# Toxic Comment Classification
Capstone Three Final Report

## 1. Problem

The 21st century is the age of social media. On one hand, many small businesses are benefiting and on the other, there is also a dark side to it. Thanks to social media, people are aware of ideas they are not used to. While few people take it positively and make an effort to get used to it, many people start going in the wrong direction and start spouting malicious words. So many social media apps take the necessary steps to remove these comments in order to predict their users, and they do so using NLP techniques.

## 2. Introduction

Maintaining constructive and inclusive online conversations is important for platform providers. One way to do this is by automatically identifying and classifying toxic comments, such as hate speech, threats, and insults. This can be challenging due to the complexities of natural language processing, including long-range dependencies and unconventional or misspelled words. Previous research has suggested using techniques like bidirectional recurrent neural networks with attention and pre-trained word embeddings to tackle these challenges, but these approaches may not always perform well on real-world data due to limited variance in methods and training data. In this study, I examine the Kaggle Toxic Comment Classification Challenge Twitter Dataset, which includes multilingual user comments and tweets, and presents both a multi-class and a multi-label classification task. I propose an ensemble of state-of-the-art classifiers to analyze false negatives and false positives and gain insights into the open challenges that all approaches share.

## 3. Data Exploration

Data is used from the “Toxic Comment Classification Challenge” on Kaggle, This specific dataset consists of a large number of Wikipedia comments which have been labeled by human raters about each comment's toxicity. Train-set and test-set include 159571 and 153164 comments respectively. The purpose is to build a model that classifies the comments of the test set into six different categories regarding the types of toxicity. More precisely, the basic goal of this assignment is to train a Natural Language Processing model, which predicts the probabilities of each type of toxicity for each comment. The following table provides information about the label frequencies in the train set. It can be easily observed that the most multitudinous class is "Toxic" which is present in 15294 comments. This is reasonable because toxic is a very general label compared to the others.

| Obscene | Insult | Toxic | Severe toxic | Identity hate | Threat |
|---|---|---|---|---|---|
| 8449 | 7877 | 15294 | 1595 | 1405 | 478 |

Table 1. Train set Class-Frequencies

- **Imbalanced Data and Bias**

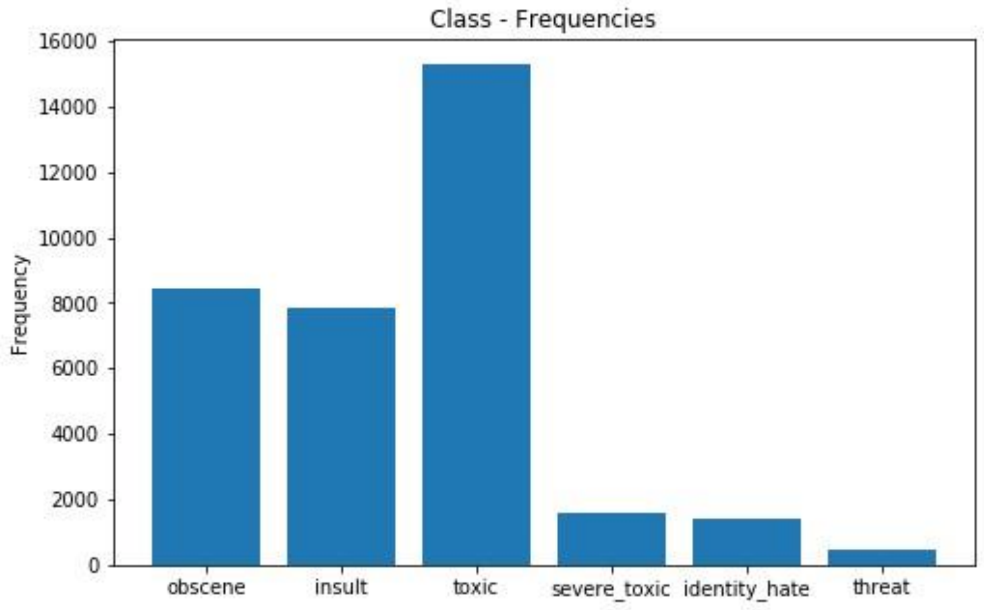


Chart 1. Frequency of each class in the training set.

The characteristic of the data set is a large number of unmarked comments. About %90 of the entire dataset consists of comments that do not belong to any class. This fact can have a strong impact on classification models. There are many solutions to the problem of imbalanced data, such as undersampling/oversampling and general methods of artificially balancing data. Unfortunately in this case this is very difficult due to the linguistic nature of the data. Also, comments cannot be discarded without captions because they would lose important information.

