Toxic Comment Classification

Stakeholder:

The stakeholder for this project is an online platform that hosts user-generated content, such as comments and discussions. The platform aims to create a safer and more inclusive environment by automatically identifying and moderating toxic comments.

Problem Statement:

The primary problem is the presence of toxic comments in user-generated content, which can contribute to a hostile online environment. The stakeholder aims to implement a machine learning model to automatically detect and filter out toxic comments, enhancing user experience and platform safety.

Dataset Source:

The dataset used for training and evaluation is the "Toxic Comment Classification Challenge" dataset from Kaggle. The dataset contains many comments labeled with various toxic categories.

This is how the dataset looks like. There are 6 labels in the dataset and one input feature and id for each comment.

	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
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Project Description:

I conducted a binary classification of the comments, deviating from the original dataset's multiclass multioutput approach. I will exclusively utilize the training set provided by Kaggle and proceed with the customary split into training and testing sets. The dataset comprises 143,346 non-toxic comments and 16,225 toxic comments. A comment is deemed toxic if it falls into at least one of the toxicity categories outlined in the original dataset. The initial toxicity labels were manually assigned by humans, following the dataset description available on Kaggle. The

dataset includes raw comments, extracted in their original form as posted by users on Wikipedia talk pages.

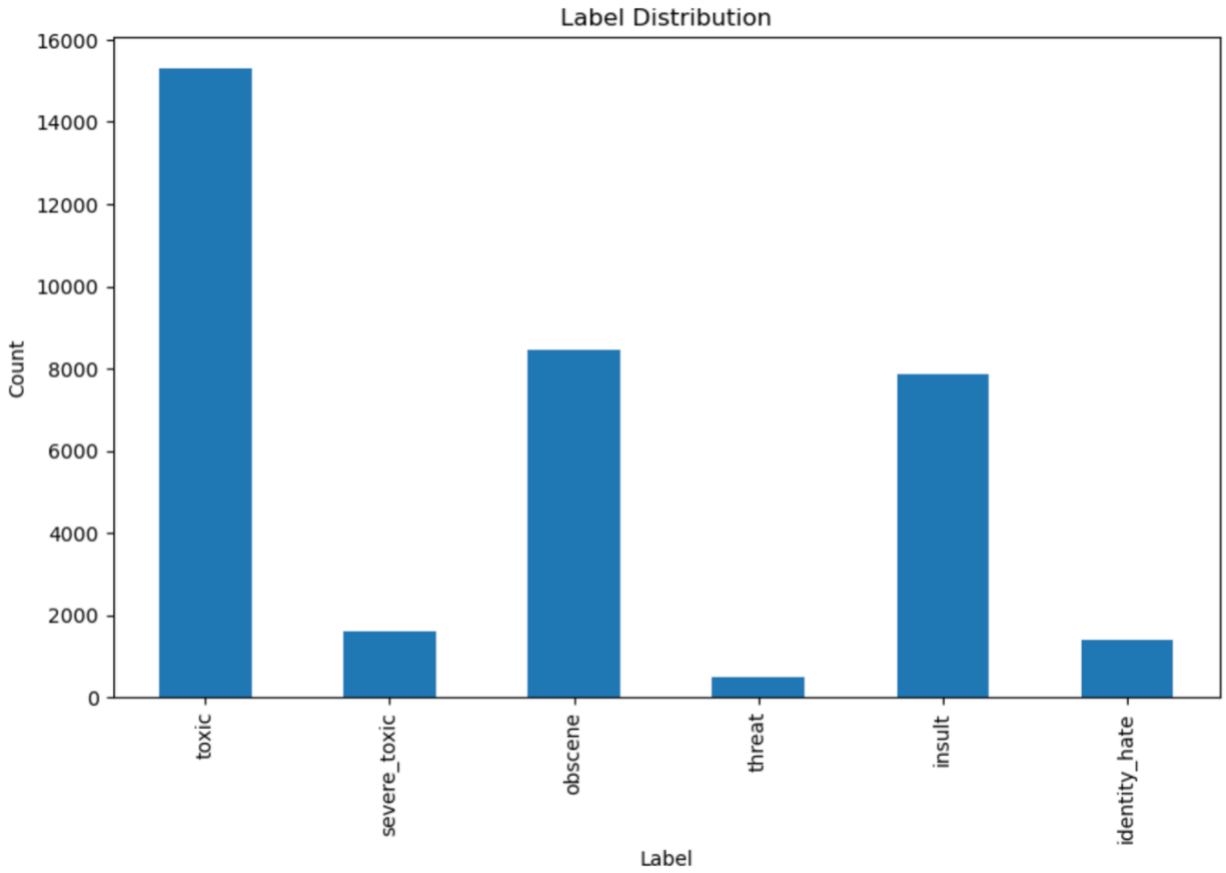
Models Explored:

1. Linear SVC
2. Logistic Regression
3. K-nearest Neighbors
4. Decision Tree Classifier
5. Random Forest
6. Naïve Bayes

Data Cleaning:

1. I merged the labels ["toxic", "severe_toxic", "obscene", "threat", "insult", "identity_hate"] into a single label, referred to as "toxic," while categorizing the rest as non-toxic.

2. I addressed missing values and conducted an analysis. We can see the label distribution below.



3. Additionally, I cleaned the comments using a regex filter to eliminate punctuation, unnecessary spaces, and other elements.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	explanation why the edits made under my userna...	0	0	0	0	0	0
1	000103f0d9cfb60f	d aww he matches this background colour i am s...	0	0	0	0	0	0
2	000113f07ec002fd	hey man i am really not trying to edit war it ...	0	0	0	0	0	0
3	0001b41b1c6bb37e	more i cannot make any real suggestions on imp...	0	0	0	0	0	0
4	0001d958c54c6e35	you sir are my hero any chance you remember wh...	0	0	0	0	0	0
5	00025465d4725e87	congratulations from me as well use the tools ...	0	0	0	0	0	0
6	0002bcb3da6cb337	cocksucker before you piss around on my work	1	1	1	0	1	0
7	00031b1e95af7921	your vandalism to the matt shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	sorry if the word nonsense was offensive to yo...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Data Preprocessing:

These are the following steps I have done for preprocessing the comments.

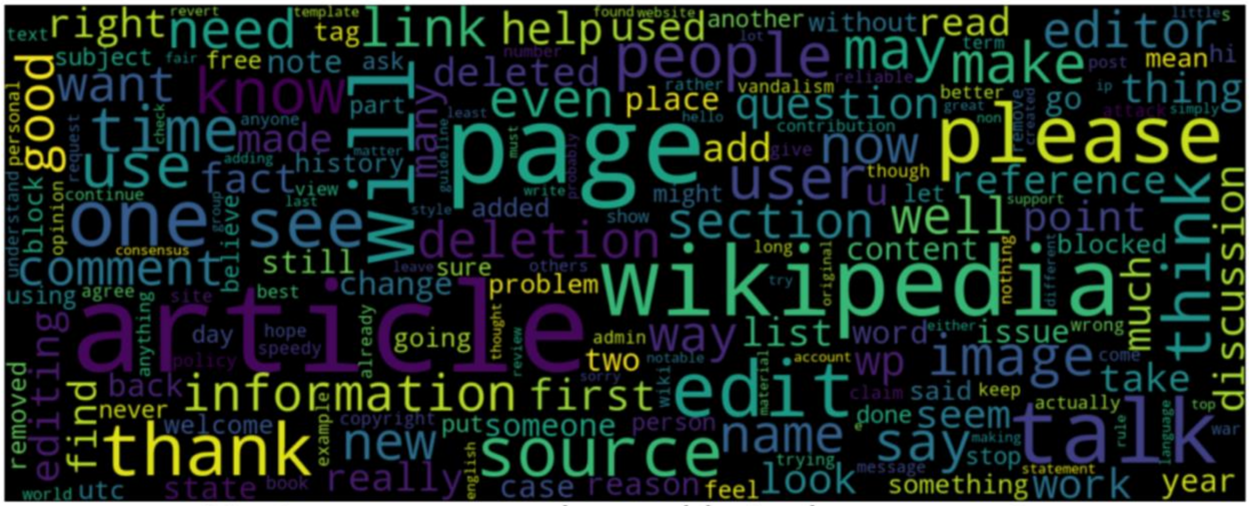
1. Eliminate stop words, which include articles and pronouns. I have used “spacy” module to remove the stop words.
2. Breaking down the text into tokens.
3. Apply stemming to the tokens. Stemming is mainly used to remove inflections such as "s" in "dogs" or "ed" in "expected."

