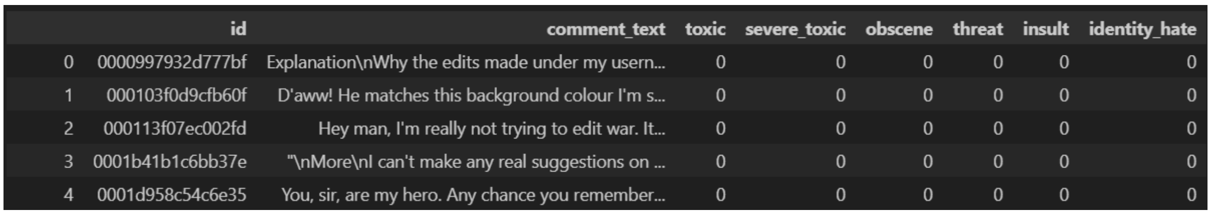
Note: The page limit is exceeding due to the images inserted for a detailed analysis, the written content is within the limit.

Introduction:

In an age dominated by digital communication, the vast landscape of user-generated content has given rise to both unprecedented opportunities for interaction and the pressing challenge of managing harmful and toxic content. The need for a safe and respectful online environment has never been more crucial. Content moderation stands as the frontline defence in maintaining the quality and safety of digital platforms. In response to this imperative, the integration of machine learning emerges as a promising solution to automate the detection and flagging of potentially harmful content. This project embarks on a journey to construct a robust model capable of navigating the complex terrain of user-generated content, contributing to the creation of safer digital spaces.

About the Data:



	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

The provided dataset consists of many Wikipedia comments which have been labelled by human raters for toxic behaviour. The types of toxicity are toxic, severe toxic, obscene, threat, insult, identity hate, and through this project we aim to implement different machine learning models that accurately predict the probability of each type of toxicity for each comment, and finally determine the most suitable model for this task.

Data-Preprocessing:

After importing the dataset using Pandas, we removed stop words from the comments because they add noise to the data and reduce model efficiency while also increasing computation time. This process also boosts feature relevance in text classification, emphasizing informative words and aiding model generalization for improved performance on unseen data. We used the NLTK (Natural Language Toolkit) library for this task. Then with the help of regex we removed URL's, non-alphanumeric characters, and extra whitespaces to make sure that the text data is cleaned and standardized, ensuring a more focused and efficient analysis in subsequent modeling tasks.

Models:

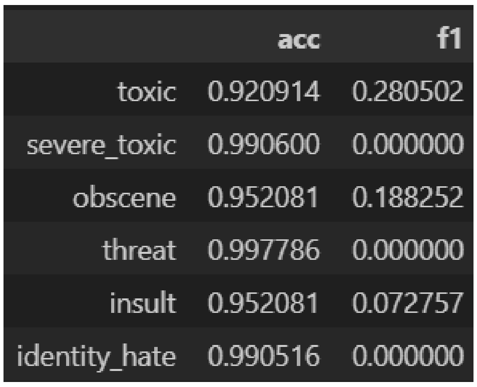
Before training any models, it is important to convert text data into features that can be understood by a machine learning model. We used different techniques for our different models.

Classifiers:

For our classifiers we used TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, a numerical statistic that reflects how important a word is to a document relative to a collection of documents. TF-IDF is calculated by multiplying the term frequency (how often a word appears in a document) by the inverse document frequency (how unique or rare a word is across documents). It captures the importance of words in distinguishing toxic and non-toxic comments, providing a quantitative representation of the textual information for model training.

Multinomial Naïve Bayes Classifier:

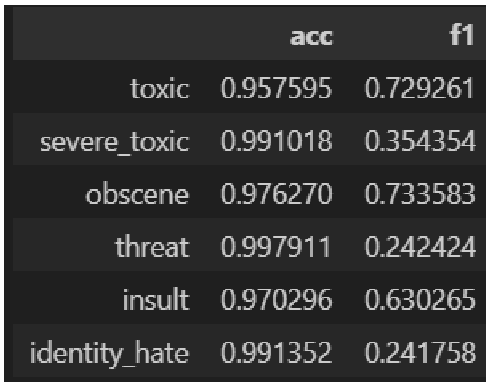
We utilize the OneVsRestClassifier to extend the binary Naïve Bayes Classifier for multi-label classification, allowing it to handle various toxicity categories. To assess the model's performance, accuracy and F1 scores are calculated. F1 is especially emphasized due to class distribution differences, as it effectively balances precision and recall. This metric provides a comprehensive evaluation of the model's capability to identify instances across different classes, ensuring robustness in imbalanced scenarios. The figure shows our accuracy and f1 scores. The F1 scores are low, so some changes to our model are required.