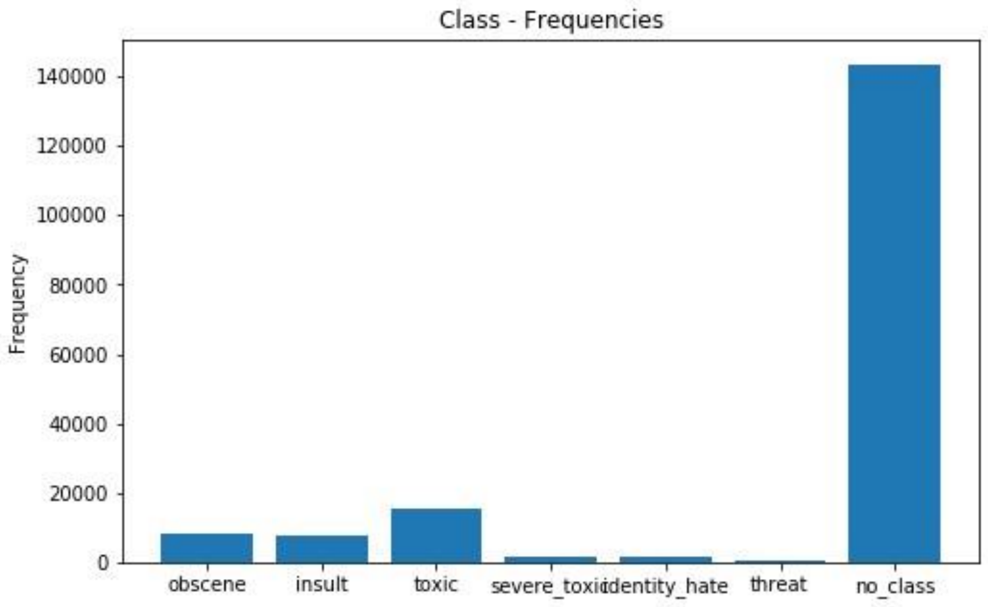


Chart 2. Frequencies of imbalanced data

## 4. Method

This dataset was used to classify the comments as toxic or non-toxic. For this project, textual data preprocessing techniques are used first. After that, basic NLP methods like TF-IDF for converting textual data into vectors are performed, and then machine learning algorithms are used to label the comments and find out KPIs.

