# Toxic Comment Classification

The growing prevalence of online abuse and harassment has created widespread concern among internet users worldwide. This unpleasant environment frequently prevents people from expressing themselves and engaging in constructive discussions, essentially silencing the exchange of alternative viewpoints. In response, platforms are continuously seeking for ways to improve discourse. However, many people continue to struggle with effectively managing and moderating user interactions, resulting in restricted or completely disabled comment areas in many online communities.

Addressing this issue is crucial for creating a safer and more inclusive digital environment in which users may engage in respectful and productive arguments without fear of harassment or abuse. To do this, more effective ways for recognizing and reducing toxic conduct in online discussions must be created.

The dataset is separated into two parts: training and testing. The train dataset contains labeled comments that may be used to create and train models, whilst the test dataset contains unlabeled comments that can be used to assess the performance of the produced models.

The comments have benn labelled by human raters for toxic behaviour/character. The types of toxicity are:

- Toxic
- Severe_toxic
- Obscene
- Threat
- Insult
- Identify_hate

The training dataset has 159,571 observations while the testing dataset has 153,164 observations. The test_labels dataset contains certain observations with labels of -1 which are not scored since it was originally used in a Kaggle competition.

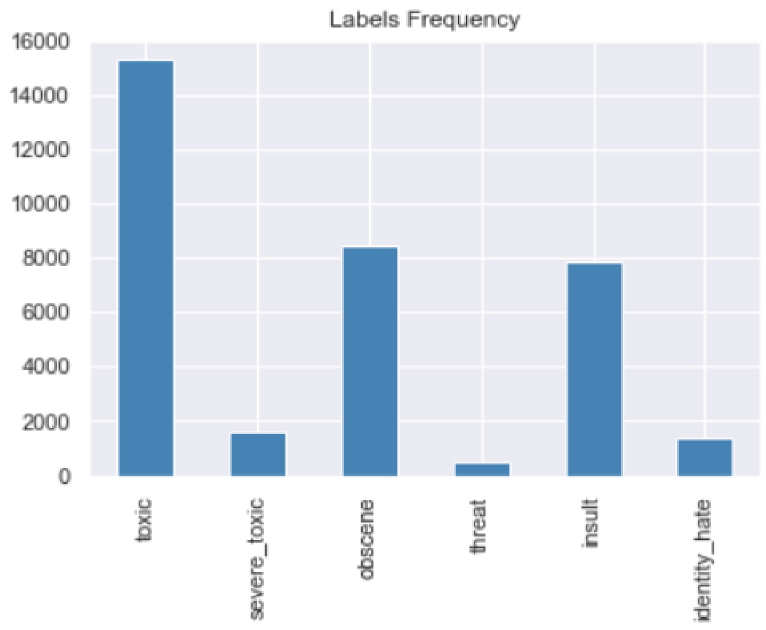
Model interpretability and computational efficiency need to be prioritized when creating machine learning algorithms that predict toxicity level of given comments. The main reason is to make sure that the resulting models will have a quick turnaround time and could easily be used in existing systems during live monitoring. It is necessary to encourage the creation of workable and effective solutions that deal with reducing negativity in online discussions while creating a more welcoming and cheerful digital environment.

As a result, the issue of online toxicity requires quick attention and resolution. By combining the collective wisdom of the data science community with the promise of machine learning, safer and even more inclusive digital environments may be effectively achieved for everyone.

