# Toxicity Prediction and EDA

# Aim

IUsing a dataset of 1.8million + tweets for exploring various trends in the data, and training Machine Learning and Deep Learning models to do our prediction task. Here I have treated the problem as a regression problem so that people can get the toxicity rating on a scale of 0 to 1 and not an absolute verdict on whether a message is toxic or not.

The model could also have been trained as a classification model.

# Introduction

Increasing toxicity in social media messages and live streaming platforms requires automated methods that can handle these messages and report to the channel/id owners so that they can take the required action and the community can be kept clean.

Current methods mostly include manual moderation by humans which is not efficient as well as always accurate.

Currently the models are being trained on an English language dataset, and is planned to be extended to Hindi Language as well.

# Exploratory Data Analysis

**Finding dimensions, and primary view of the dataset.**

```
train_df.shape
```

```
(1804874, 45)
```

Thus, we can see that the dataset has 45 columns (indicating 45 features) and 1.8 million+ tuples , indicating 1.8m+ tweets.

This is the primary view of the dataset:

```
#describe the dataset
train_df.head(5)
```

| | id | target | comment_text | severe_toxicity | obscene | identity_attack | insult | threat | asian | atheist | ... | article_id | rating | fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 59848 | 0.000000 | This is so cool. It's like, 'would you want yo... | 0.000000 | 0.0 | 0.000000 | 0.00000 | 0.0 | NaN | NaN | ... | 2006 | rejected | |
| **1** | 59849 | 0.000000 | Thank you!! This would make my life a lot less... | 0.000000 | 0.0 | 0.000000 | 0.00000 | 0.0 | NaN | NaN | ... | 2006 | rejected | |
| **2** | 59852 | 0.000000 | This is such an urgent design problem; kudos t... | 0.000000 | 0.0 | 0.000000 | 0.00000 | 0.0 | NaN | NaN | ... | 2006 | rejected | |
| **3** | 59855 | 0.000000 | Is this something I'll be able to install on m... | 0.000000 | 0.0 | 0.000000 | 0.00000 | 0.0 | NaN | NaN | ... | 2006 | rejected | |
| **4** | 59856 | 0.893617 | haha you guys are a bunch of losers. | 0.021277 | 0.0 | 0.021277 | 0.87234 | 0.0 | 0.0 | 0.0 | ... | 2006 | rejected | |

5 rows × 45 columns

## Some of the columns in the dataset are: (Most of them are numbers).

```
0   id                                  int64
1   target                              float64
2   comment_text                        object
3   severe_toxicity                     float64
4   obscene                             float64
5   identity_attack                     float64
6   insult                              float64
7   threat                              float64
8   asian                               float64
9   atheist                             float64
10  bisexual                            float64
11  black                               float64
12  buddhist                            float64
13  christian                           float64
14  female                              float64
15  heterosexual                        float64
16  hindu                               float64
17  homosexual_gay_or_lesbian           float64
18  intellectual_or_learning_disability float64
19  jewish                              float64
20  latino                              float64
21  male                                float64
22  muslim                              float64
23  other_disability                    float64
24  other_gender                        float64
25  other_race_or_ethnicity             float64
26  other_religion                      float64
27  other_sexual_orientation            float64
28  physical_disability                 float64
29  psychiatric_or_mental_illness       float64
30  transgender                         float64
31  white                               float64
32  created_date                        object
33  publication_id                      int64
34  parent_id                           float64
35  article_id                          int64
36  rating                              object
37  funny                               int64
38  wow                                 int64
```

**Number of Parent tweets and number of tweet replies distribution in the dataset: ( 1 is for parent, 0 is for reply).**

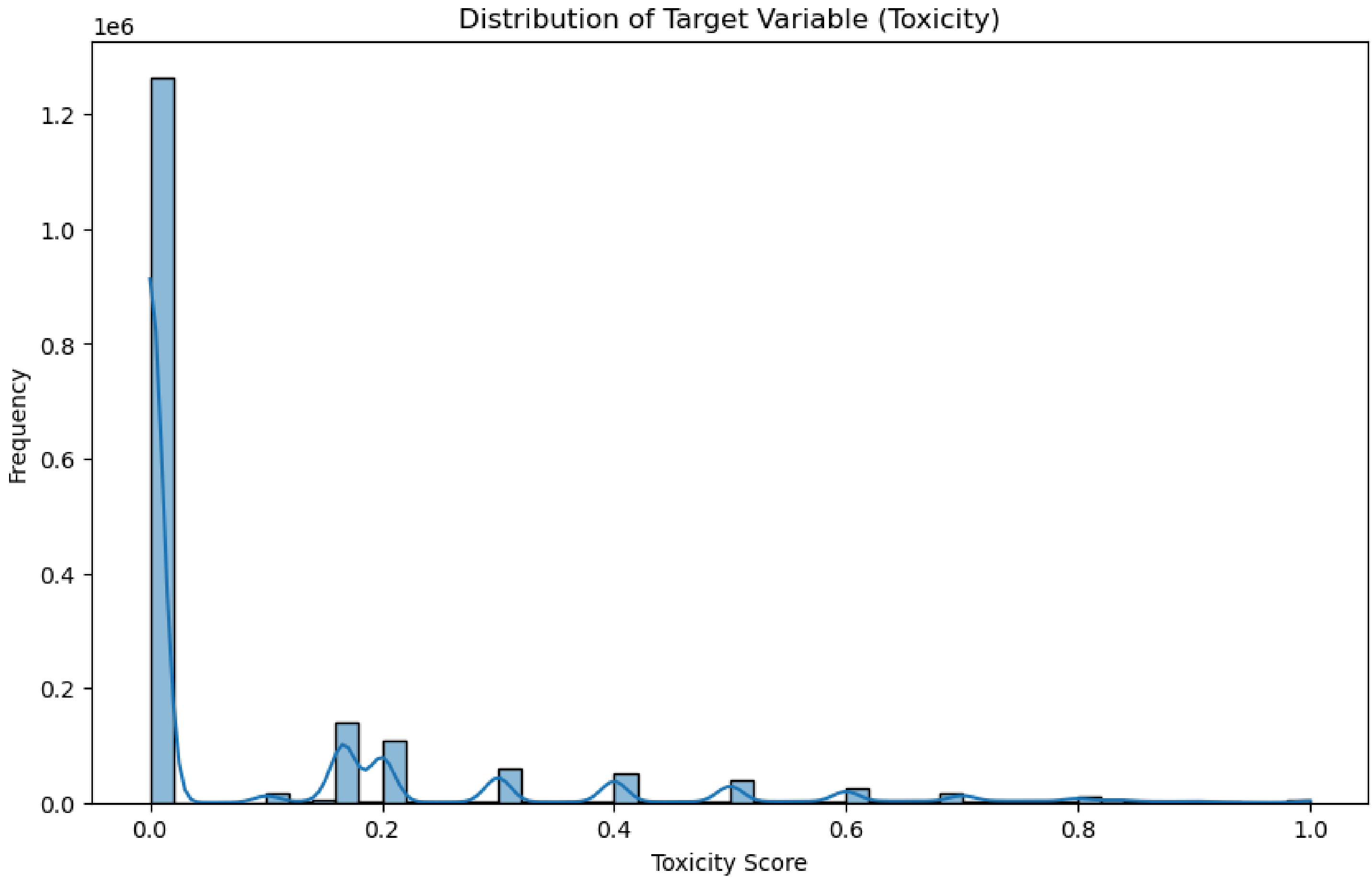
```
is_reply
1    1026225
0     778646
Name: count, dtype: int64
```

## Analysing the target column:

**Number of unique values of the target column:**

```
target
0.000000    1264761
0.166667     138501
0.200000     107492
0.300000      59098
0.400000      50013
              ...
0.026684          1
0.924561          1
0.007458          1
0.145161          1
0.870088          1
```