## 5. Exploratory Data Analysis

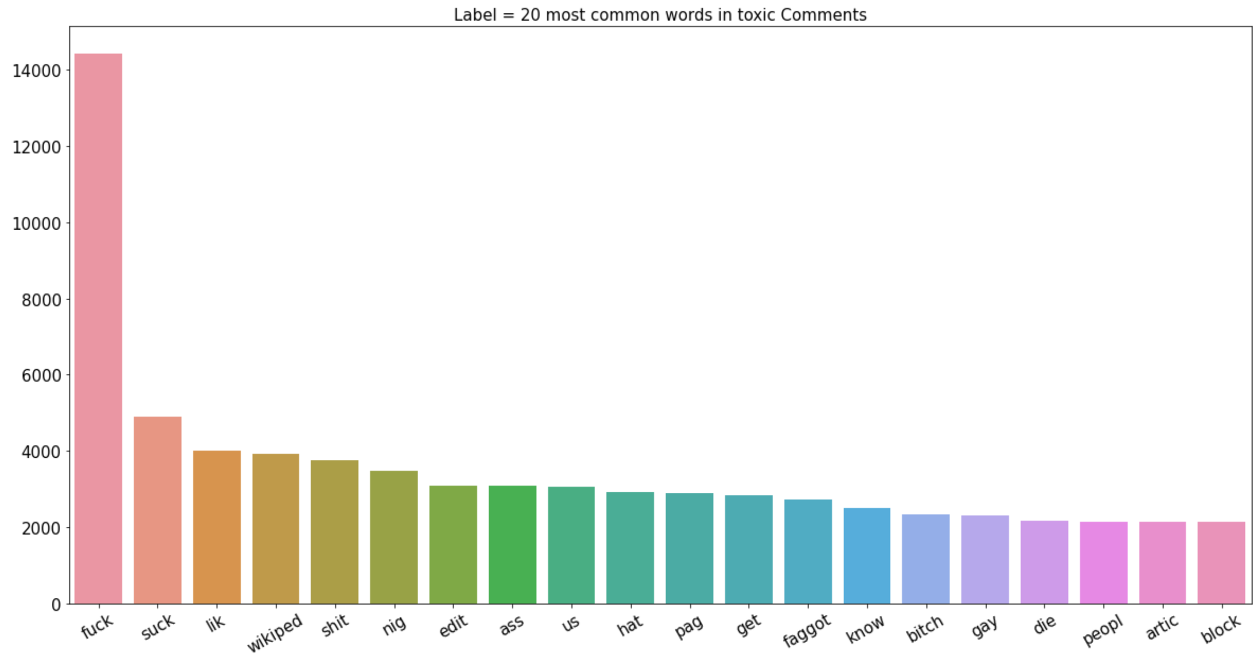


Chart 3. 20 Most Common Words in Toxic Comments

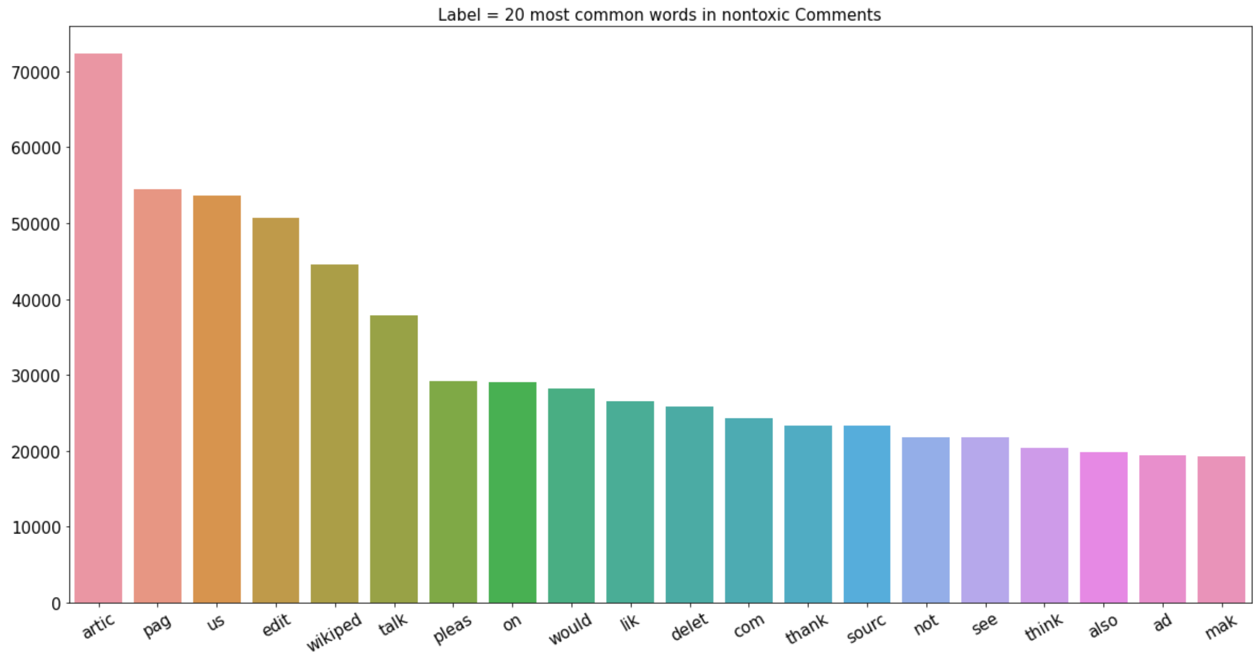


Chart 4. 20 Most Common Words in Nontoxic Comments

The above bar plots show the 20 most common words for a given label or category, with the words on the x-axis and the counts on the y-axis. The plots are used to visualize the distribution of words within a particular category and to identify the most frequently used words in that category.

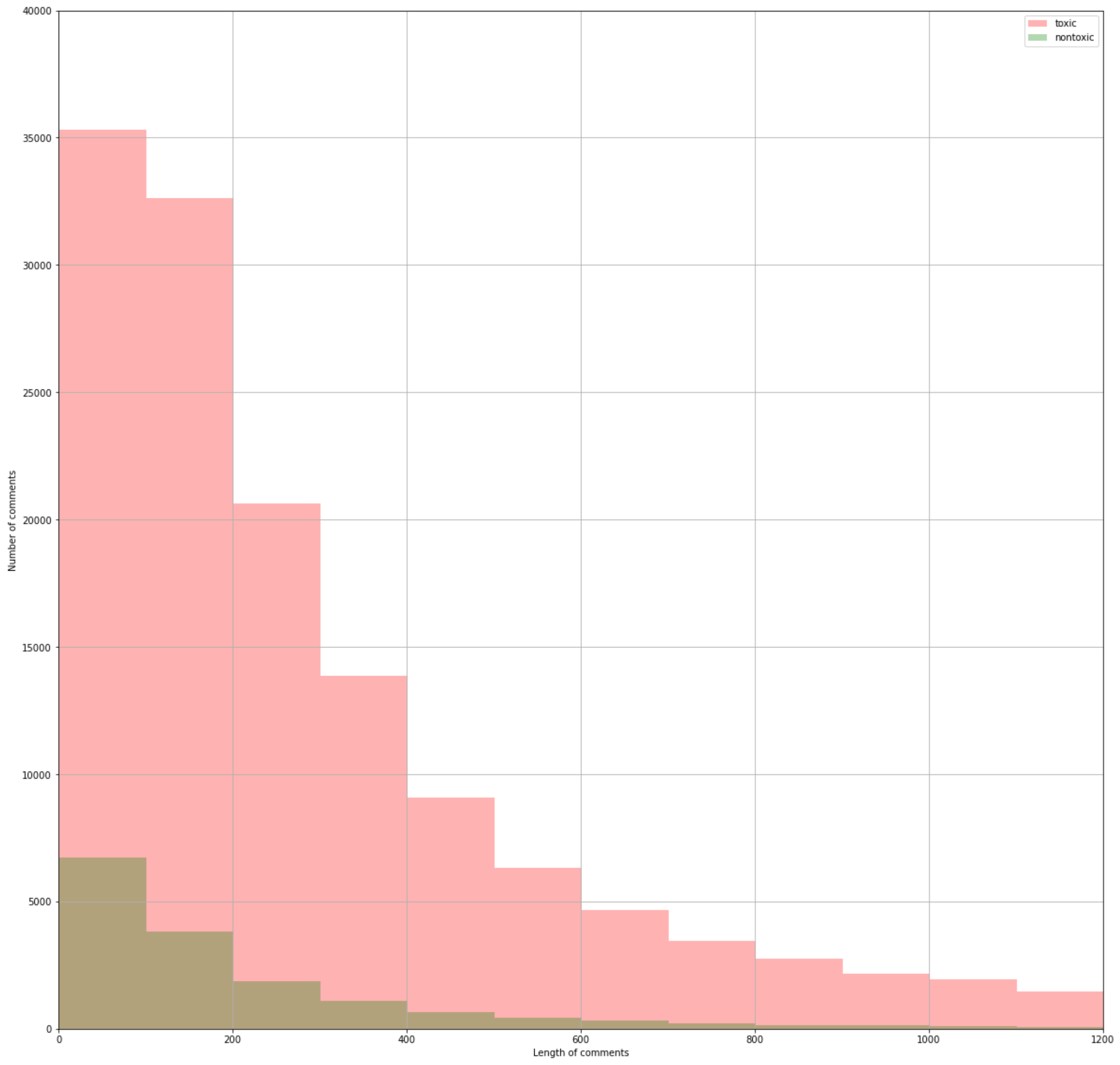


Chart 5. Histogram of Length of Comments Based on Category

The above histogram compares the lengths of comments labelled as "toxic" to the lengths of comments labelled as "non-toxic". The x-axis of the histogram represents the length of the comments, and the y-axis represents the number of comments. The histogram allows us to compare the distribution of comment lengths for toxic and non-toxic comments and identify any patterns or trends in the data.

