete values (0/1).

* State the set of assumptions (if any) related to the problem under consideration

The goal is to create a classifier model that can predict if input text is inappropriate (toxic). 1. Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments. 2. Create a baseline score with a simple logistic regression classifier. 3. Explore the effectiveness of multiple machine learning approaches and select the best for this problem. 4. Select the best model and tune the parameters to maximize performance. 5. Build a the final model with the best performing algorithm and parameters and test it on a holdout subset of the data

* Hardware and Software Requirements and Tools Used

I have used Jupyter notebook for this project, and following are the tools and libraries I have used in this particular project :



**Model/s Development and Evaluation**

* Identification of possible problem-solving approaches (methods)

Following are the approaches i have followed for solving this problem :

* Defined Various functions for plotting graphs, for example:
* **category\_percentage** ( in this i have defined what are the weight percentages of each categories)
* **corr\_between\_labels** ( in this i have defined the correlation between each categories )
* **avg\_word\_len\_plot** ( in this i have defined what are the average word length in each category )
* **generate\_wordclouds** ( in this i have defined visualization of toxic comment and clean comments in the word clouds )
* For text transformation I have used **Word Net Lemmatization**
* For toxanization I have used **Tfidf Vectorizer**.
* Testing of Identified Approaches (Algorithms)

Algorithms used for the training and testing are as follows:

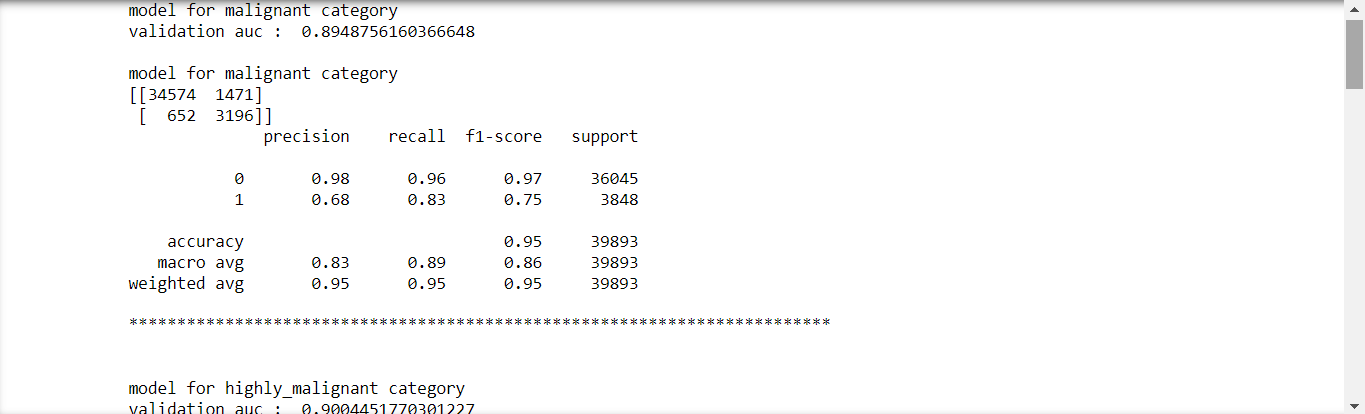
**Train test split** : The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

**Tfidf Vectorizer**: TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. Let’s take sample example and explore two different spicy sparse matrix before go into deep explanation .

**Logistic Regression**: Logistic regression is another technique borrowed by machine learning from the field of statistics.

It is the go-to method for binary classification problems (problems with two class values). In this post you will discover the logistic regression algorithm for machine learning.



* Run and Evaluate selected models

As a natural language processing problem, is a classification task that involves high dimensionality data. I will vectorize the data and test multiple classification algorithms.

I will vectorize the text data using the term frequency – inverse document frequency (tf-idf) statistic. This technique takes into account not only the frequency of words or character n-grams in the text, it also takes into account the relevancy of those tokens across the dataset as a whole. The inverse document frequency reduces the weight of common tokens while boosting the weight of more unique tokens. I will establish a benchmark for performance with the top 10,000 words, and the number of tokens and the mix of words and character n-grams will be a parameter to tune for higher performance later on.

I will also create a number of engineered features containing various attributes of the comment text, such as average word length, capitalization, and number of exclamation points. I will run the benchmark test without these features and experiment with them to optimize the solution.