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	label	tokenized_comments	stemmed_comments
0	0000997932d777bf	explanation why the edits made under my userna...	0	0	0	0	0	0	0	[explanation, edits, username, hardcore, metal...	[explan, edit, usernam, hardcor, metallica, fa...
1	000103f0d9cfb60f	d aww he matches this background colour i am s...	0	0	0	0	0	0	0	[d, aww, matches, background, colour, seemingl...	[d, aww, match, background, colour, seem, stuc...
2	000113f07ec002fd	hey man i am really not trying to edit war it ...	0	0	0	0	0	0	0	[hey, man, trying, edit, war, guy, constantly,...	[hey, man, tri, edit, war, guy, constant, remo...
3	0001b41b1c6bb37e	more i cannot make any real suggestions on imp...	0	0	0	0	0	0	0	[real, suggestions, improvement, wondered, sec...	[real, suggest, improv, wonder, section, stati...
4	0001d958c54c6e35	you sir are my hero any chance you remember wh...	0	0	0	0	0	0	0	[sir, hero, chance, remember, page]	[sir, hero, chanc, rememb, page]

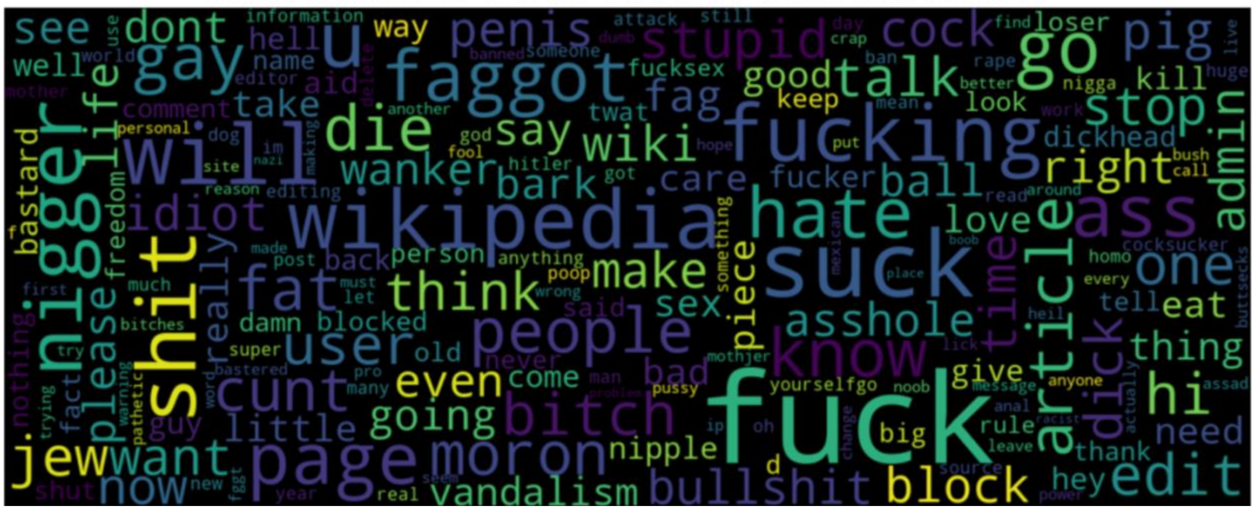
Data Visualization:

I have used “wordcloud” library to display the most common words in both categories.

Most common words used in non-toxic comments



Most common words used in toxic comments



Feature Selection/Engineering, Text Embedding:

I have tried three different embeddings – TF-IDF, Word2Vec and Doc2Vec, to understand which is helpful.