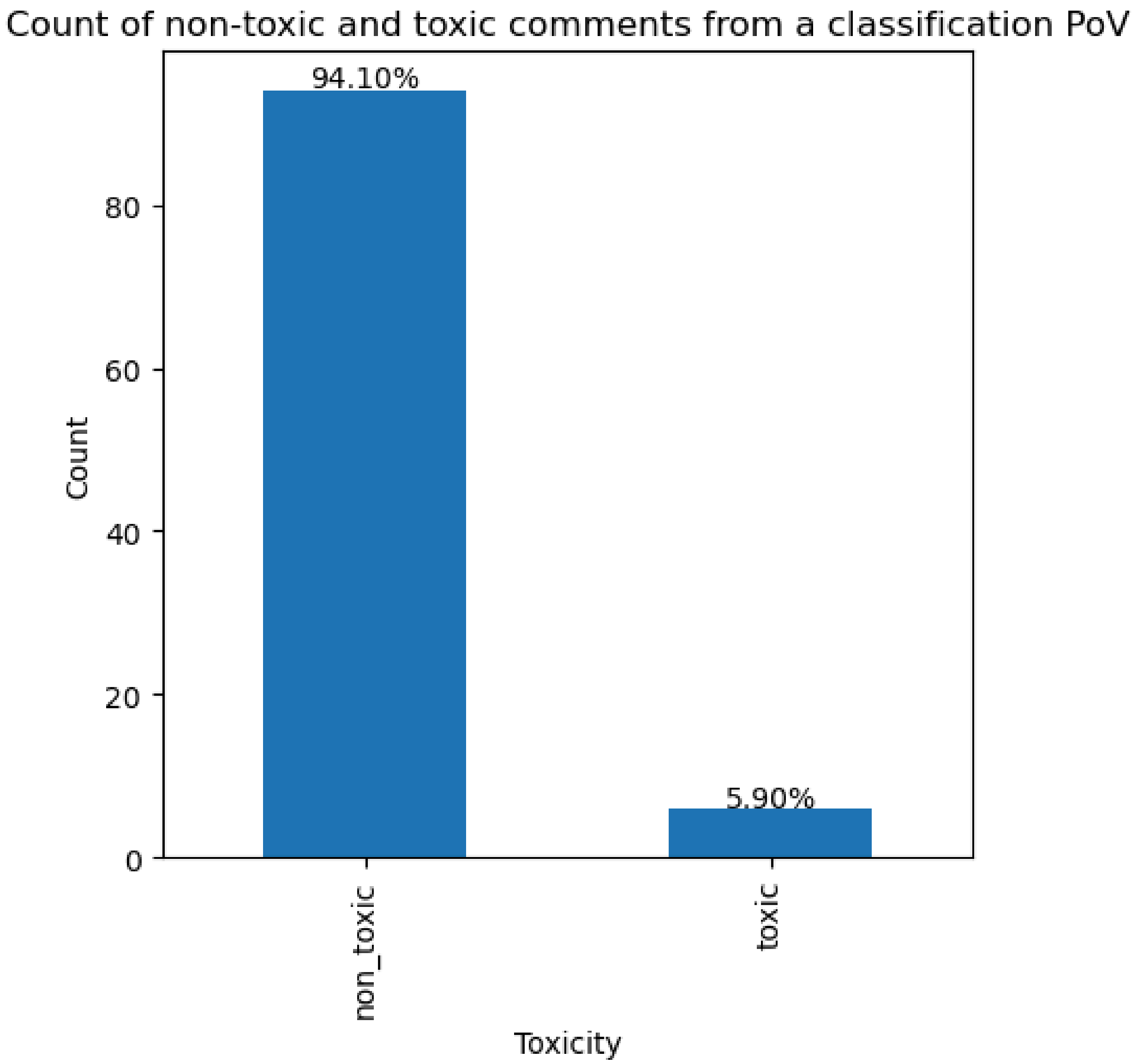
**Distribution of toxicity rating across the dataset:**




Thus, we can see that most of the comments are non-toxic ( which have a toxicity score of 0)

**Percentage of Toxic and Non-Toxic comments when considered as a classification problem:**

**(Considering the threshold for being toxic as 0.5)**




**Thus we can see that the dataset we are using is not uniform and mostly skewed with most of the comments being non toxic as we saw earlier.**

# Looking for sub-features:

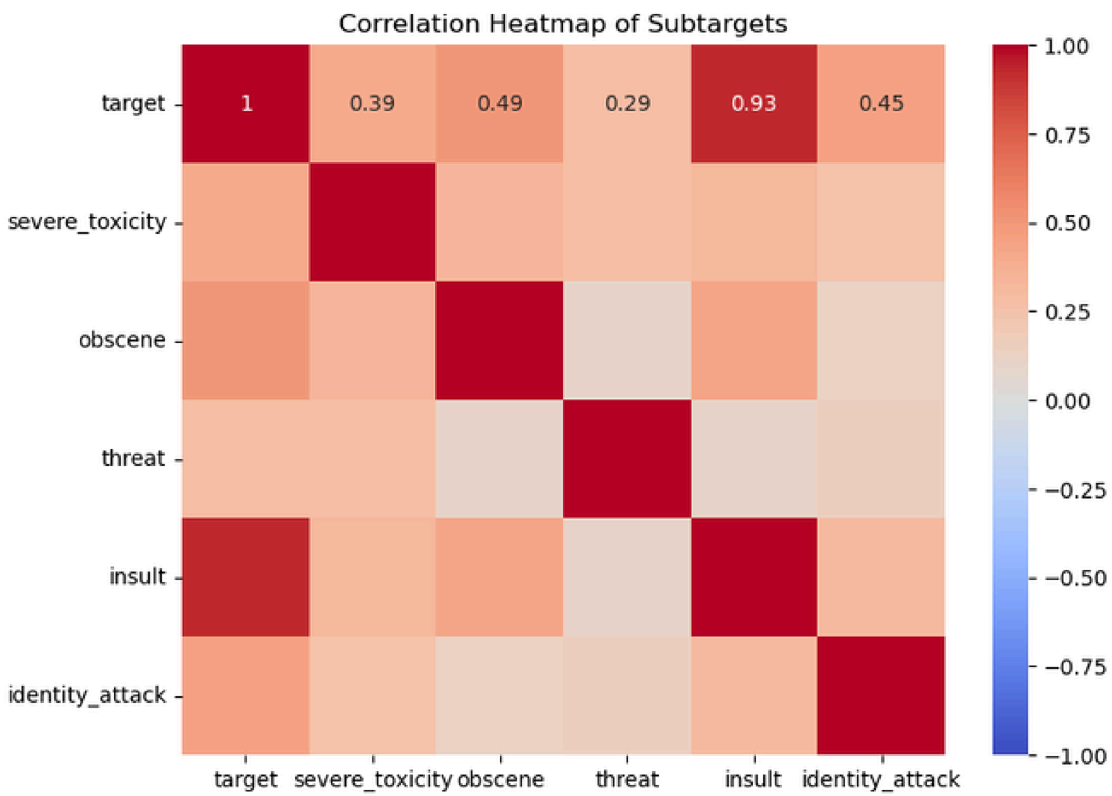
**There are several sub-classifications of a comment if it is found out to be toxic.**