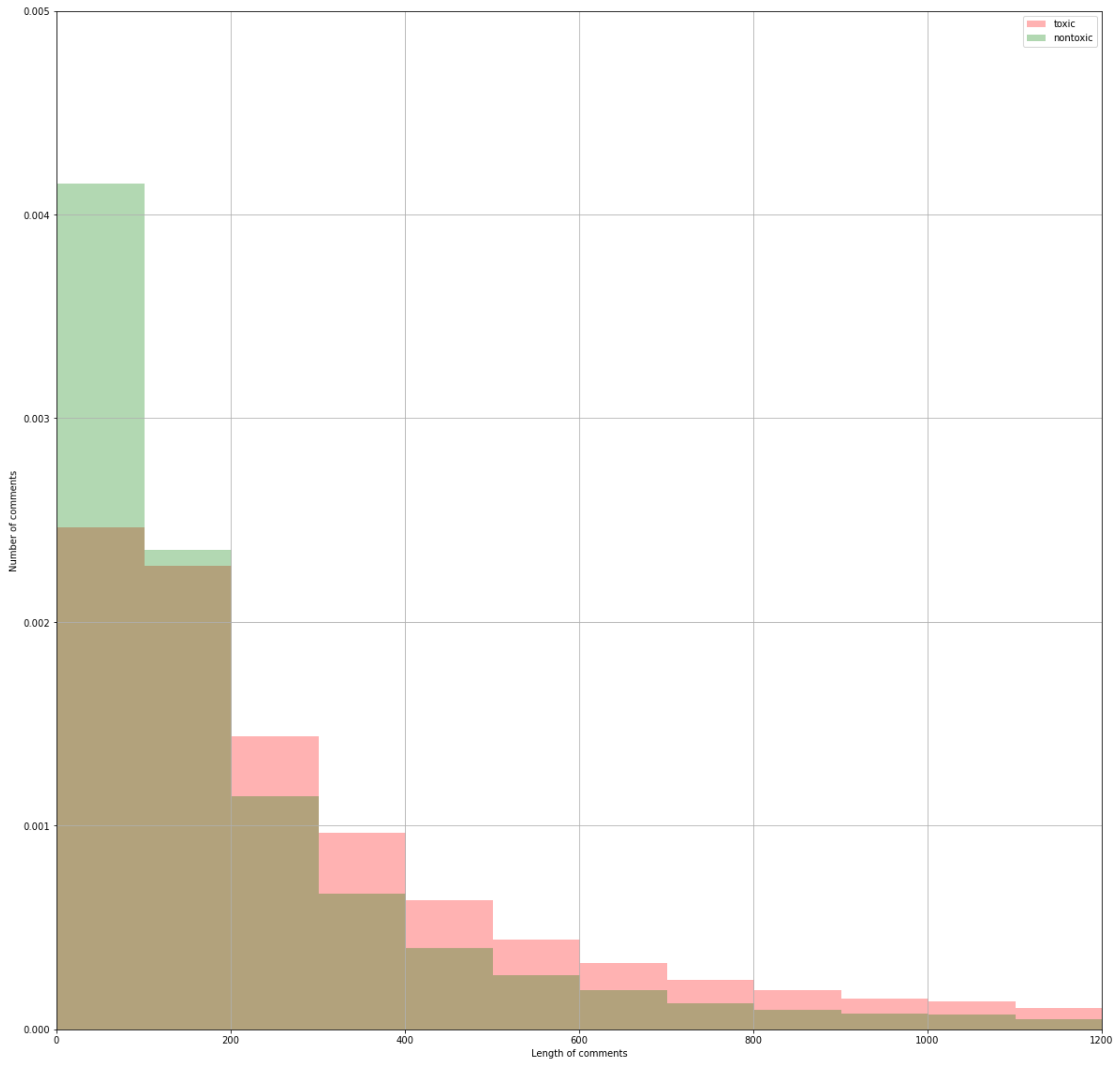


Chart 6. Normalized Histogram of Length of Comments Based on Category

The histogram compares the lengths of comments labelled as "toxic" to the lengths of comments labelled as "non-toxic". The x-axis of the histogram represents the length of the comments, and the y-axis represents the relative frequency of the comments. The histograms are normalized to make it easier to compare the relative frequencies of the comments.

By looking at the graphs and applying simple logic, we can think that comments of 200 words or less are more likely to be non-toxic. In this context, as a starting theory, let us create an algorithm to label comments longer than 200 words as toxic and shorter ones as nontoxic and look at important performance indicators.

## 6. TfidfVectorizer

TfidfVectorizer is a class in the scikit-learn library that can be used to transform a collection of raw documents into a matrix of TF-IDF features. TF-IDF stands for "Term Frequency-Inverse Document Frequency," and it is a statistical measure that reflects how important a word is to a document in a collection of documents.

