NLP Final Project

Toxic Comment Multi tag classification

Group Name: D

Group Members:

First name	Last Name	Student number
Jash	Vaghasiya	C0884733
Nivedini	Kathagonda	C0872720
Keval	Parmar	C0882386
Monil	Rupawala	C0882370
Sai Divya Madhuri	Guntupalli	C0882360

Submission date: ***17/08/2023***

Contents

Abstract

In this Study, we tackles the critical challenge of identifying and addressing toxic comments in online interactions. The anonymity afforded by the internet often leads to the proliferation of hurtful language that would seldom be expressed in face-to-face conversations. Leveraging the Wiki Corpus dataset spanning 2004 to 2015, this study utilizes exploratory data analysis (EDA) and feature engineering. The dataset's six toxicity categories, curated by human raters with some subjectivity, guide the analysis. Through EDA, the report unveils distribution patterns of toxic comments, paving the way for effective feature engineering to transform raw text data into meaningful input for machine learning models.

Subsequently, employing an ensemble of tree-based algorithms—Random Forest, Decision Tree, Gradient Boosting, and XGBoost—the study constructs a classification framework. These models, varying in complexity, collectively strive to curb the dissemination of harmful content across digital platforms, promoting a more inclusive online space. By harnessing the strengths of each algorithm, the research aims to strike a balance between precision and recall, enhancing accurate identification of toxic comments while minimizing false positives. Through evaluating and comparing model performance, this report contributes to the ongoing mission of fostering a safer and more respectful online environment by utilizing data analysis and machine learning techniques.

Introduction

In an era defined by digital connectivity, the realm of toxic online content presents a pressing challenge that warrants meticulous exploration. With the widespread adoption of online platforms for communication, individuals often feel emboldened to express hurtful and harmful sentiments that may not be voiced in face-to-face interactions. This burgeoning issue underscores the significance of effectively identifying and addressing toxic comments within the digital landscape.

The objective of this project is to delve into the intricate fabric of toxic online discourse and develop a robust classification framework. By harnessing advanced machine learning techniques, we aim to create a model capable of accurately detecting and categorizing toxic tweets. This model holds multifaceted potential, from enhancing content moderation on social media platforms to shedding light on the evolving dynamics of online interactions.

Analyzing toxic tweets assumes a paramount role, serving both a micro and macro perspective. On a micro level, this analysis empowers individuals to make informed decisions about the

content they consume and engage with, ensuring a safer and more respectful online experience. Simultaneously, from a macro perspective, it offers insights into the larger patterns and trends of toxic communication, facilitating the formulation of proactive strategies to curtail its proliferation.

Navigating the intricate landscape of toxic content necessitates a systematic and data-driven approach. By embarking on this endeavor, we endeavor to contribute to the ongoing dialogue surrounding digital etiquette and create a more inclusive and harmonious online environment.

Methodology

Dataset Details:

The project dataset contains several key components sourced from Kaggle:

id: A unique identifier associated with each entry. It contains non-null values for all 159,571 entries and is of object type.

comment_text: This column contains the textual content of the comments. It is also non-null for all entries and is of object type.

toxic: An integer column indicating whether the comment is classified as toxic. The values are binary (0 or 1) and denote non-toxic (0) or toxic (1) comments.

severe_toxic: This integer column indicates whether the comment is classified as severely toxic. Similarly, it uses binary values (0 or 1) to denote non-severely toxic (0) or severely toxic (1) comments.

obscene: An integer column that denotes whether the comment is classified as obscene. Like the previous columns, it uses binary values (0 or 1) to indicate non-obscene (0) or obscene (1) comments.

threat: This column, also integer type, indicates whether the comment contains threats. The values are binary (0 or 1), representing non-threatening (0) or threatening (1) comments.

insult: An integer column representing whether the comment is insulting. Similar to the previous columns, it uses binary values (0 or 1) to denote non-insulting (0) or insulting (1) comments.

identity_hate: This integer column signifies whether the comment contains identity-based hate. The values are binary (0 or 1), representing non-identity-hate (0) or identity-hate (1) comments.

	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

Data cleaning and preprocessing:

Dropping Columns while feature selection:

The **ID** is unique identification number so there is no use of that column.