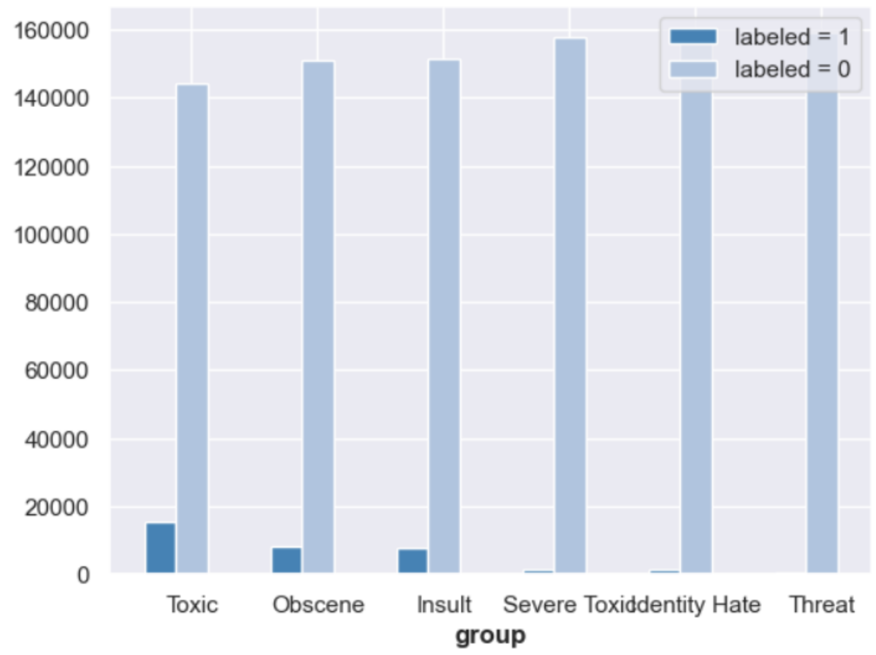
Dataset link: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

## Data Processing and EDA

Because all of our data is text comments, we created our own tokenize() code that removes punctuation and special characters, stems and/or lemmatizes the comments, and filters out those with lengths less than three. After comparing vectorizers (TFIDFVectorizer and CountVectorizer), we decided on TFIDFVectorizer because it performed better.



plot for the labeled data frequency. There is significant class imbalance since majority of the comments are considered non-toxic.



Below is the word cloud for the toxic comments mentioned as part of dataset.

Most common words assosiated with toxic comment

Below is the word cloud for the obscene comments mentioned as part of dataset.

Most common words assosiated with obscene comment

An undesired statistical observation would be that a good fraction, the largest one possibly, is made up of decent comments. Class labels like threat however were assigned to very few other than mentioned above. This is an indication that one must be ready to handle classification problems due to class imbalances and as such one may resort to applying alternative sampling techniques, picking suitable performance measures and opting for stable methods when dealing with it.

## Evaluation Metric Selection

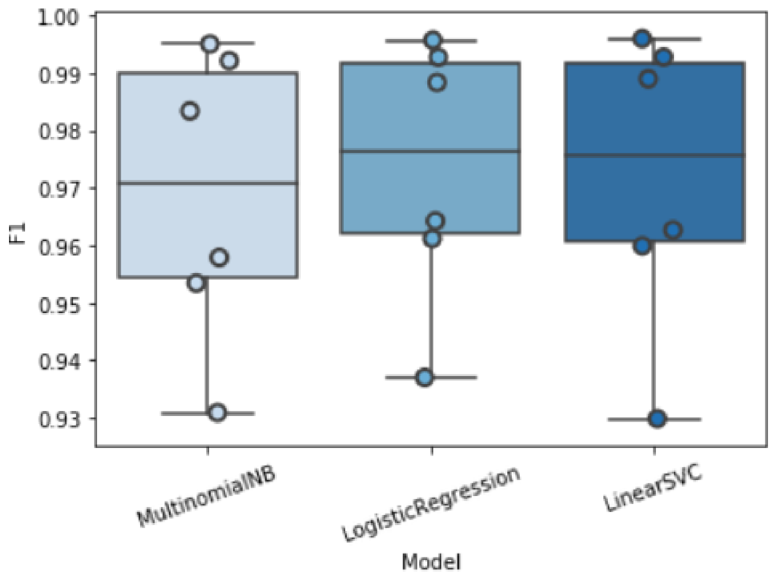
Choosing evaluation metrics During modelling phase we are compelled to opt for different measures of assessment that can aid in the evaluation of models depending on the nature of our data:

- Recall
- F-score
- Hamming loss

## Model Comparison

We first utilized k-fold cross validation and compared the performance of the three models—Multinomial Naive Bayes, Logistic Regression, and Linear SVC—without making any hyperparameter adjustments. Multinomial Naive Bayes served as our baseline model. The performance of Logistic Regression and Linear SVC surpasses that of Multinomial Naive Bayes.