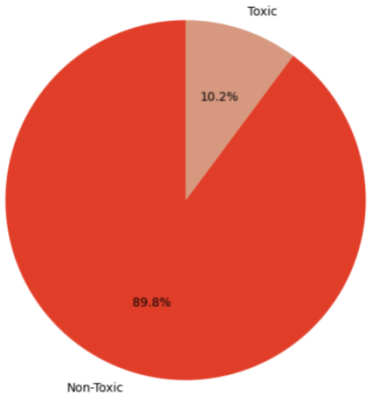
1. **TF-IDF:** Its assigns weights to words based on frequency, with common words receiving lower weights. Representing words involves using an array of scores, but this "Bag of Words" approach has drawbacks like losing word order and context.
2. **Word2Vec:** Word2Vec, a 2013 NLP technique, employs a neural network to learn word representation from text. It aims to create a semantic understanding where similar words have close vector representations. In our project, comments are represented by averaging word vectors. This simple yet effective approach is used to form a continuous bag-of-words (CBOW) representation, predicting the current word from a context window without considering the order of context words. I have built a function that looks up the model for vector representations of tokens in a document, averaging them to represent the entire comment. In essence, a comment is the average of its word representations.

3. **Doc2Vec:** Doc2Vec, unlike Word2Vec, directly represents a document as a vector by incorporating a Paragraph Vector during training. Using the Distributed Bag-Of-Words (DBOW) model in this project, we predict context words from a target word.

After preparing and setting parameters, the trained model for each embedding is saved to disk to avoid retraining.

Data Imbalance:

The dataset is heavily imbalanced. The ratio of toxic vs non-toxic comments is almost 1:9



To tackle this 9:1 imbalance, I used the SMOTE algorithm from the “imblearn” library. This technique creates synthetic data points through vector operations, applied post-vector encoding. With default settings, SMOTE balances the dataset to a 1:1 ratio, enhancing the final classification model's performance, as observed in tests. This operation really boosted the performance of the final classification model for this project. This operation must be done after the vector encoding phase.

Feature Normalization:

In this step, I used the normalizer from “sklearn” to normalize the data. The normalizer individually scales each sample to a unit norm, ensuring that its norm (l1, l2, or inf) equals one. This scaling operation is crucial for text classification. I applied normalization for each text encoding method used, and the resulting normalized vectors serve as input for the classification models. Through testing, I have observed that this approach enhanced the final model performance.

Evaluation Metrics:

The metrics I am considering are:

1. Accuracy
2. F1 score
3. Recall

4. Precision
5. Specificity
6. True Positive
7. True Negative
8. False Positive
9. False Negative
10. Size of the Test Set

The choice of the important metric to maximize (precision vs recall) depends on the domain and the application itself. There is no right or wrong metric here. In this context, toxic comments classification, we can decide to choose False negatives over False positives or False positives over False Negatives based on the application we are using this for. For example, if we use this application for YT Kids, we need to maximize recall as False Negative will have a higher misclassification cost. In some other context, we would need to maximize precision to avoid the reporting of non-toxic comments.

For this project I am planning on choosing to **maximize precision**, i.e., cost of False Positives should be greater than cost of False Negatives. I will be doing threshold tuning to figure out at which threshold this is possible.

Model Selection and Hyperparameter Tuning:

I am attaching how all the models performed in each embedding and their evaluation metrics.

TF-IDF:

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	Linear SVC	0.828306	0.823587	0.801557	0.846861	0.855054	28723	30640	5194	7111	71668
0	K-Nearest Neighbours(3)	0.737861	0.699182	0.609282	0.820204	0.866440	21833	31048	4786	14001	71668
0	Logistic Regression	0.828040	0.823746	0.803678	0.844843	0.852403	28799	30545	5289	7035	71668
0	Naive Bayes	0.811743	0.795978	0.734470	0.868729	0.889016	26319	31857	3977	9515	71668
0	Decision Tree Classifier	0.794874	0.770006	0.686750	0.876233	0.902997	24609	32358	3476	11225	71668
0	Random Forest Classifier	0.816948	0.791864	0.696434	0.917601	0.937462	24956	33593	2241	10878	71668

Word2Vec:

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	Linear SVC	0.923383	0.922172	0.907825	0.936980	0.938941	32531	33646	2188	3303	71668
0	K-Nearest Neighbours(3)	0.856226	0.857463	0.864905	0.850148	0.847547	30993	30371	5463	4841	71668
0	Logistic Regression	0.923160	0.921970	0.907909	0.936473	0.938410	32534	33627	2207	3300	71668
0	Naive Bayes	0.842035	0.834843	0.798487	0.874668	0.885584	28613	31734	4100	7221	71668
0	Decision Tree Classifier	0.825961	0.812984	0.756572	0.878487	0.895351	27111	32084	3750	8723	71668
0	Random Forest Classifier	0.876346	0.864408	0.788302	0.956781	0.964391	28248	34558	1276	7586	71668

Doc2Vec:

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	Linear SVC	0.923885	0.922792	0.909723	0.936242	0.938048	32599	33614	2220	3235	71668
0	K-Nearest Neighbours(3)	0.877658	0.880411	0.900681	0.861034	0.854635	32275	30625	5209	3559	71668
0	Logistic Regression	0.923997	0.922931	0.910169	0.936056	0.937824	32615	33606	2228	3219	71668
0	Naive Bayes	0.884900	0.879130	0.837166	0.925524	0.932634	29999	33420	2414	5835	71668
0	Decision Tree Classifier	0.802073	0.785489	0.724759	0.857327	0.879388	25971	31512	4322	9863	71668
0	Random Forest Classifier	0.852459	0.830425	0.722526	0.976208	0.982391	25891	35203	631	9943	71668

Observations (Doc2Vec):

- **Linear SVC and Logistic Regression** achieved high accuracy, precision, and recall, indicating good overall performance.
- **K-Nearest Neighbors** showed slightly lower performance, likely due to its sensitivity to the choice of k.
- **Naive Bayes** performed well but had a trade-off between recall and precision.
- **Decision Tree Classifier** exhibited lower accuracy and recall, indicating challenges in capturing the complexity of the data.
- **Random Forest Classifier** achieved high accuracy, precision, and recall, demonstrating its effectiveness in handling complex relationships.

Linear SVC and Logistic Regression seem well-suited for this toxic comment classification task, balancing precision and recall.

I have performed a Hyperparameter search to find the best parameters by setting up a parameter grid for the model. I conducted a randomized search with 8 cross-validation folds and 15 iterations, obtaining the best parameters found. While GridSearchCV explores all parameter combinations exhaustively, it is computationally complex and time-consuming compared to the faster and less exhaustive randomized search. However, the random search may result in a suboptimal parameter combination, introducing uncertainty about finding the true best parameters. I will convert the probability of belonging to a class into a final binary class using a specified threshold.

My final reasonings are:

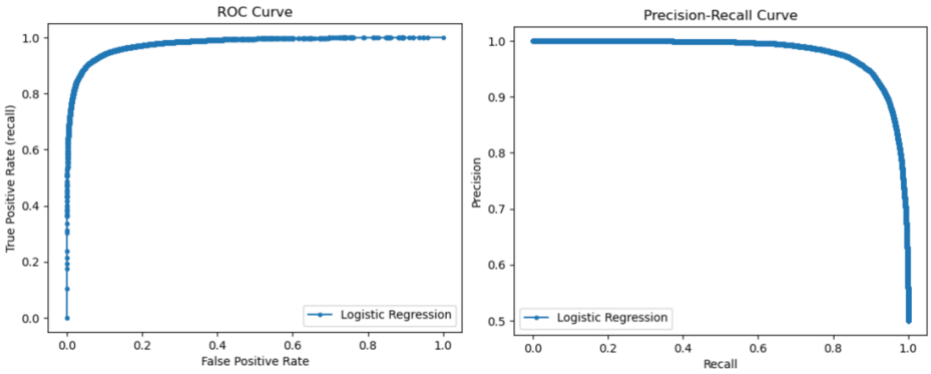
- **Linear SVC:** Efficient in high-dimensional spaces, effective in cases with a clear margin of separation.
- **Logistic Regression:** Simple, interpretable, and efficient for text classification tasks.

Considerations	Logistic Regression	Linear SVC
Interpretability	Offers easily interpretable coefficients.	Coefficient interpretation is less straightforward.