Missing Values:

No Null values found in dataset.

Duplicate Values:

There are no duplicate values in the dataset.

Text pre-processing:

- The **tokenize_text** function employs the word_tokenize approach to break down content into individual words, transforming sentences or sections into lists of distinct words. This step serves as a foundational preprocessing step, facilitating subsequent data processing.
- The **lowercase_text** function ensures uniformity by converting all characters in the content to lowercase. This normalization mitigates case-sensitivity concerns, treating words like "Like" and "the" equivalently, thus eliminating potential discrepancies.
- The **remove_stopwords** function targets common stopwords, often overlooked in text analysis due to their high frequency across various texts and limited contribution to meaningful insights. These stopwords include terms like "and," "or," and "not," and their removal allows the focus to shift towards words carrying more significant semantic weight.
- Addressing accentuation marks, the **remove_punctuation** function utilizes regular expressions to eliminate them from the content. Particularly beneficial when analyzing word meanings rather than sentence structures, this step reduces punctuation noise and refines the analysis focus.
- The **lemmatize_text** function streamlines textual data using the WordNet Lemmatizer, breaking down words into their base or root forms. Variations such as "running," "ran,"

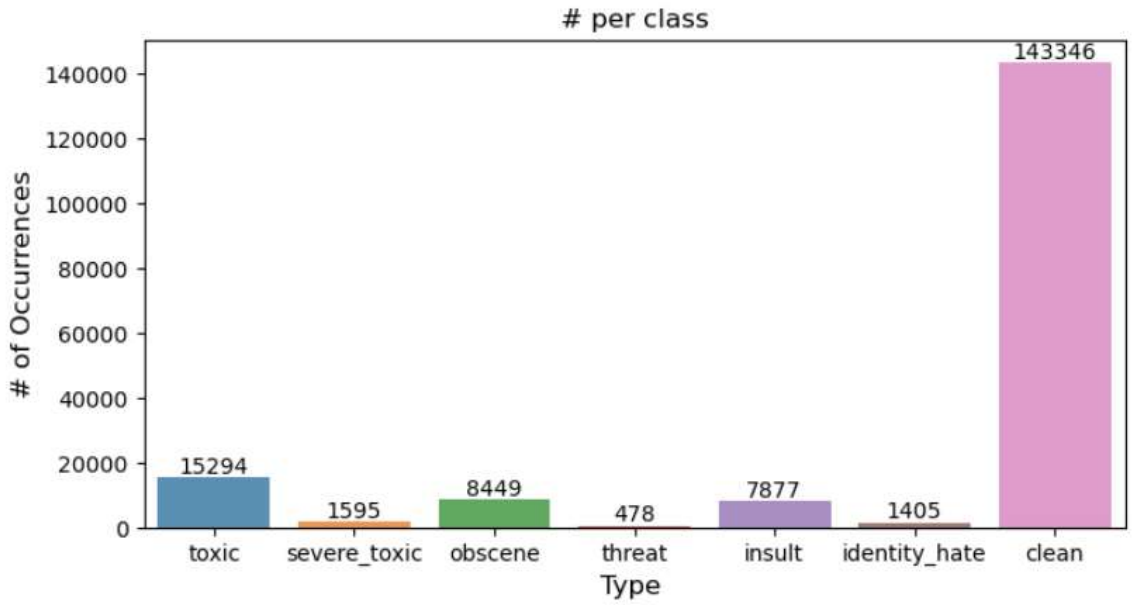
and "runner" can all be reduced to the common base form "run." This not only reduces data dimensionality but also enhances the ability to grasp the core meanings of words regardless of their grammatical forms.