The TfidfVectorizer class converts a collection of documents into a matrix of TF-IDF features by tokenizing the documents, extracting a vocabulary of words, and computing the TF-IDF score for each word in each document. The resulting matrix is a sparse matrix where each row represents a document and each column represents a word from the vocabulary. The value in each cell of the matrix is the TF-IDF score for the corresponding word in the document.

TF-IDF is calculated as follows:

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

IDF = log(Total number of documents / Number of documents with term t in it)

TF-IDF = TF * IDF

We used the TfidfVectorizer class to extract features for use in machine learning algorithm of text classification. It is often used in natural language processing tasks because it can help identify the most important words in a document and filter out less important words.

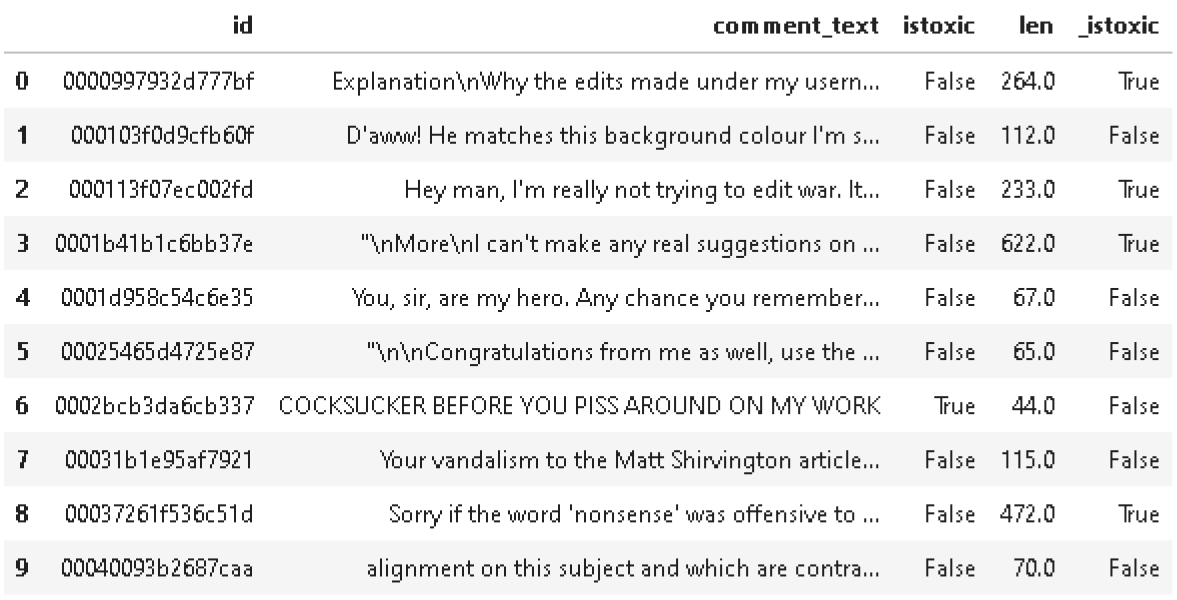
The vectorizer is initialized with the following parameters:

- ngram_range: The range of n-grams to be extracted from the documents. In this case, only unigrams (single words) are extracted.
- max_features: The maximum number of features (i.e., unique words) to be extracted from the documents. In this case, only the most frequent 10,000 words will be used.
- min_df: The minimum number of documents that a word must appear in to be included in the matrix. In this case, a word must appear in at least 2 documents to be included.
- max_df: The maximum proportion of documents that a word can appear in to be included in the matrix. In this case, a word can appear in at most 70% of the documents.

- strip_accents: A flag that specifies whether to remove accent marks from the words in the documents. In this case, accent marks are removed.
- analyzer: The type of tokens (i.e., words, characters, or subwords) to be extracted from the documents. In this case, only words are extracted.
- stop_words: The list of stop words (i.e., common words that are not useful for text analysis) to be removed from the documents. In this case, the default English stop words list is used.

## 7. Basic Model

To begin with, this model uses a basic principle. Those who exceed the 200 word threshold are labeled as 'toxic', and those who fall under labelled as 'non-toxic'



| | id | comment_text | istoxic | len | _istoxic |
|---|---|---|---|---|---|
| 0 | 0000997932d777bf | Explanation\nWhy the edits made under my usern... | False | 264.0 | True |
| 1 | 000103f0d9cfb60f | D'aww! He matches this background colour I'm s... | False | 112.0 | False |
| 2 | 000113f07ec002fd | Hey man, I'm really not trying to edit war. It... | False | 233.0 | True |
| 3 | 0001b41b1c6bb37e | "\nMore\nI can't make any real suggestions on ... | False | 622.0 | True |
| 4 | 0001d958c54c6e35 | You, sir, are my hero. Any chance you remember... | False | 67.0 | False |
| 5 | 00025465d4725e87 | "\n\nCongratulations from me as well, use the ... | False | 65.0 | False |
| 6 | 0002bcb3da6cb337 | COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK | True | 44.0 | False |
| 7 | 00031b1e95af7921 | Your vandalism to the Matt Shirvington article... | False | 115.0 | False |
| 8 | 00037261f536c51d | Sorry if the word 'nonsense' was offensive to ... | False | 472.0 | True |
| 9 | 00040093b2687caa | alignment on this subject and which are contra... | False | 70.0 | False |