Probabilistic Output	Naturally provides probabilities.	Does not inherently produce probabilities.
Scalability	Typically, more computationally efficient, especially for large datasets.	May be less efficient with increasing dataset size.
Handling Imbalanced Data	Handles imbalanced datasets well.	Requires careful hyperparameter tuning for imbalanced data.
Implementation Simplicity	Simpler and easier to implement.	Involves more complex optimization tasks.
Robustness to Outliers	Less sensitive to outliers.	Can be influenced by outliers.
Implementation Simplicity	It is conceptually simpler and often easier to implement.	SVMs, including linear SVC, involve more complex optimization tasks, and parameter tuning might require more effort.

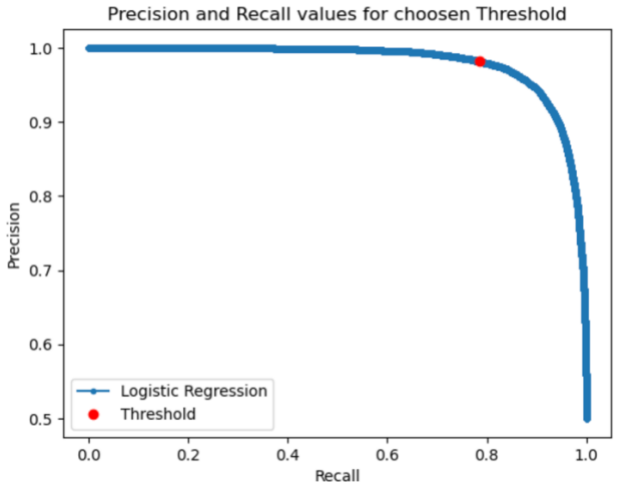
I will now evaluate the Logistic Regression model since it is very simple and efficient for text classification tasks by changing and finding the optimal threshold value.

Before tuning the threshold to evaluate the performance of my model, I will plot ROC curve and Precision-Recall curve plot. These plots will help to decide the "trade off" between all the metrics.



In the context of imbalanced classes, it is preferable to utilize a Precision-Recall curve instead of a ROC curve. Even when the dataset is balanced, examining the precision-recall curve can be valuable. This curve illustrates the tradeoff between precision and recall at various thresholds, allowing us to select a threshold value that aligns with the desired precision and recall values.

Since, I decided to **maximize precision (cost of FP > cost of FN)**, I have plotted at which threshold this happens.



Threshold value = 0.8717

Let's predict on test dataset using the first, simple logistic regression model.

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	Logistic Regression, D2V	0.923997	0.922931	0.910169	0.936056	0.937824	32615	33606	2228	3219	71668

Let's now predict using our chosen **final model** on test dataset.

	Model	Accuracy	f1	Recall	Precision	Specificity	TP	TN	FP	FN	y_test size
0	Logistic Regression, custom threshold	0.885151	0.87239	0.785148	0.981442	0.985154	28135	35302	532	7699	71668

We can see that the false positives have dropped to 532, precision has increased to 0.98(our goal was to maximize precision).

Future Work/Improvements:

1. Given more time, a more extensive hyperparameter search could be conducted, and additional models, such as transformers (e.g., BERT), could be explored for potential performance gains.
2. Further analysis of misclassified instances could provide insights into improving model robustness.
3. Conduct a more extensive hyperparameter search for each model to fine-tune performance.
4. Explore ensemble methods or advanced models to potentially achieve a better balance between precision and recall.

Can I Recommend to Client?

- **Precision Focus:** The model has been optimized for high precision, resulting in a precision of 98.14%.
- **Recall Trade-off:** However, this comes at a trade-off with recall (78.51%), which means there's a higher chance of missing some toxic comments.
- **Intended Use Case:** If the primary concern is minimizing false positives (misclassifying non-toxic comments as toxic), and the platform can tolerate a lower recall, then this model may be suitable.
- **User Feedback and Safety:** Gather user feedback to understand the tolerance for false positives and false negatives.

While the model is showing promising results, it's essential to fine-tune them based on the specific needs of the platform. The precision and recall achieved should be acceptable for deployment, but continuous monitoring and periodic retraining may be necessary to adapt to evolving online discourse patterns.