- Lastly, the **preprocess_text** function acts as a comprehensive wrapper, systematically applying all the aforementioned operations to a specific column within a pandas DataFrame. This systematic approach ensures a consistent and thorough preprocessing pipeline, preparing textual data for subsequent analysis with enhanced clarity and relevance. DataFrame. Its central reason is to cleanse and standardize the content for consequent inquire about or utilization in machine learning models. While the first audit substance remains within the 'unprocessed_review' column, the cleaned adaptation replaces the 'review' column, coming about in a refined dataset that's more conducive to important examination. This handle guarantees that the information is well-prepared, reliable, and prepared for progressed investigation or modeling endeavors.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	clean	unprocessed_comment	count_sent	count_word	co
0	0000997932d777bf	[explanation, edits, made, username, hardcore,...	0	0	0	0	0	0	True	Explanation\nWhy the edits made under my usern...	2	43	
1	000103f0d9cfb60f	[daww, match, background, colour, im, seemingl...	0	0	0	0	0	0	True	D'aww! He matches this background colour I'm s...	1	17	
2	000113f07ec002fd	[hey, man, im, really, trying, edit, war, guy,...	0	0	0	0	0	0	True	Hey man, I'm really not trying to edit war. It...	1	42	
3	0001b41b1c6bb37e	[cant, make, real, suggestion, improvement, wo...	0	0	0	0	0	0	True	"\n\nMore\nI can't make any real suggestions on...	5	113	
4	0001d958c54c6e35	[sir, hero, chance, remember, page, thats]	0	0	0	0	0	0	True	You, sir, are my hero. Any chance you remember...	1	13	
...	...	...	...	...	...	...	...	...	...	...	...	...	

EDA and Visualization

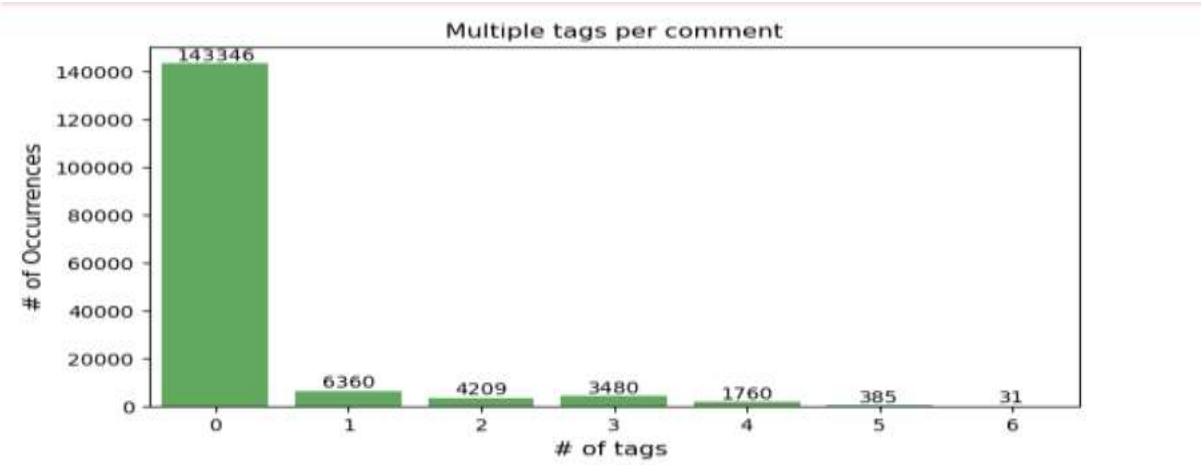
Distribution of Tags:



- Dataset Composition: The dataset comprises 159,571 comments categorized into distinct toxicity types, including "Toxic," "Severe Toxic," "Obscene," "Threat," "Insult," and "Identity Hate."

- Challenges and Relevance: The imbalanced data distribution presents challenges in effectively detecting and addressing toxicity. This scenario mirrors real-world online interactions, highlighting the importance of developing robust strategies to predict and manage toxic content.

Multiple Tags Distribution :