1. severe_toxicity
2. obscene
3. identity_attack
4. insult
5. threat

### Their datatypes:

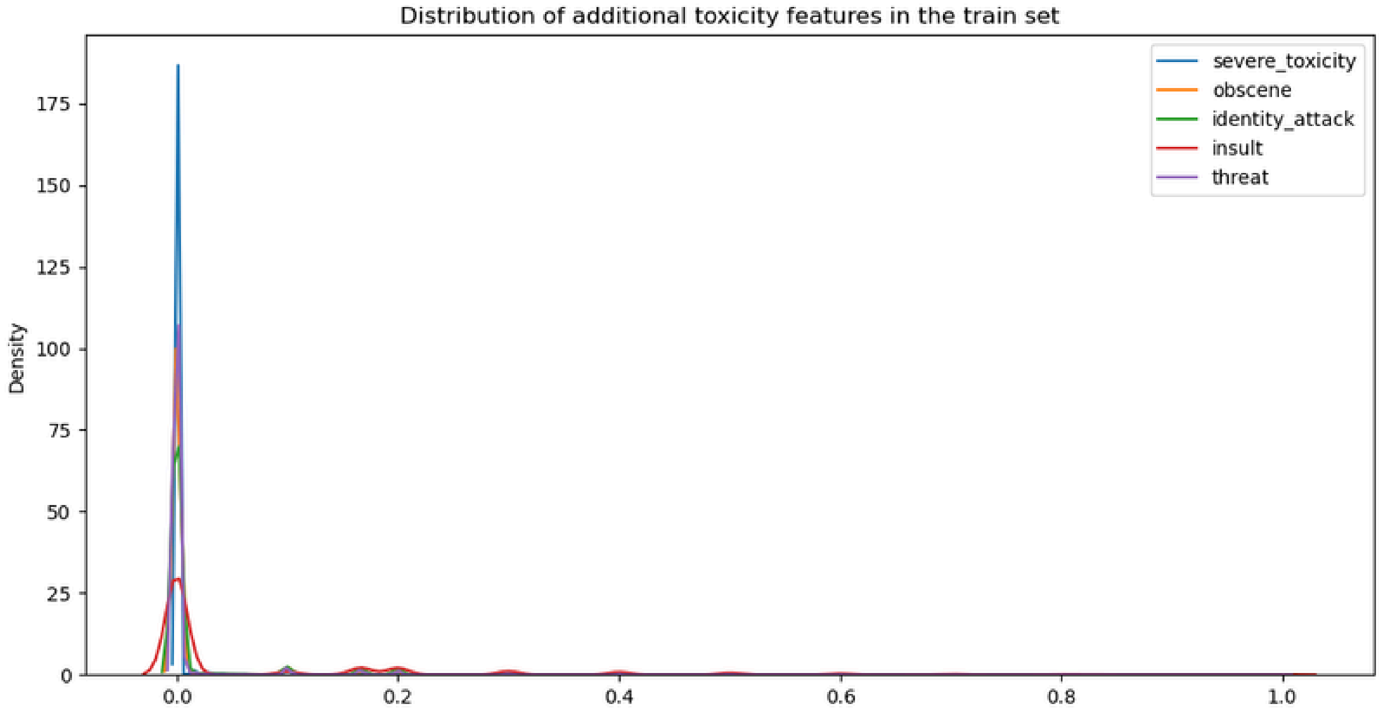
```
1   target                          float64
2   comment_text                    object
3   severe_toxicity                 float64
4   obscene                         float64
5   identity_attack                 float64
6   insult                          float64
7   threat                          float64
```

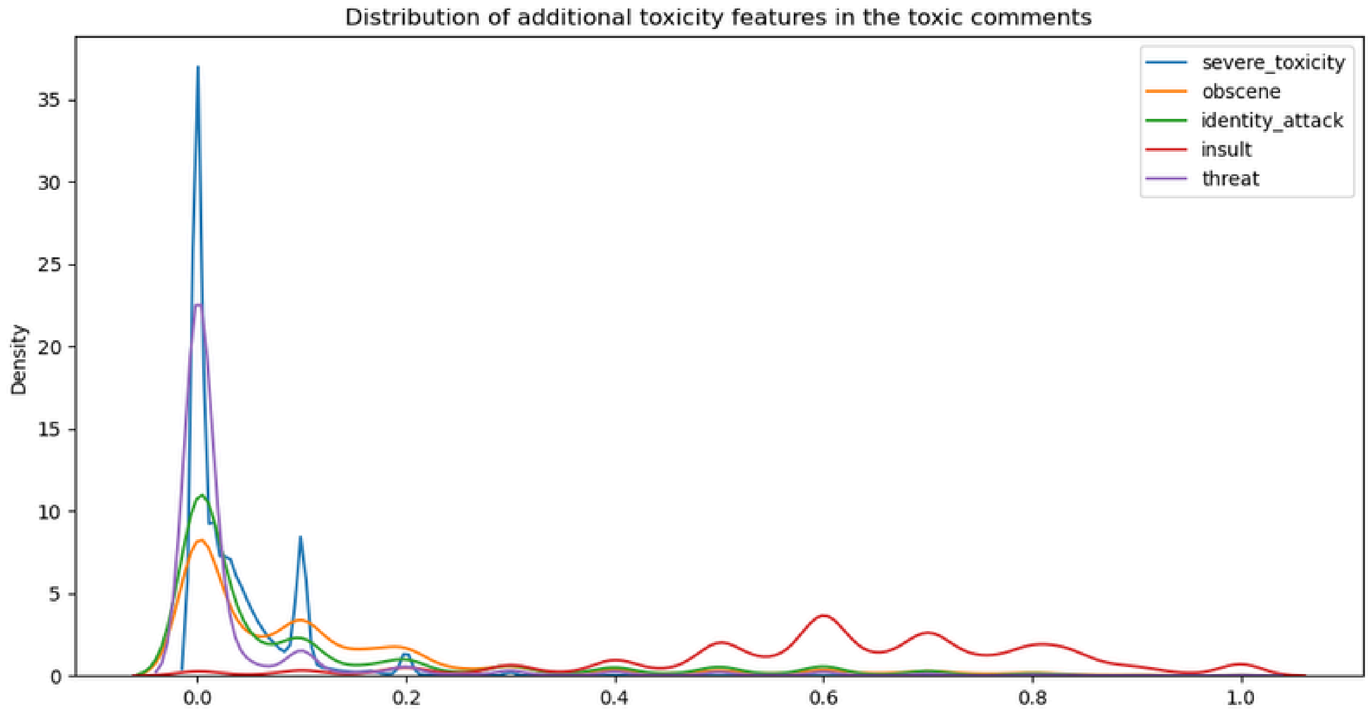
## Finding correlation among these sub-features and the target variable:




**Thus, we can see that all of them are positively correlated, since all of them have values >1.**

# Their distribution in the Entire Dataset, and Toxic comments Dataset:

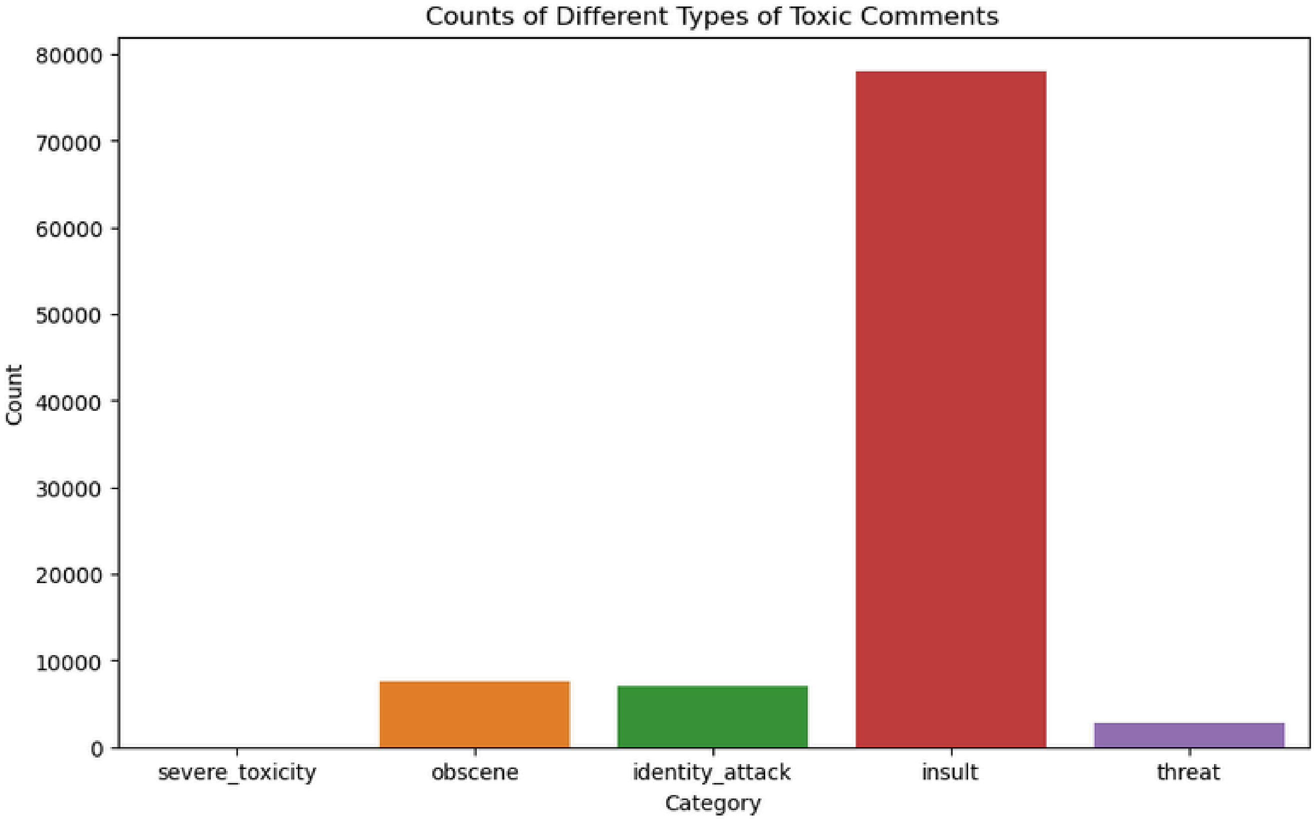



**Thus we can see, that most of the comments have their ratings close to zero, and very few have higher ratings.**

**Among the toxic comments, most of them are insults.**

**Cross verification by plotting bar plots ( considering 0.5 to be the threshold for each of these subclasses):**

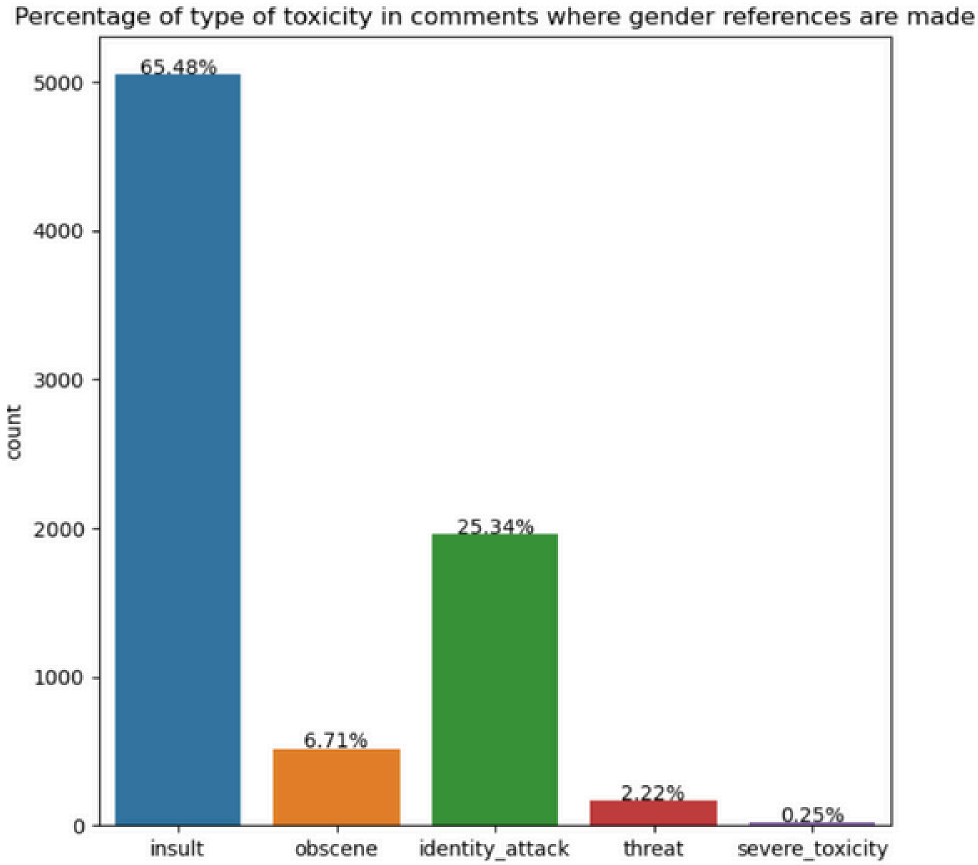



**Our observation that most of the toxic comments are insults turned out to be correct.**

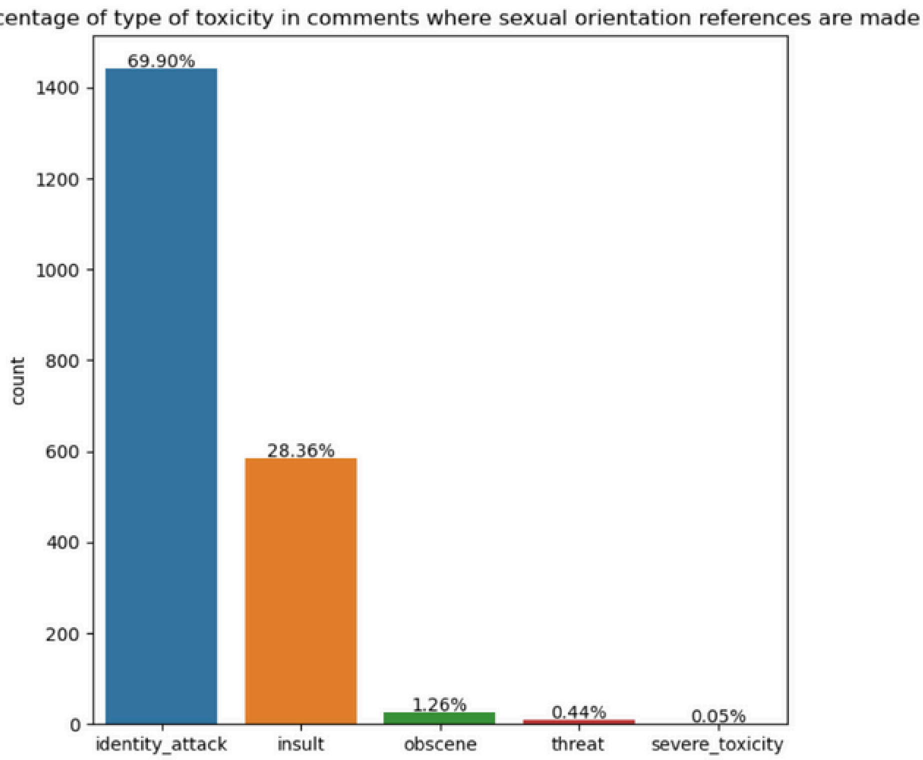
**Several other Features are present in the dataset:**

1. **Gender related**
2. **Sexual orientation related**
3. **Race/Ethnicity related**

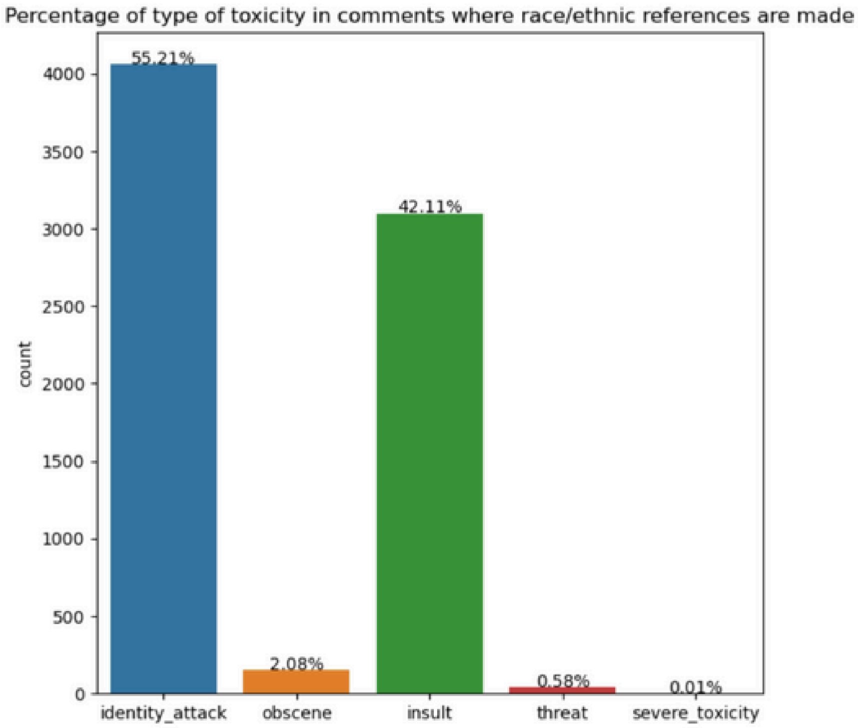
**Let us try to analyse which kind of reference hints towards which kind of toxicity:**




Thus we can see that toxic comments that include gender references are mostly Insults in nature.