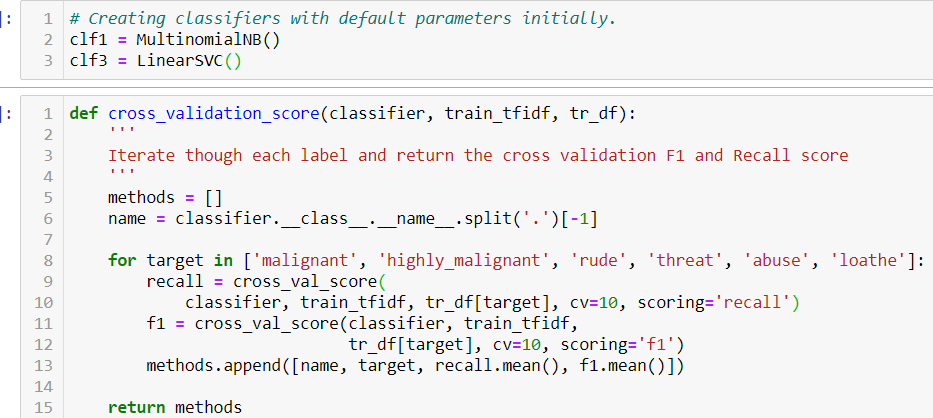
With the benchmark vectorization and features, I will experiment with multiple algorithms with default parameters to determine the most effective approach to the problem. The models I will use are:

Logistic Regresssion (Benchmark)

Multinomial Naive Bayes

Linear SVC





* Key Metrics for success in solving problem under consideration

**Cleaning:**

The dataset is relatively clean. There is a minor data leak, IP addresses appended to some comments. For various reasons, mainly that this data may slightly compromise the model’s ability to generalize to new data, I’ve used a regular expression to strip all IP addresses from comments.

**Feature Engineering :**

During the exploratory data analysis, I found that many attributes of comments outside of the words themselves may be useful in predicting whether they are toxic. The features I added to the dataset are:

Comment length in characters

Percent of letters in a comment that are capitalized

Average length of words in a comment

Number of exclamation marks in a comment

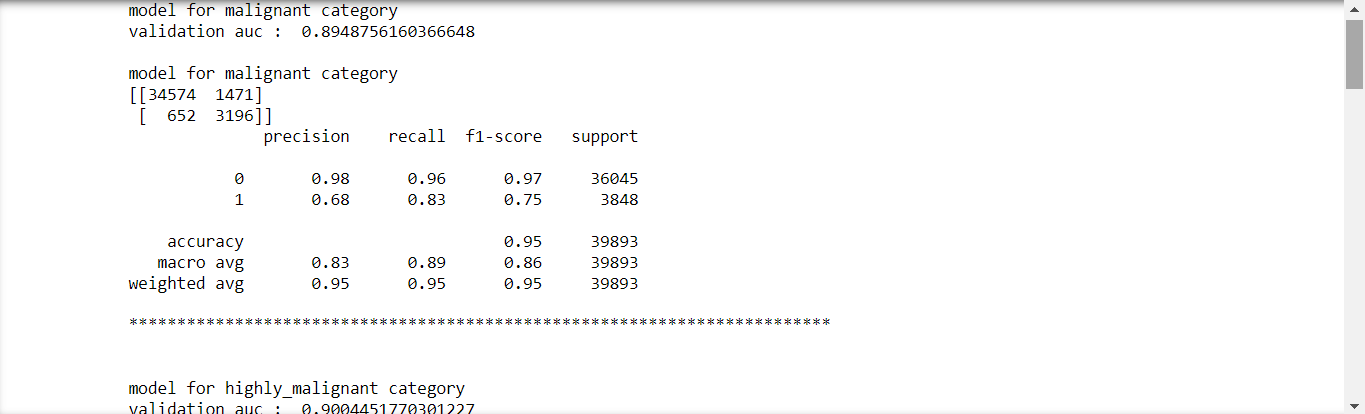
Number of question marks in a comment

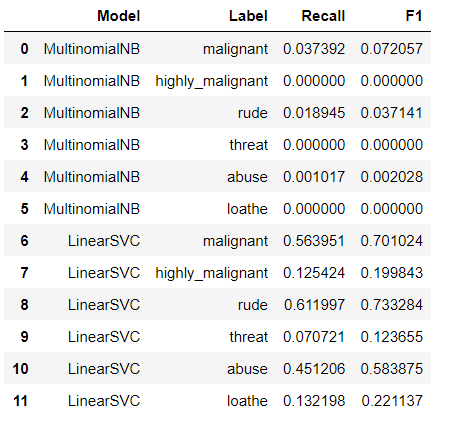
**Vectorization :**

As discussed previously, I am using a term frequency – inverse document frequency (tf-idf) statistic to vectorize text. The number of features and presence of character n-grams is a parameter to tune for model optimization.