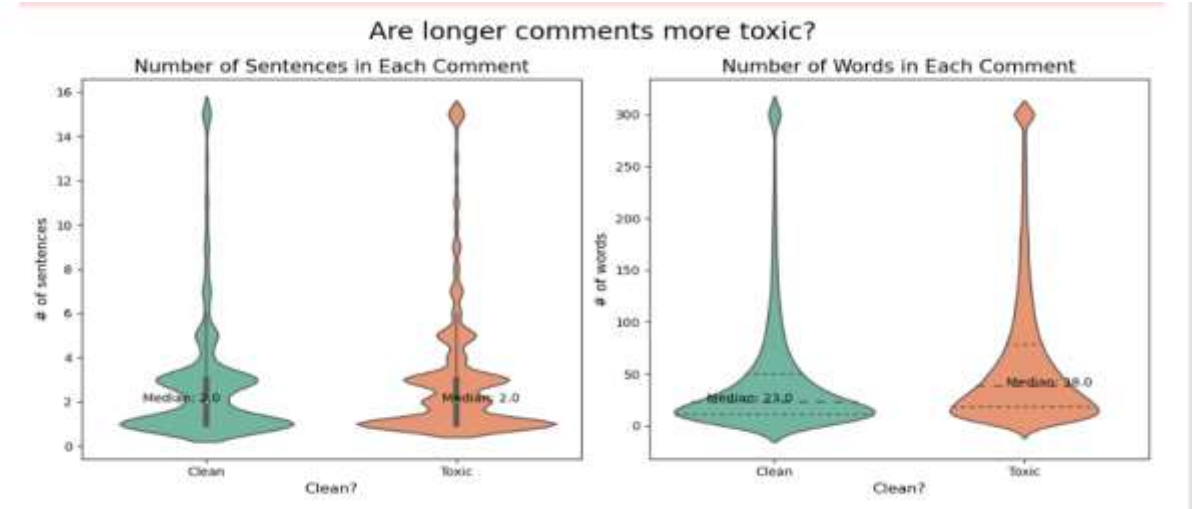


- Toxicity Diversity: The dataset showcases a diverse spectrum of toxicity, with the highest count (143,346 instances) attributed to non-toxic comments. However, it also reveals

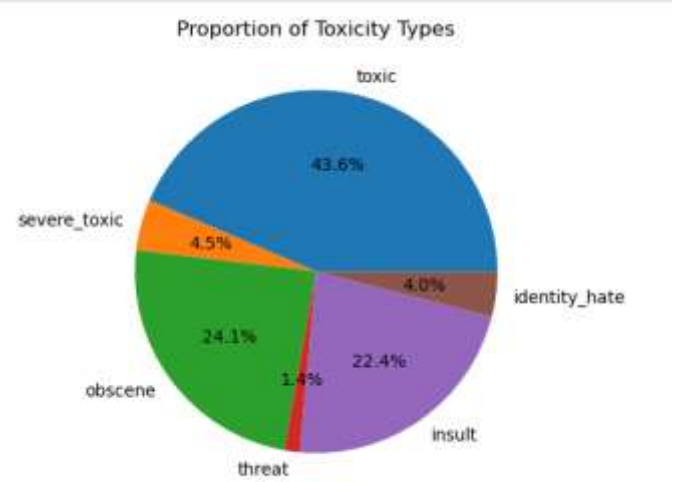
instances of singular, dual, triple, and even more complex combinations of toxic attributes within comments.

- Multi-Dimensional Insights: A nuanced exploration of the dataset highlights various degrees of toxicity. While 6,360 comments exhibit single toxic tags, over 4,209 instances manifest a complex interplay of three toxic attributes. This multifaceted distribution provides rich insights into the intricate nature of toxicity within online discussions.

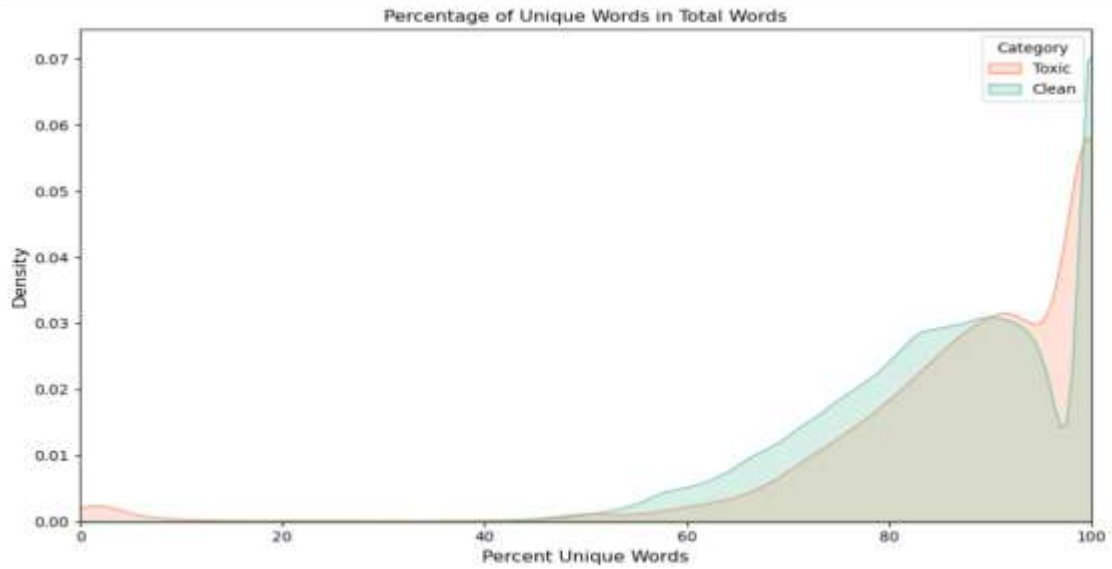
Effect of length of comments:



Distribution of Toxic Labels:



Percentage of unique words:



- According to the KDE distribution map, clean comments have a larger percentage of unique words, indicating a rich and varied vocabulary. Toxic remarks, on the other hand, frequently have a greater dispersion and smaller concentration of unique words, presumably reflecting repeated linguistic patterns linked with toxicity.

WordClouds

- **Clean Comments Word cloud**

WordCloud of Toxic comments

Observation of all the Word Clouds:

Clean Comment Word Cloud:

- Contains words related to positive interactions and engagement on a platform, such as "talk," "page," "like," and "article." Represents a positive and open communication environment where users are discussing topics without any offensive or sensitive language.

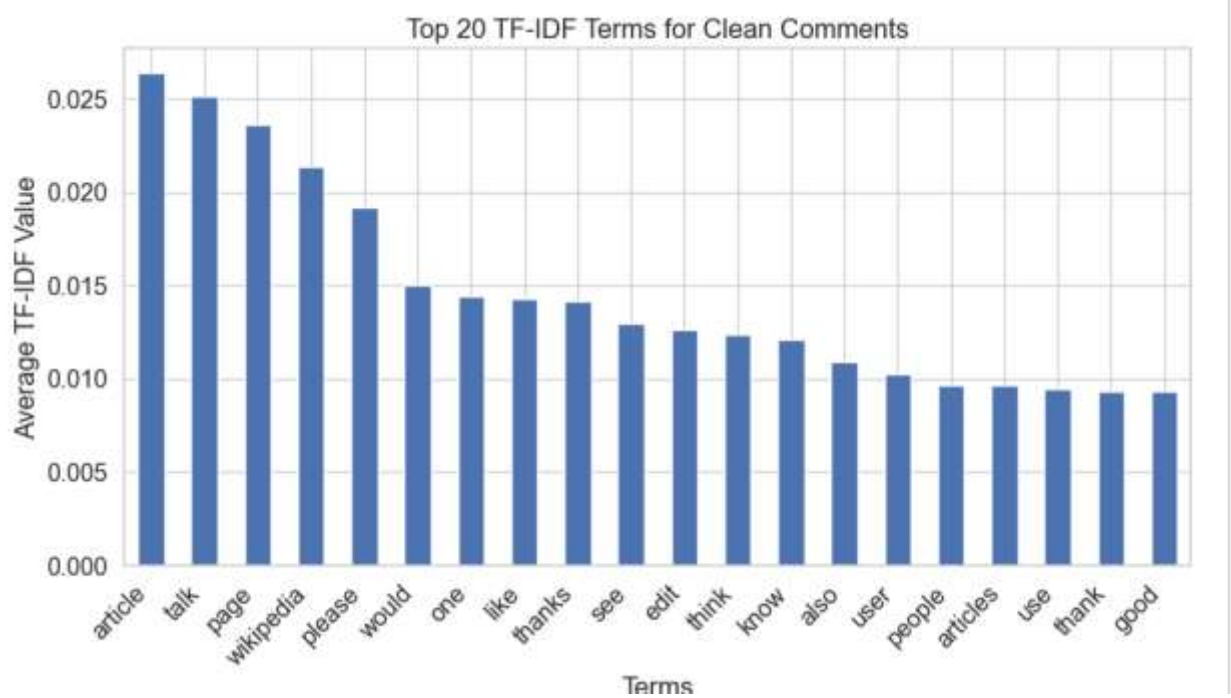
Toxic Comment Word Cloud:

- Comprises words that have been intentionally filtered or replaced due to their potentially offensive or inappropriate nature.Reflects a moderation or content control mechanism in place to maintain a respectful and safe online space by masking or blocking certain words that could be harmful or violate community guidelines.