	acc	f1
toxic	0.920914	0.280502
severe_toxic	0.990600	0.000000
obscene	0.952081	0.188252
threat	0.997786	0.000000
insult	0.952081	0.072757
identity_hate	0.990516	0.000000

Logistic Regression Classification:

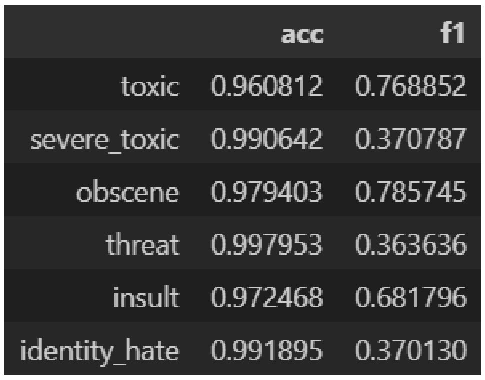
Next, we employ a Logistic Regression classifier within a One-vs-Rest framework. The model is trained and evaluated on toxicity categories. Again, we focus on the F1-score. This model gave better results than the Naïve Bayes Classifier, but it still demonstrates challenges in handling the dataset's complexities. The figure shows the improvements in F1 scores compared to the initial model.



	acc	f1
toxic	0.957595	0.729261
severe_toxic	0.991018	0.354354
obscene	0.976270	0.733583
threat	0.997911	0.242424
insult	0.970296	0.630265
identity_hate	0.991352	0.241758

Linear Support Vector Classifier:

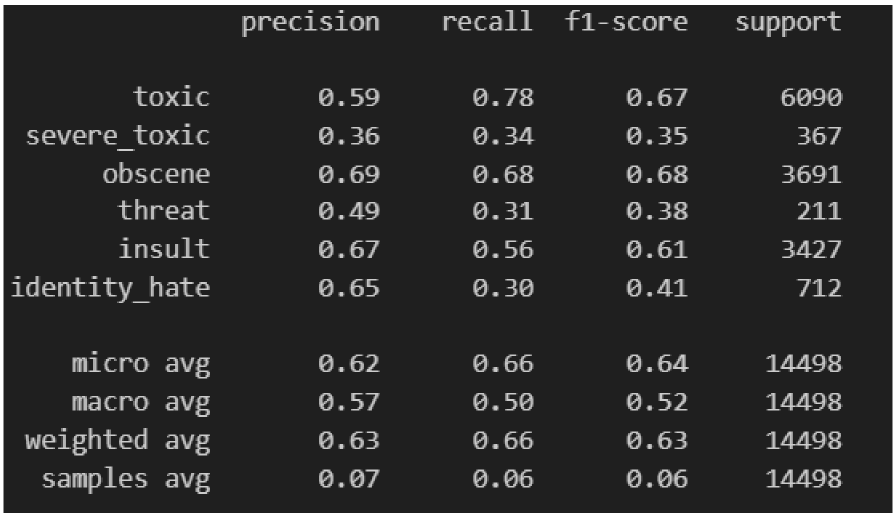
Finally, we establish a machine learning pipeline for toxicity detection using a Linear Support Vector Classifier (LinearSVC) within a One-vs-Rest framework. The model is trained and evaluated on toxicity categories, and it demonstrates enhanced performance compared to previous models. This improvement suggests that the LinearSVC offers a more promising approach for addressing the intricacies within the dataset, showcasing potential advancements in toxicity detection. The figures show



	acc	f1
toxic	0.960812	0.768852
severe_toxic	0.990642	0.370787
obscene	0.979403	0.785745
threat	0.997953	0.363636
insult	0.972468	0.681796
identity_hate	0.991895	0.370130

the results. It is clearly performing better than the other models but again it's overall performance for evaluating classes like "sever_toxic", "threat", and "identity_hate" is not satisfactory.