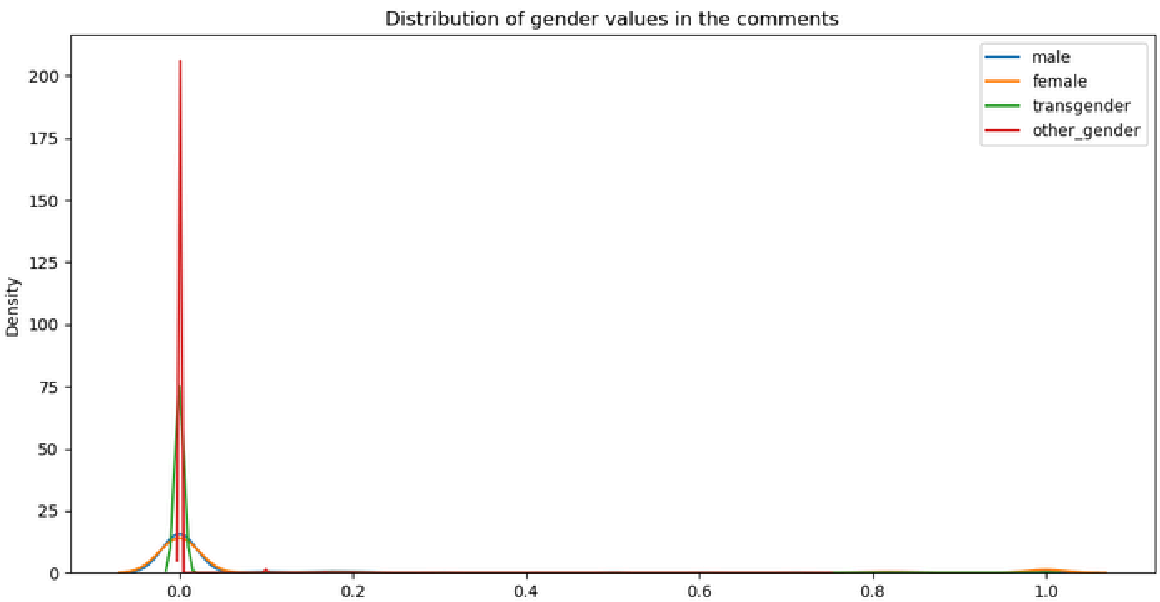



Thus we can see that mostly the toxic comments where sexual orientation references are made come under the category of identity_attack.




We can observe that toxic comments that include racial/ethnicity comments mostly fall under the category of identitty attack or insult.

**Observing the distribution of these values show that most of the comments do not have references of these kind and hence point towards zero, so we do not plot this distribution for other similar features as we cannot get any major insights from them.**



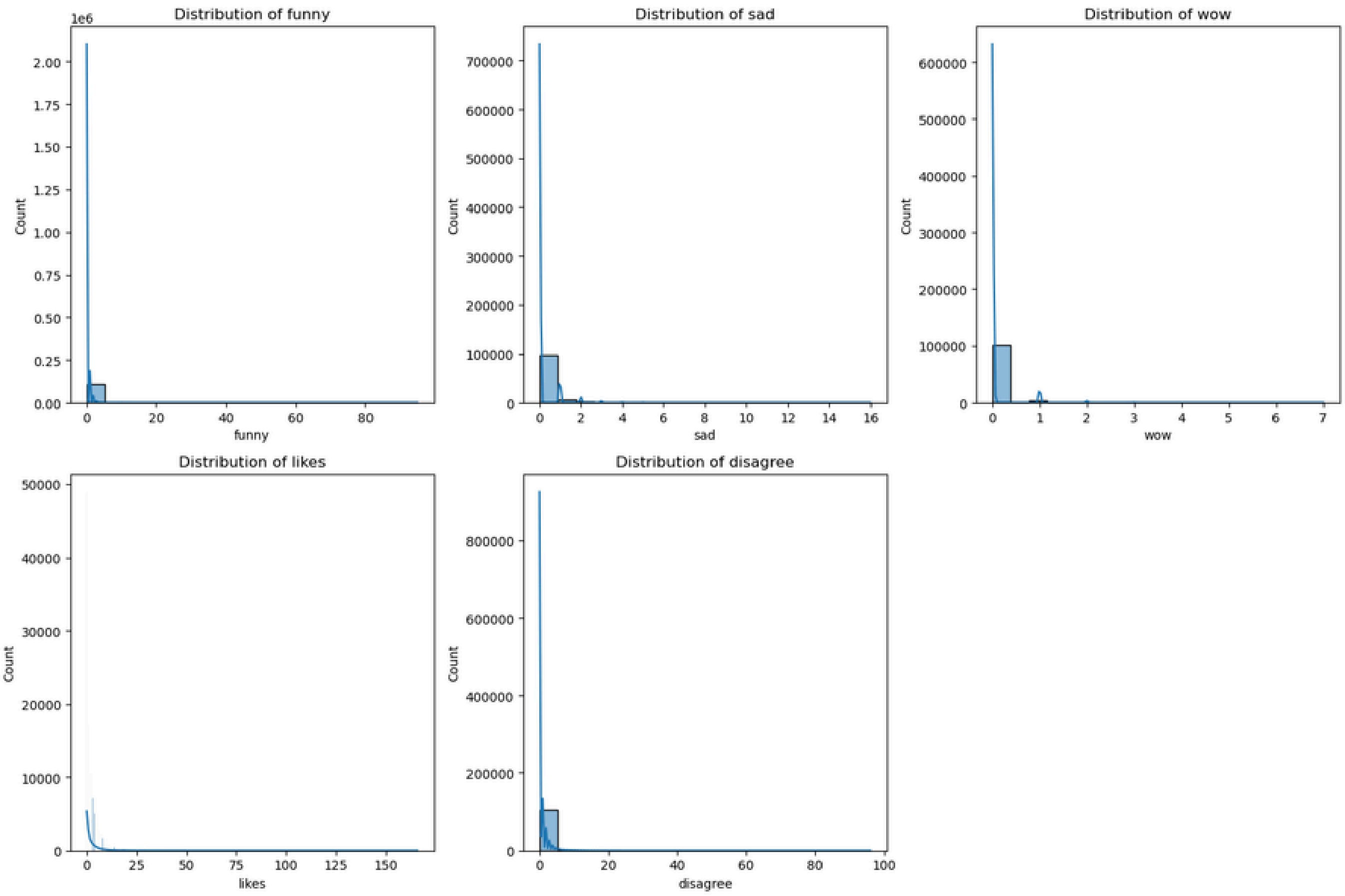
# User feedback Features:

- funny
- sad
- wow
- likes
- disagree

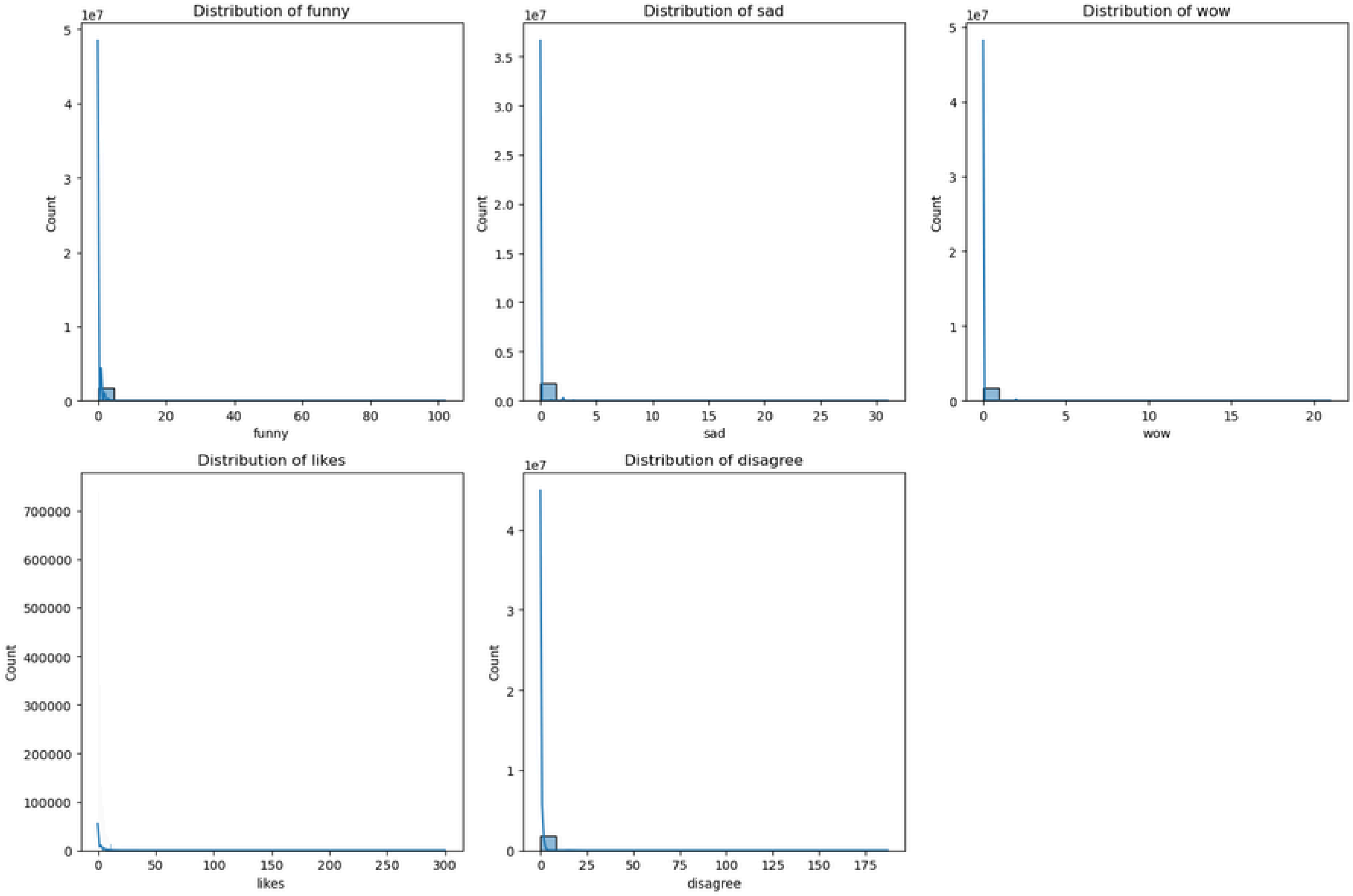
**Their statistics:**

| | funny | sad | wow | likes | disagree |
|---|---|---|---|---|---|
| count | 1.804871e+06 | 1.804871e+06 | 1.804871e+06 | 1.804871e+06 | 1.804871e+06 |
| mean | 2.779246e-01 | 1.091175e-01 | 4.420704e-02 | 2.446166e+00 | 5.843692e-01 |
| std | 1.055308e+00 | 4.555366e-01 | 2.449361e-01 | 4.727925e+00 | 1.866590e+00 |
| min | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 25% | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 50% | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 1.000000e+00 | 0.000000e+00 |
| 75% | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 3.000000e+00 | 0.000000e+00 |
| max | 1.020000e+02 | 3.100000e+01 | 2.100000e+01 | 3.000000e+02 | 1.870000e+02 |

**Their distribution in toxic comments:**



**Their distribution in overall dataset:**




**Since most of the values are close to 0 again( for toxic as well as non toxic comments) , there is no valuable insights that can be gained from these features.**

# Some more statistical Explorations:

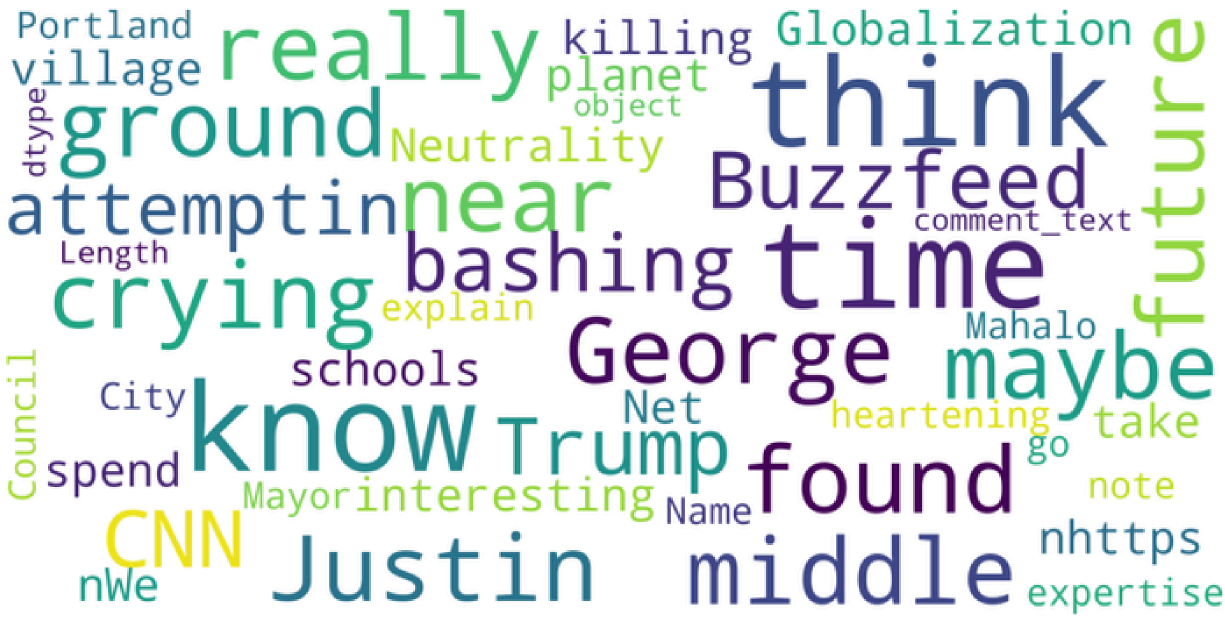
```
Average length of toxic comments: 252.8707510475582
Average length of non-toxic comments: 300.01503680156947
```

```
Average number of exclamations in toxic comments :0.3984948984385276
Average number of exclamations in non-toxic comments : 0.21415799151335377
```

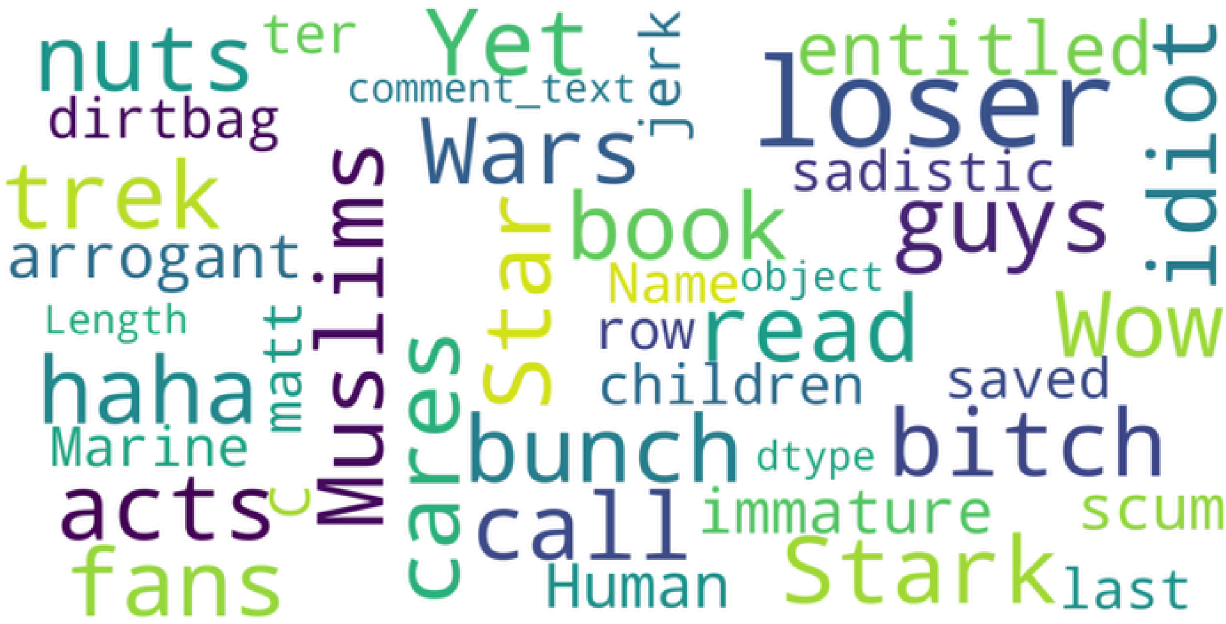
**Toxic comments are being noticed to be slightly shorter than non toxic comments.**