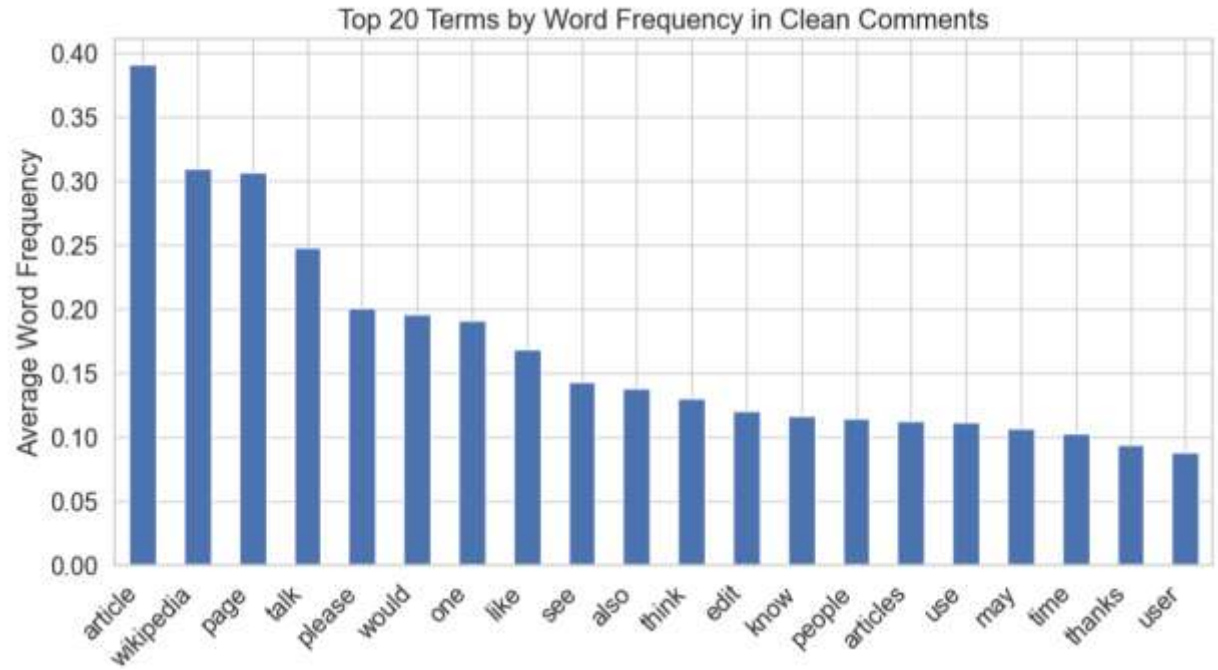
Overall:

The clean comment word cloud shows positive involvement with phrases such as 'chat,' 'page,' 'like,' and 'article,' whereas the censored word cloud emphasises attempts to preserve a courteous atmosphere by obscuring potentially harmful words.