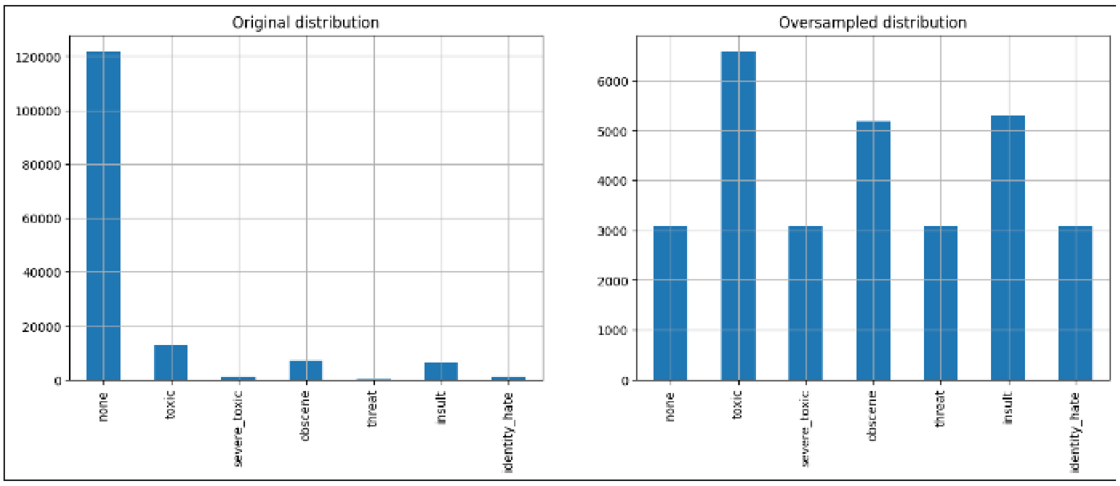
Running on the test data with SVC as it gave us the best results, yielded the following:



```
               precision    recall  f1-score   support

        toxic       0.59      0.78      0.67      6090
 severe_toxic       0.36      0.34      0.35       367
      obscene       0.69      0.68      0.68      3691
       threat       0.49      0.31      0.38       211
       insult       0.67      0.56      0.61      3427
identity_hate       0.65      0.30      0.41       712

    micro avg       0.62      0.66      0.64     14498
    macro avg       0.57      0.50      0.52     14498
 weighted avg       0.63      0.66      0.63     14498
  samples avg       0.07      0.06      0.06     14498
```

Sampling:

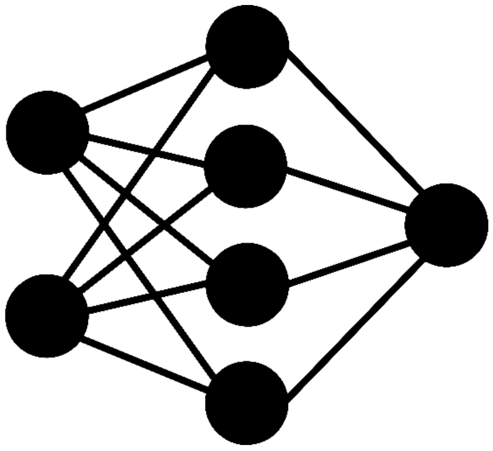
After testing out the performance of our classifiers, we decided to solve the issue of class imbalance. Upon conducting EDA, initial exploration highlighted that a significant portion of comments lacked classification. These unclassified comments were filtered out, revealing an imbalanced distribution of toxicity types. Certain toxicities were prevalent, while others were underrepresented. Recognizing this imbalance, we employed `LeastSampleClassSampler` during model training to address the uneven class distribution, ensuring a more balanced and robust toxicity detection model. (The graphs below show the class distribution before and after using the sampling technique)



Data Preprocessing:

Instead of using TFIDF vectorization as before we used the Embed4all feature extractor from the gpt4all library. This extractor generates contextualized embeddings for words in sentences, influenced not only by the word itself but also by its surrounding context in the text. The output is a numerical representation for the entire input text, ideal for training machine learning models.

Custom Neural Network:



Model Architecture:

The architecture comprises three linear layers with ReLU activation, specifically designed for toxicity detection. The input size is 384, leading to intermediate layers of 768 and 384 units. The output layer corresponds to the number of toxicity categories, tailoring the model for this specific task. The training loop runs for 10 epochs. The AdamW optimizer is employed with a learning rate of 0.001.