Upon examining the test data performance of these models, we find that, on average, Linear SVC outperforms the other models based on F1 score, while Multinomial Naive Bayes performs worse than the other two models.



The Above result table and plot displayed above compare the various models after they have been trained and demonstrate how they function with test data.

Notice that Multinomial Naive Bayes does not perform as well as the other two models while Linear SVC in general out performs the others based on F1 score.

So Overall, without any hyperparameter tuning Linear SVC performs the best initially.

## Manual Hyperparameter Tuning

The F1 score of the Logistic Regression model increased to an average of 0.9479 after correcting for the unbalanced data, while the Linear SVC score increased to 0.9515.

Before Hyperparameter Tuning:

| | Model | F1 | Recall | Hamming_Loss | Training_Time |
|---|---|---|---|---|---|
| **0** | LogisticRegression | 0.947932 | 0.934071 | 0.065929 | 1.888246 |
| **1** | LinearSVC | 0.951508 | 0.941634 | 0.058366 | 7.990794 |

After Hyperparameter Tuning:

| | Model | F1 | Recall | Hamming_Loss | Traing_Time |
|---|---|---|---|---|---|
| **0** | LinearSVC | 0.971706 | 0.971524 | 0.028476 | 4.592842 |
| **1** | LogisticRegression | 0.973227 | 0.974330 | 0.025670 | 8.529908 |

Similar increase of F1 score can be observed for the Logistic regression.

## Grid Search

We found the "optimal" hyperparameters for the models with the aid of grid search, and we were able to average the best score of 0.9566 for Logistic Regression and 0.9585 for Linear SVC.

## Ensembling

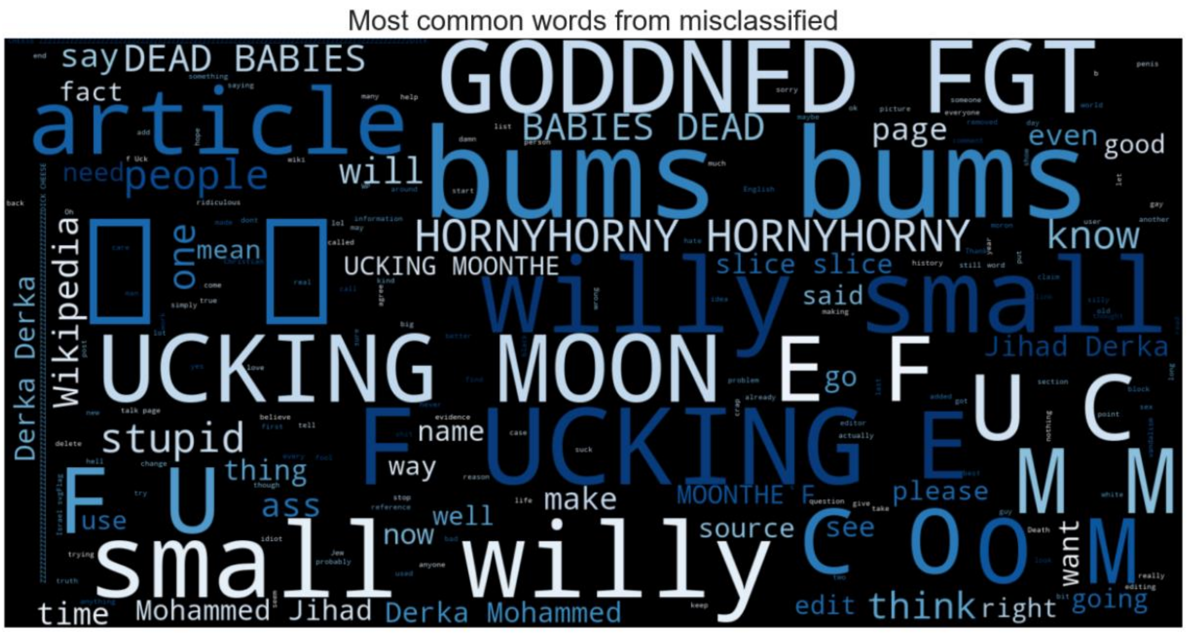
We first experimented with a few tree-boosting models before using a vote classifier to combine one of the boosting models with the foundational models from earlier sections. Using assembling, we obtain an F1 score of 0.973566 and a Hamming Loss of 0.024639.