**Also, toxic comments have more use of special character '!' and most probably other special characters too.**

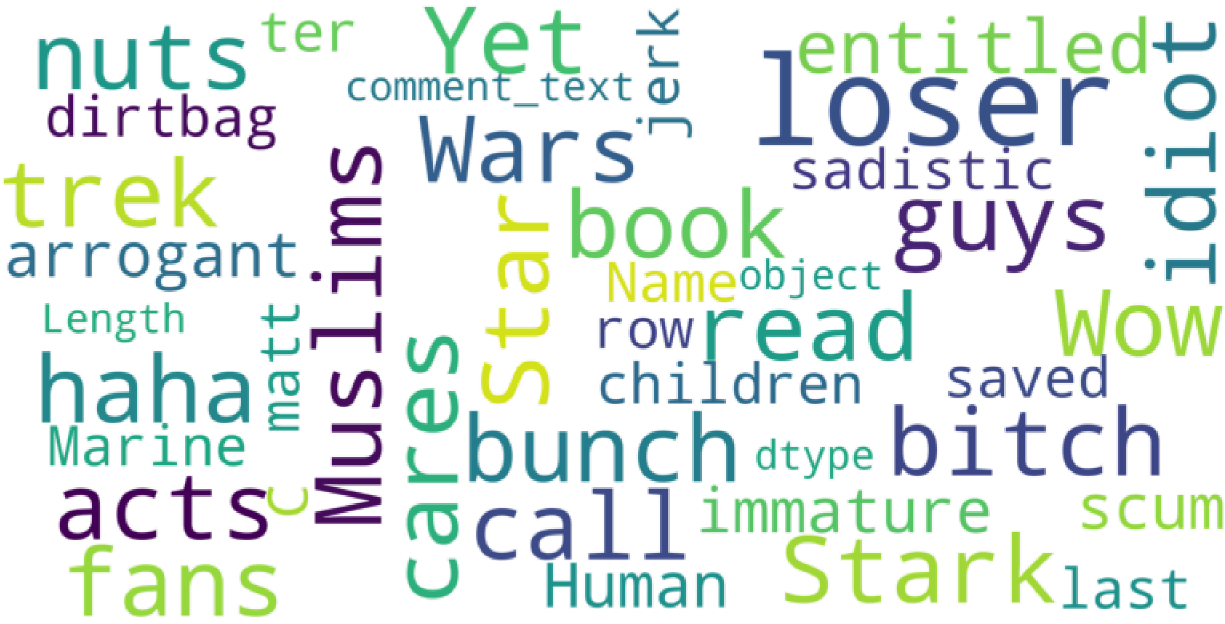
## Vizualising prevalent words using WordCloud:




Prevalent words in all comments




Prevalent words in toxic comments




Prevalent words in insult comments

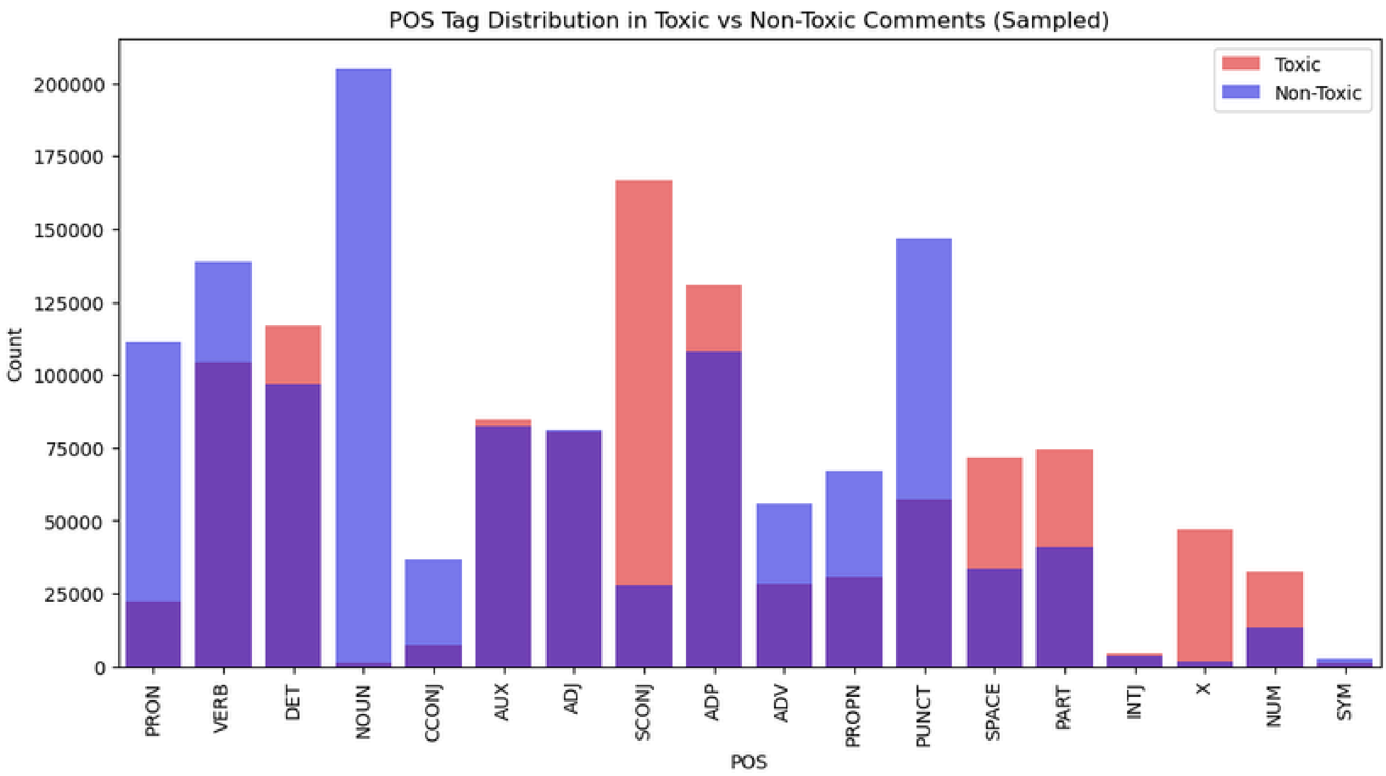

Prevalent words in obscene comments


Prevalent words in severe toxic comments


Prevalent words in clean/non-toxic comments

**Note: These wordclouds have been created from a sample of the data due to computational resource limitations. They are good for providing a generalised idea but may not be completely accurate.**

# PoS Tagging for additional Insights:




- Although the data used here for POS Tagging is random sampled, I'm pretty sure that the distribution would change if the entire set of toxic and non toxic comments was considered for this analysis.
- However, one major analysis which I think is valid is that toxic comments are more likely to contain words that do not fall into any specific category (denoted by X), probably because toxic messages are more likely to contain short forms, slang words, typos and so on which may not always be grammatically correct.