**Implementation :**

I have implemented Logistic Regression model, multinomial NB, Linear SVC

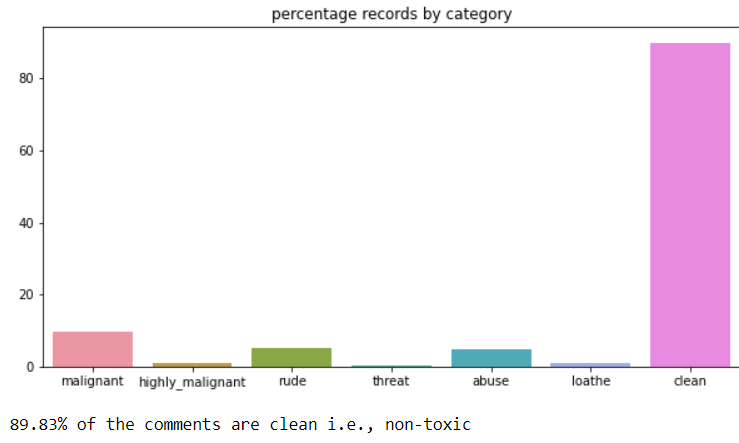
****

****

* Visualizations

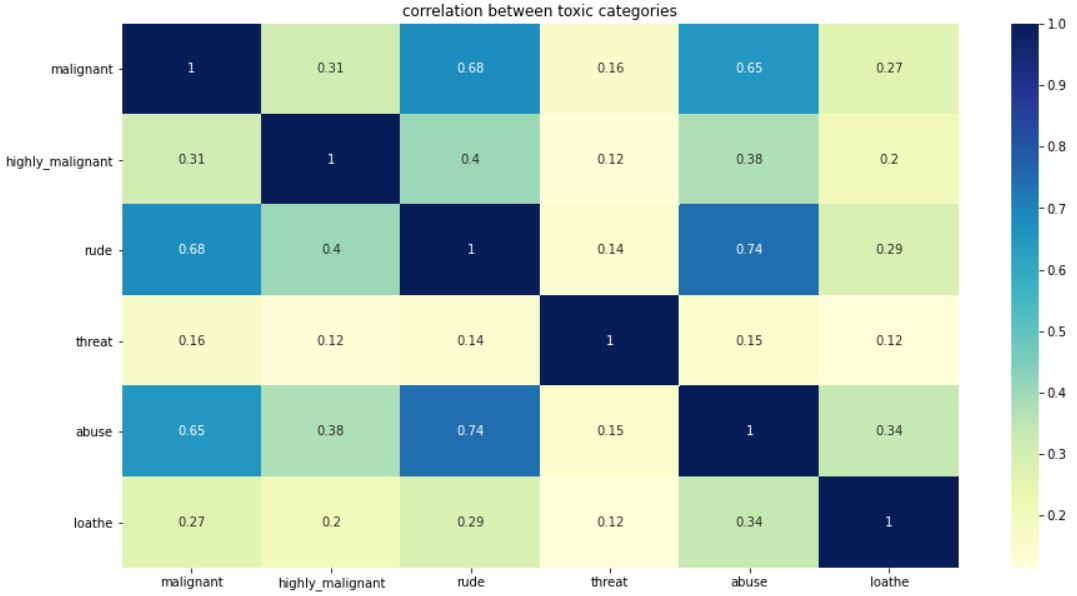
I have defined different functions for different graphs

* Percentage of data vs Category:



This is clearly a case with high class imbalance.

* Correlation And Interrelation:
* Correlation between labels:
* Rude and Abuse, Malignant and Rude exhibit a strong correlation



* Average word length in clean and toxic comments:
* The average word length in both clean and toxic comments is unreasobaly high;
* we need to clean the data and inspect further, if the issue persists





* Wordclouds - unprocessed data:
* We observe a heavy use of upper-cased, racist text (offensive nouns) in toxic comments
* Clean comments on the other had fewer nouns and no observable upper case text





* Interpretation of the Results

Give a summary of what results were interpreted from the visualizations, preprocessing and modelling.

**CONCLUSION**

* Learning Outcomes of the Study in respect of Data Science

Reflection

* The process for this project was as follows:
* Analyze the problem and propose a useful solution.
* Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments.
* Develop an objective that fits a practical use case and addresses the major class imbalance.
* Create a baseline score with a simple logistic regression classifier.
* Explore the effectiveness of multiple machine learning algorithms.
* Select the best model based on a balance of performance and efficiency.
* Refine the preprocessing strategies to optimize model performance.
* Tune model parameters to maximize performance.
* Build a the final model with the best performing algorithm and parameters and test it on a holdout subset of the data.

The most difficult yet most interesting aspect of the project was understanding the relationship between the size of input data and the performance of various machine learning algorithms.

* Limitations of this work and Scope for Future Work

Improvement

I believe that there are a number of ways that the solution could be improved.

Recurrent neural networks offer extremely high performance on natural language processing problems, and if the architecture for the inferrence step were implemented efficiently the computational overhead would be minimal.

Another great strategy could be using multiple models, a sort of divide an conquer method where the problem is divided into multiple smaller, contextual problems. While the solution laid out here generalizes to the entire dataset, no one solution will be able to generalize perfectly to the diverse variety of inputs you’ll get from Internet users. By training models on different situations, like a model that’s only been trained on short or long comments, to only detect whether a comment is toxic when profanity is present, etc, and storing them in memory, you could use a simple decision tree to feel comments into the model that would be most effective. A few that I can think of are:

Short comments

Long comments

We can also make class imbalance balanced.