**Table 2. Labelled Data by Basic Model**

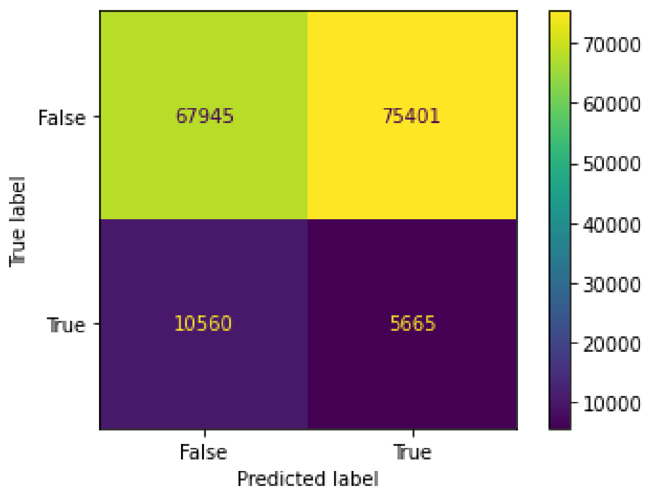


**Chart 7. Confusion Matrix of Basic Model**

**This model has scores as follows:**

```
Precision: 0.070
Recall: 0.349
Accuracy: 0.461
F1 Score: 0.116
```

## 8. Logistic Regression

Logistic regression is a statistical method that is used to model the relationship between a dependent variable and one or more independent variables. In scikit-learn, the LogisticRegression class is a machine learning model that can be used for binary classification tasks, where the target variable can take on only two values (e.g., 0 or 1, True or False).

To use the LogisticRegression class, we provided it with a set of labeled training data that includes both the independent variables and the dependent variable. We used the model to make predictions about the dependent variable for new, unseen data.

The LogisticRegression class uses a logistic function to model the relationship between the independent variables and the dependent variable. The logistic function is defined as:

f(x) = 1 / (1 + e^-x)

Where x is a linear combination of the independent variables. The output of the logistic function is a probability value between 0 and 1, which can be interpreted as the likelihood that the dependent variable takes on a particular value (e.g., 0 or 1).

Once the model is trained, we used it to make predictions about the dependent variable for new data using the predict method. Then we evaluated the model's performance using metrics.

**This model has scores as follows:**

```
Mean Absolute Error: 0.0756

Mean Squared Error:0.0333

Train Accuracy: 0.9605

Test Accuracy: 0.9552

Test F1 Score: 0.7381
```

## 9. Advanced Classifiers

- **K-Neighbors Classifier:** is a classifier implemented in scikit-learn that is based on the k-nearest neighbors algorithm. It works by finding the k nearest training samples to a given test sample and using the class labels of those training samples to predict the class label of the test sample. KNeighborsClassifier is a simple and effective classifier that can be used for both classification and regression tasks. It is sensitive to the choice of the value of k, which determines the number of nearest neighbors to consider when making predictions.

- **Random Forest Classifier**: is a classifier implemented in scikit-learn that is based on the random forest algorithm. It creates a collection of decision trees and uses them to make predictions about the class label of a given test sample. Random forests are often used for tasks involving large datasets or high-dimensional data because they can handle these types of data effectively. They are also robust to noise and outliers in the data, and they can handle missing values.

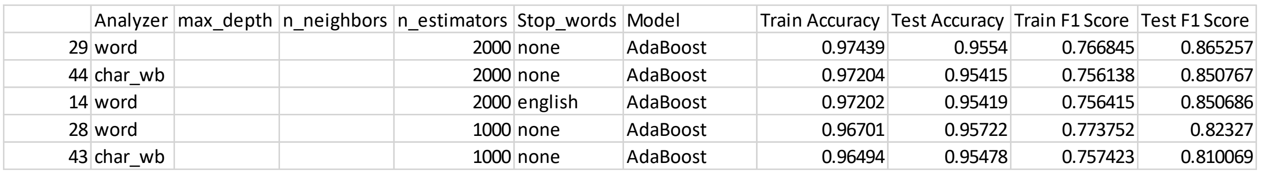
- **Ada Boost Classifier**: is a classifier implemented in scikit-learn that is based on the AdaBoost algorithm. It works by iteratively building a series of weak classifiers and combining them to form a strong classifier. AdaBoostClassifier is a powerful and popular classifier that has been used to achieve high accuracy on a wide range of tasks. It is sensitive to the choice of the base classifier, which determines the type of weak classifiers used in the ensemble. It is also sensitive to the choice of the learning rate, which controls the contribution of each weak classifier to the final strong classifier.