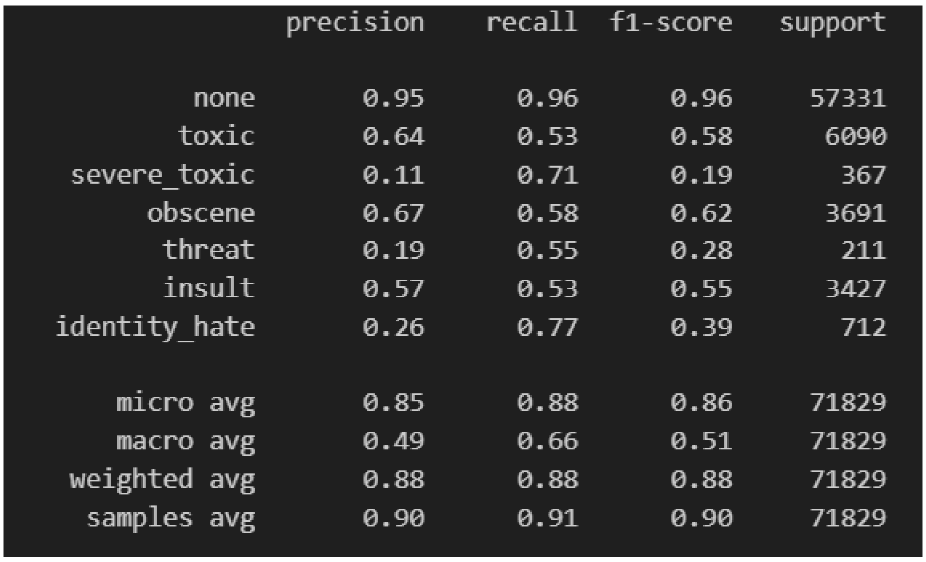
Batch Processing:

For each batch, optimizer parameters are zeroed to prevent accumulation of gradients from previous batches. The forward pass computes logits (raw output scores), and the binary cross-entropy loss is calculated using the `BCEWithLogitsLoss` function. This loss encompasses a built-in activation and is used because we wanted to measure loss over different classes.

Backward Pass and Weight Update:

The backward pass is executed to compute gradients, and the optimizer updates the model weights accordingly. This step refines the model's understanding based on the computed loss, facilitating learning.

The following is the performance of our model on the test data (Clearly a lot of improvement from the classifiers):



```
               precision    recall  f1-score   support

         none       0.95      0.96      0.96     57331
        toxic       0.64      0.53      0.58      6090
 severe_toxic       0.11      0.71      0.19       367
      obscene       0.67      0.58      0.62      3691
       threat       0.19      0.55      0.28       211
       insult       0.57      0.53      0.55      3427
identity_hate       0.26      0.77      0.39       712

    micro avg       0.85      0.88      0.86     71829
    macro avg       0.49      0.66      0.51     71829
 weighted avg       0.88      0.88      0.88     71829
  samples avg       0.90      0.91      0.90     71829
```

Pretrained Encoder-Transformers:

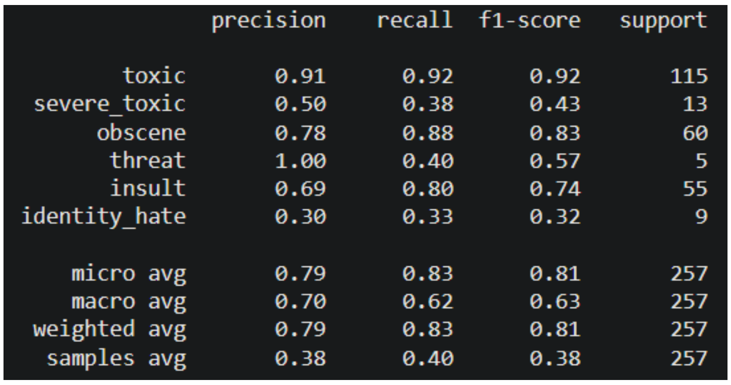
Model Architecture:

The foundation of this model is based on Bidirectional Encoder Representations from Transformers (BERT) which has been then molded to be used for multi-label sequence classification. It is for this very reason that a function 'BertForMultiLabelSequenceClassification' was defined using PyTorch as the primary inspiration. The forward pass for this model takes input IDs, attention masks, and token type IDs as inputs and passes them through the BERT model to return logits. This model is then trained for 5 epochs using AdamW as the optimizer and a learning rate of 1e-05 to fine tune the model.