## Data Preprocessing:

**Handling missing values- Data without actual text is of no use to us.**

```
#removing rows that have the text missing
train_df.dropna(subset=['comment_text'], inplace=True)
```

**Lemmatization and removal of Stopwords:**
**((I have used Spacy library to achieve this)**

```
def preprocess(text_string):
    text_string = text_string.lower()
    text_string = re.sub(r'[^A-Za-z0-9]+', ' ', text_string) #Remove special characters and punctuations
    doc = nlp(text_string)
    new_text = []
    for token in doc:
        if token.text.lower() not in STOP_WORDS:
            new_text.append(token.lemma_)
    text_string = ' '.join(new_text)
    return text_string
```

# Training the models:

## 1. Bag of Words (BoW) approach:

### a. SGD Regressor ( Stochastic Gradient Descent Regressor)

```
#Hyperparameters tuning
param_grid={ 'alpha' :[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100],
            'penalty' :['l1', 'l2']}
sgd=SGDRegressor()
```

```
Best Parameters: {'alpha': 0.0001, 'penalty': 'l1'}
Mean Squared Error on CV set: 0.024298003603473017
```

### b. Decision Tree Regressor

```
param_grid = {'max_depth': [3, 5, 7],
              'min_samples_leaf': [10, 100, 1000]}

dt=DecisionTreeRegressor()
```

```
Best Parameters: {'max_depth': 7, 'min_samples_leaf': 100}
Mean Squared Error on CV set: 0.031656608167548304
```

#### Feature weights in SGDreg

| | weights |
|---|---|
| idiot | 0.434650 |
| stupid | 0.350389 |
| stupidity | 0.323367 |
| moron | 0.320754 |
| pathetic | 0.301164 |
| hypocrite | 0.283425 |
| crap | 0.266751 |
| dumb | 0.266327 |
| idiotic | 0.264953 |
| ass | 0.246647 |
| ignorant | 0.234407 |
| clown | 0.228581 |
| damn | 0.215632 |
| scum | 0.211157 |
| ridiculous | 0.208342 |
| fool | 0.206690 |
| loser | 0.205507 |
| jerk | 0.198379 |
| silly | 0.197860 |
| shit | 0.184310 |

#### Feature weights in decision Tree

| | weights |
|---|---|
| stupid | 0.354878 |
| idiot | 0.257678 |
| ignorant | 0.086890 |
| pathetic | 0.074313 |
| dumb | 0.072259 |
| fool | 0.069534 |
| stupidity | 0.065688 |
| year | 0.002779 |
| don fool | 0.002214 |
| people | 0.002021 |
| don | 0.001853 |
| good | 0.001797 |
| thing | 0.001514 |
| state | 0.001183 |
| time | 0.000840 |
| way | 0.000754 |
| go | 0.000627 |
| need | 0.000608 |
| issue | 0.000584 |
| public | 0.000438 |

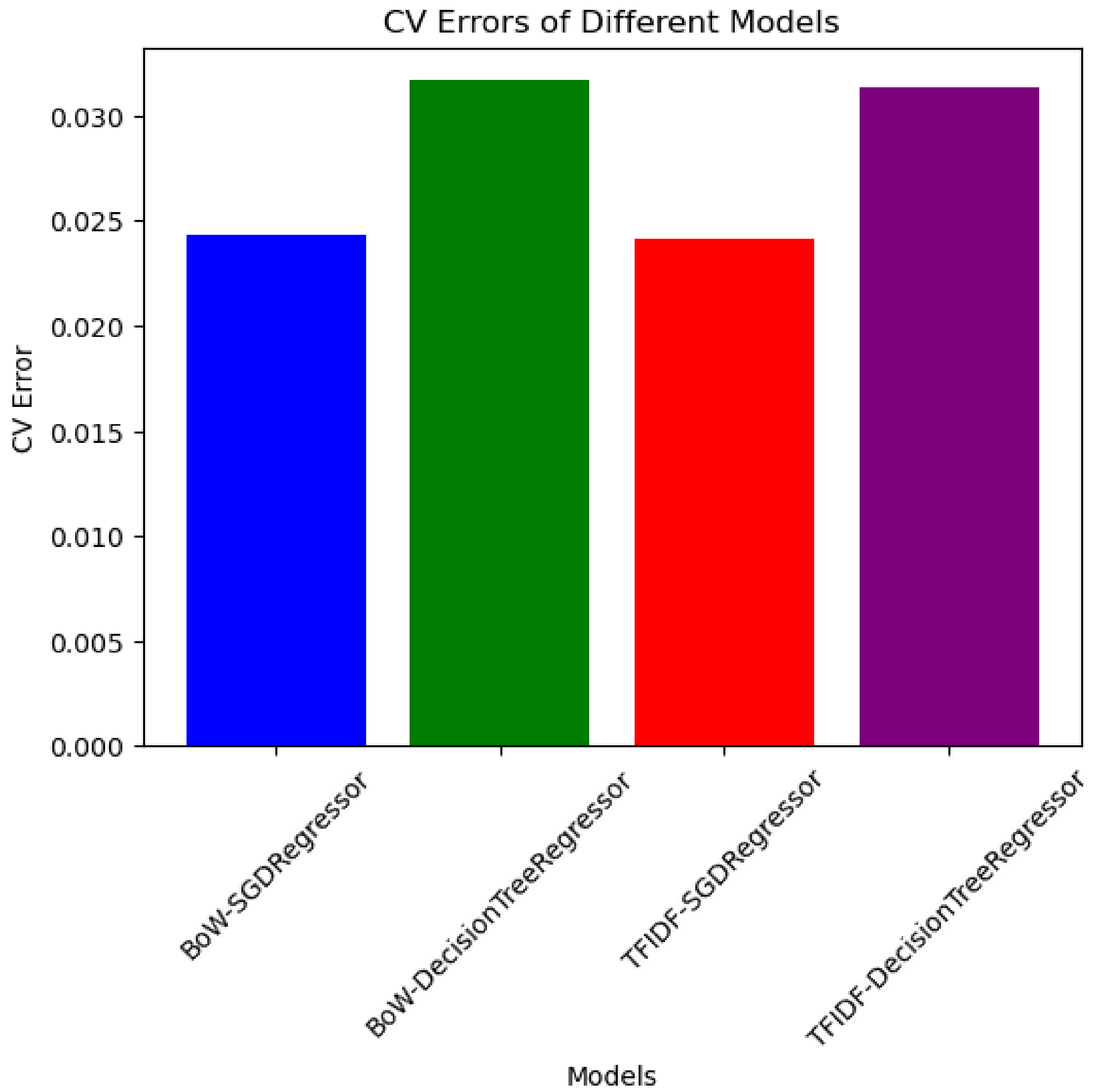
# 2. TF-IDF approach:

### a. SGD Regressor ( Stochastic Gradient Descent Regressor)

```
Best Parameters: {'alpha': 1e-05, 'penalty': 'l2'}
Mean Squared Error on CV set: 0.02415963883069067
```

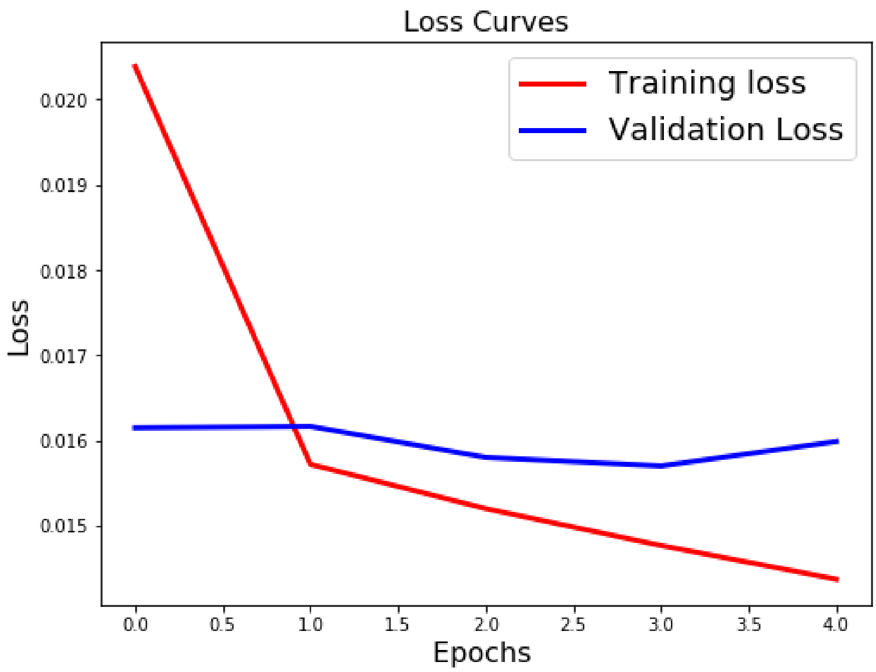
### b. Decision Tree Regressor

```
Best Parameters: {'max_depth': 7, 'min_samples_leaf': 10}
Mean Squared Error on CV set: 0.03138149202136317
```



# LSTM

**I used two LSTM layers and a dense layer, output function used was Sigmoid since we want values from 0 to 1, and optimizer used was rmsprop.**




## Outcomes of LSTM Model:

- LSTM Model : Mean Squared Error on CV set: 0.0157

**This was the best model trained so far with minimum Mean Squared Error on cross validation set.**