## 10. Hyper Parameter Table

| | Analyzer | max_depth | n_neighbors | n_estimators | Stop_words | Model | Train Accuracy | Test Accuracy | Train F1 Score | Test F1 Score |
|---|---|---|---|---|---|---|---|---|---|---|
| 29 | word | | | 2000 | none | AdaBoost | 0.97439 | 0.9554 | 0.766845 | 0.865257 |
| 44 | char_wb | | | 2000 | none | AdaBoost | 0.97204 | 0.95415 | 0.756138 | 0.850767 |
| 14 | word | | | 2000 | english | AdaBoost | 0.97202 | 0.95419 | 0.756415 | 0.850686 |
| 28 | word | | | 1000 | none | AdaBoost | 0.96701 | 0.95722 | 0.773752 | 0.82327 |
| 43 | char_wb | | | 1000 | none | AdaBoost | 0.96494 | 0.95478 | 0.757423 | 0.810069 |
| 13 | word | | | 1000 | english | AdaBoost | 0.96494 | 0.95478 | 0.757423 | 0.810069 |
| 27 | word | | | 500 | none | AdaBoost | 0.96237 | 0.95714 | 0.769645 | 0.795325 |
| 12 | word | | | 500 | english | AdaBoost | 0.96024 | 0.95434 | 0.750286 | 0.779614 |
| 42 | char_wb | | | 500 | none | AdaBoost | 0.96024 | 0.95434 | 0.750286 | 0.779614 |
| 39 | char_wb | 500 | | | none | RandomForest | 0.99244 | 0.95557 | 0.961242 | 0.759253 |
| 9 | word | 500 | | | english | RandomForest | 0.99231 | 0.95546 | 0.960518 | 0.757617 |
| 26 | word | | | 200 | none | AdaBoost | 0.95686 | 0.95457 | 0.747709 | 0.756653 |
| 38 | char_wb | 200 | | | none | RandomForest | 0.98321 | 0.95521 | 0.909538 | 0.750349 |
| 8 | word | 200 | | | english | RandomForest | 0.98315 | 0.95546 | 0.909161 | 0.749765 |
| 11 | word | | | 200 | english | AdaBoost | 0.95446 | 0.95141 | 0.722831 | 0.737472 |
| 41 | char_wb | | | 200 | none | AdaBoost | 0.95446 | 0.95141 | 0.722831 | 0.737472 |
| 25 | word | | | 100 | none | AdaBoost | 0.9523 | 0.95145 | 0.719595 | 0.72093 |
| 40 | char_wb | | | 100 | none | AdaBoost | 0.95015 | 0.9483 | 0.693498 | 0.700967 |
| 10 | word | | | 100 | english | AdaBoost | 0.95015 | 0.9483 | 0.693498 | 0.700967 |
| 24 | word | 500 | | | none | RandomForest | 0.99962 | 0.95095 | 0.998138 | 0.700128 |
| 23 | word | 200 | | | none | RandomForest | 0.9964 | 0.95081 | 0.981884 | 0.698734 |
| 7 | word | 100 | | | english | RandomForest | 0.9691 | 0.95001 | 0.81965 | 0.692377 |
| 37 | char_wb | 100 | | | none | RandomForest | 0.96785 | 0.94897 | 0.81093 | 0.684652 |
| 22 | word | 100 | | | none | RandomForest | 0.97955 | 0.94592 | 0.887499 | 0.651407 |
| 15 | word | | 1 | | none | KNN | 0.99962 | 0.91565 | 0.998095 | 0.425441 |
| 4 | word | | 5 | | english | KNN | 0.92764 | 0.90874 | 0.501173 | 0.363398 |
| 34 | char_wb | | 5 | | none | KNN | 0.92764 | 0.90874 | 0.501173 | 0.363398 |
| 3 | word | | 4 | | english | KNN | 0.92821 | 0.91379 | 0.470938 | 0.35445 |
| 33 | char_wb | | 4 | | none | KNN | 0.92821 | 0.91379 | 0.470938 | 0.35445 |
| 17 | word | | 3 | | none | KNN | 0.92986 | 0.91475 | 0.48841 | 0.339537 |
| 16 | word | | 2 | | none | KNN | 0.92985 | 0.91431 | 0.468962 | 0.318152 |
| 18 | word | | 4 | | none | KNN | 0.92444 | 0.88265 | 0.492727 | 0.290298 |
| 19 | word | | 5 | | none | KNN | 0.90766 | 0.84759 | 0.500629 | 0.27791 |
| 0 | word | | 1 | | english | KNN | 0.9967 | 0.41105 | 0.983836 | 0.220299 |
| 30 | char_wb | | 1 | | none | KNN | 0.9967 | 0.41105 | 0.983836 | 0.220299 |
| 2 | word | | 3 | | english | KNN | 0.43415 | 0.33005 | 0.24534 | 0.211564 |
| 32 | char_wb | | 3 | | none | KNN | 0.43415 | 0.33005 | 0.24534 | 0.211564 |
| 31 | char_wb | | 2 | | none | KNN | 0.97924 | 0.4158 | 0.888343 | 0.211375 |
| 1 | word | | 2 | | english | KNN | 0.97924 | 0.4158 | 0.888343 | 0.211375 |
| 6 | word | 10 | | | english | RandomForest | 0.89961 | 0.89754 | 0.01372 | 0.011288 |
| 36 | char_wb | 10 | | | none | RandomForest | 0.89951 | 0.8975 | 0.011797 | 0.010486 |
| 21 | word | 10 | | | none | RandomForest | 0.8995 | 0.89746 | 0.011622 | 0.009683 |
| 35 | char_wb | 1 | | | none | RandomForest | 0.89891 | 0.89695 | 0 | 0 |
| 5 | word | 1 | | | english | RandomForest | 0.89891 | 0.89695 | 0 | 0 |
| 20 | word | 1 | | | none | RandomForest | 0.89891 | 0.89695 | 0 | 0 |