Optimizer and Learning Rate:

As it has been highlighted earlier, AdamW has been used as the optimizer in this model. The primary drivers for this choice is to tackle the issue of weight decay as has been reported with the original Adam optimizer. AdamW optimizer has been used in tandem with a low learning rate of 1e-05 which assists in fine tuning the model and providing stability and generalization. The application of this model's fine tuning can be visualized using tqdm which shows the decreasing loss across the 5 epochs.

Validation:

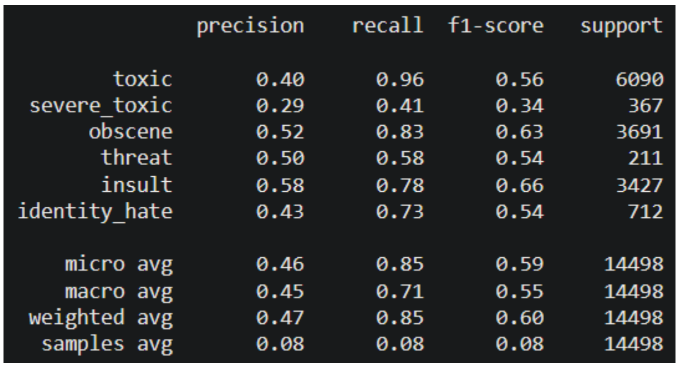


```
               precision    recall  f1-score   support

        toxic       0.91      0.92      0.92       115
 severe_toxic       0.50      0.38      0.43        13
      obscene       0.78      0.88      0.83        60
       threat       1.00      0.40      0.57         5
       insult       0.69      0.80      0.74        55
identity_hate       0.30      0.33      0.32         9

    micro avg       0.79      0.83      0.81       257
    macro avg       0.70      0.62      0.63       257
 weighted avg       0.79      0.83      0.81       257
  samples avg       0.38      0.40      0.38       257
```

The model was then validated on a portion of the training data to analyze its performance and provided with the following results:

These results depict a rather decent performance for the said model where it has demonstrated good performance for some classes (toxic and obscene) but tends to struggle with the minority classes.

Testing:



```
               precision    recall  f1-score   support

        toxic       0.40      0.96      0.56      6090
 severe_toxic       0.29      0.41      0.34       367
      obscene       0.52      0.83      0.63      3691
       threat       0.50      0.58      0.54       211
       insult       0.58      0.78      0.66      3427
identity_hate       0.43      0.73      0.54       712

    micro avg       0.46      0.85      0.59     14498
    macro avg       0.45      0.71      0.55     14498
 weighted avg       0.47      0.85      0.60     14498
  samples avg       0.08      0.08      0.08     14498
```

The same model was then tested on a designated testing dataset to reevaluate its performance and returned the following results:

These results suggest an effective stance in model's effectiveness in identifying toxic comments but still faces challenge in identifying identity hate and severe toxicity with similar trends and patterns in the functioning of the model. This suggests that while the model is relatively efficient in identifying toxic, insult and obscene comments, it struggles in minority and individual instances.

These results are similar to the ones obtained for the Custom Neural Networks with minor fluctuations; the neural networks returned a higher improved precision but decreased recall for the "toxic" class compared to the current testing data for transformers. This suggests that the transformer while has a lower precision for predicting toxic comments, the instances where it predicts positive are correct.

Discussion:

A general overview of the results reveal that traditional classification models of Multinomial Naïve Bayes and Logistic Regression recorded a suboptimal performance which pushed for the need for a rather nuanced approach. It was for this that a Linear Support Vector was used for classification and while it showed improvement, the results from Custom Neural Networks and Pre-Trained Encoded Transformers gave the best results but still struggled with minority data.

The issue of addressing class imbalance through 'LeastSampleClassSampler' enhanced the robustness of the model and the use of embedding improved the model's understanding of the context and thus improved the results.

However, there is still room for improvement where the use of ensemble methods, fine-tuning hyperparameters, experimenting with advanced architectures and incorporating transfer learning can be utilized.