| | Model | F1 | Recall | Hamming_Loss | Traing_Time |
|---|---|---|---|---|---|
| **0** | AdaBoostClassifier | 0.967605 | 0.969771 | 0.030229 | 43.462685 |
| **1** | GradientBoostingClassifier | 0.968930 | 0.971647 | 0.028353 | 202.102620 |
| **2** | XGBClassifier | 0.972683 | 0.973129 | 0.026871 | 33.490446 |

| | Model | F1 | Recall | Hamming_Loss | Training_Time |
|---|---|---|---|---|---|
| **0** | Ensemble | 0.973294 | 0.973994 | 0.026006 | 36.859941 |

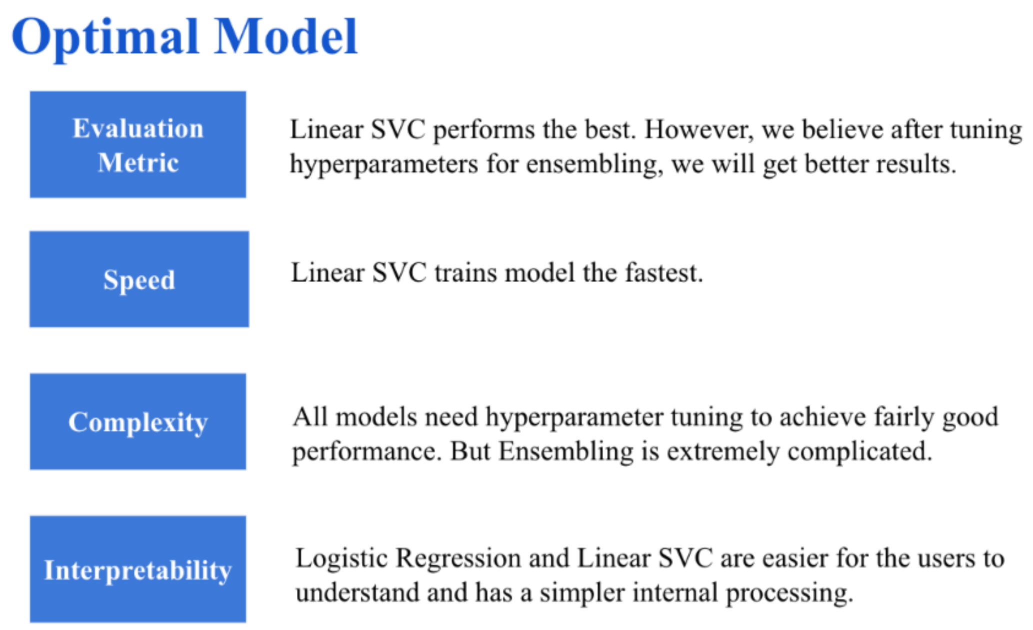
## Testing

When test set is used, 1347 items misclassified as non-toxic which originally are toxic. Most common words which are misclassified as nontoxic are shown in below word cloud.



"Ucking" is a common word in test set which was not learnt. In training set also there is very less frequent occurrence. So, it makes sense why it did not learn.

## Overall Results



### Optimal Model

| | |
|---|---|
| **Evaluation Metric** | Linear SVC performs the best. However, we believe after tuning hyperparameters for ensembling, we will get better results. |
| **Speed** | Linear SVC trains model the fastest. |
| **Complexity** | All models need hyperparameter tuning to achieve fairly good performance. But Ensembling is extremely complicated. |
| **Interpretability** | Logistic Regression and Linear SVC are easier for the users to understand and has a simpler internal processing. |

When it comes to evaluation metrics, Linear SVC excels. However, we anticipate improved outcomes after fine-tuning the ensembling hyperparameters. In addition, the fastest trains are linear SVC trains. In terms of interpretability, linear SVC has a simpler internal processing and is also easier for users to understand. For this reason, we decide that Linear SVC is the best model.