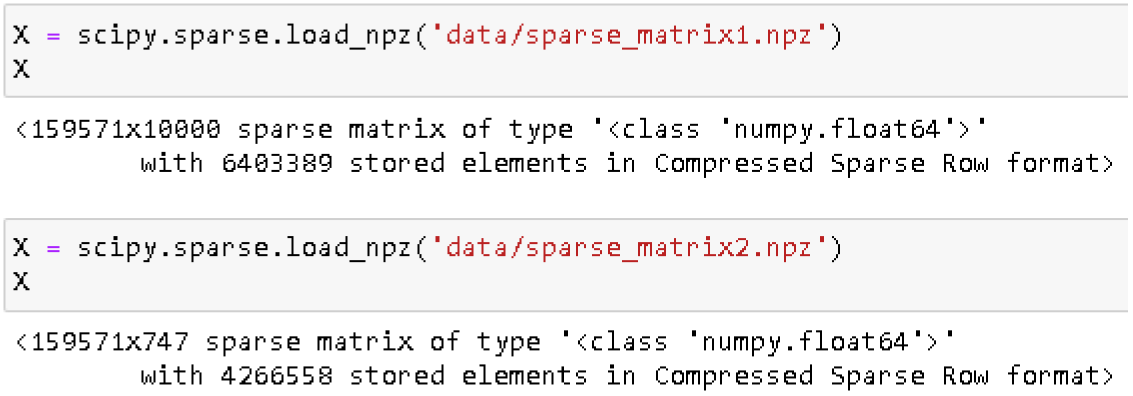
## 11. Conclusion

The study aimed to classify toxic and non-toxic comments using the Kaggle Toxic Comment Classification Challenge Twitter Dataset. The dataset included 159571 comments in the train set and 153164 comments in the test set, and presented both a multi-class and a multi-label classification task. The goal was to build a natural language processing model that could predict the probabilities of different types of toxicity for each comment.



| | Analyzer | max_depth | n_neighbors | n_estimators | Stop_words | Model | Train Accuracy | Test Accuracy | Train F1 Score | Test F1 Score |
|---|---|---|---|---|---|---|---|---|---|---|
| 29 | word | | | 2000 | none | AdaBoost | 0.97439 | 0.9554 | 0.766845 | 0.865257 |
| 44 | char_wb | | | 2000 | none | AdaBoost | 0.97204 | 0.95415 | 0.756138 | 0.850767 |
| 14 | word | | | 2000 | english | AdaBoost | 0.97202 | 0.95419 | 0.756415 | 0.850686 |
| 28 | word | | | 1000 | none | AdaBoost | 0.96701 | 0.95722 | 0.773752 | 0.82327 |
| 43 | char_wb | | | 1000 | none | AdaBoost | 0.96494 | 0.95478 | 0.757423 | 0.810069 |

To tackle the challenges of natural language processing, the study proposed an ensemble of state-of-the-art classifiers and analyzed false negatives and false positives to gain insights into the open challenges that all approaches shared. The results of the study showed that the best performing models were all from the AdaBoostClassifier, with the best model achieving a Test F1 score of 0.865257 using the parameters analyzer='word' and stop_words='none'. The model also had a Train accuracy of 0.97439 and a Test accuracy of 0.9554.



```
X = scipy.sparse.load_npz('data/sparse_matrix1.npz')
X
```

```
<159571x10000 sparse matrix of type '<class 'numpy.float64'>'
	with 6403389 stored elements in Compressed Sparse Row format>
```

```
X = scipy.sparse.load_npz('data/sparse_matrix2.npz')
X
```

```
<159571x747 sparse matrix of type '<class 'numpy.float64'>'
	with 4266558 stored elements in Compressed Sparse Row format>
```

Overall, the AdaBoostClassifier performs better than the RandomForest and KNN algorithms in terms of Test F1 Score. When we select the 'char_wb' parameter for the analyzer, the number of features decreases, as shown in the matrix dimensions above. This can negatively impact model performance, and in this case, it slightly affected the performance of our model. As we can see from the table, the larger the value of the n_estimators parameter, the better the accuracy and Test F1 Score. The stop_words parameter removes a list of stop words from the documents. In this case, selecting the 'English' stop words had a positive impact on performance, while using 'None' worked well. This suggests that articles are important when trying to predict toxic comments.

To put it in a nutshell, the results of the study demonstrated the effectiveness of the AdaBoostClassifier in classifying toxic and non-toxic comments. The high Test F1 score and accuracies achieved by the model suggest that it is a strong performer for this task. However, the study also highlighted the challenges of working with imbalanced data and the importance of carefully choosing and tuning the hyperparameters of the classifiers to achieve the best possible performance.

## 12. Future Work

One potential direction for future work is to explore the impact of the ngram_range parameter on the performance of the classifiers. Ngrams are contiguous sequences of words, and the ngram_range parameter specifies the minimum and maximum length of the ngrams to consider. Changing the ngram_range parameter could potentially improve the performance of the classifiers by allowing them to capture more complex patterns of word relationships within the comments.

Another potential direction for future work is to explore more advanced techniques for addressing the challenge of imbalanced data. For example, one approach that has been shown to be effective in some cases is the use of the synthetic minority oversampling technique (SMOTE), which generates synthetic samples of the minority class to balance the dataset. Another possibility is to use class weighting or sample weighting to assign higher weights to samples from the minority class during training.

Another area of potential improvement is the use of more sophisticated natural language processing techniques. For example, the use of advanced words embedding models, such as BERT or GPT-2, could potentially improve the performance of the classifiers by providing more accurate representations of the meaning of words in the dataset. Additionally, the use of more advanced neural network architectures, such as transformers or attention mechanisms, could also improve the performance of the classifiers by better capturing the relationships between words and the context in which they appear.

Finally, it would be interesting to investigate the performance of the classifiers on other types of toxic comment datasets, such as those from different social media platforms or those in different languages. This could help to assess the generalizability of the classifiers and identify any language- or platform-specific challenges that need to be addressed.