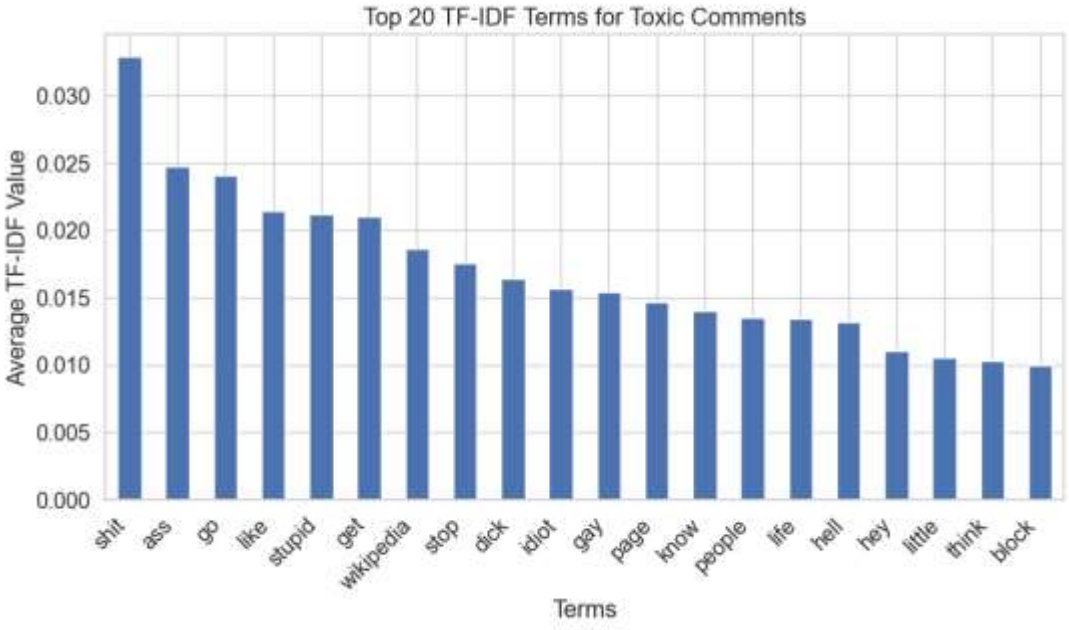
Top 20 TF-IDF Terms for Clean Comments



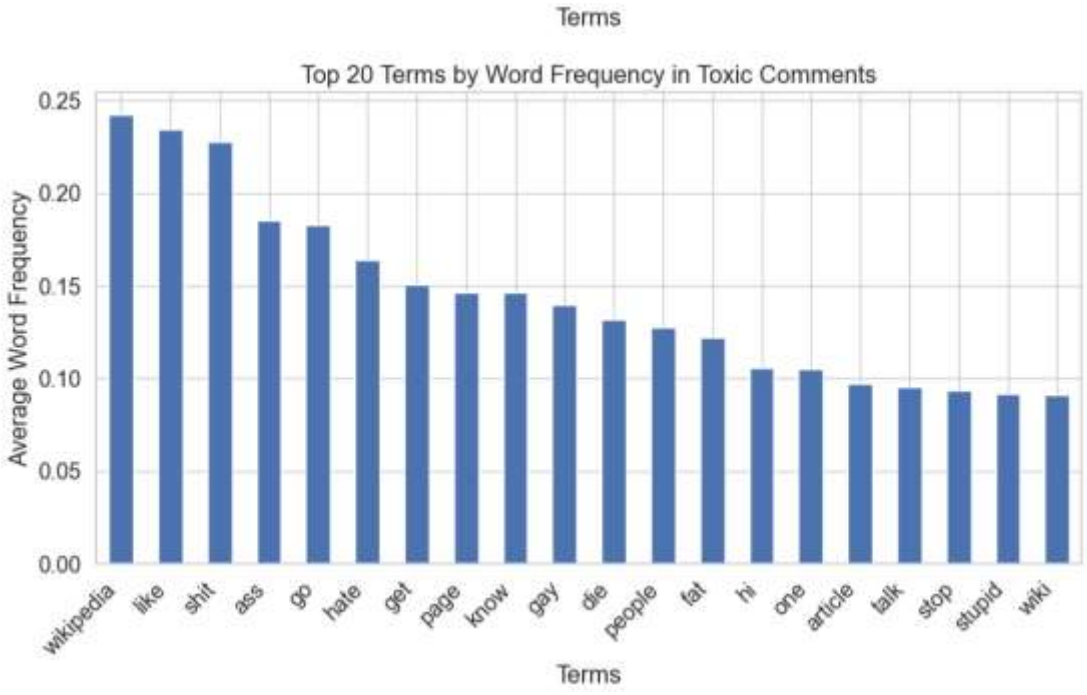
Top 20 Terms by word frequency for Clean Comments



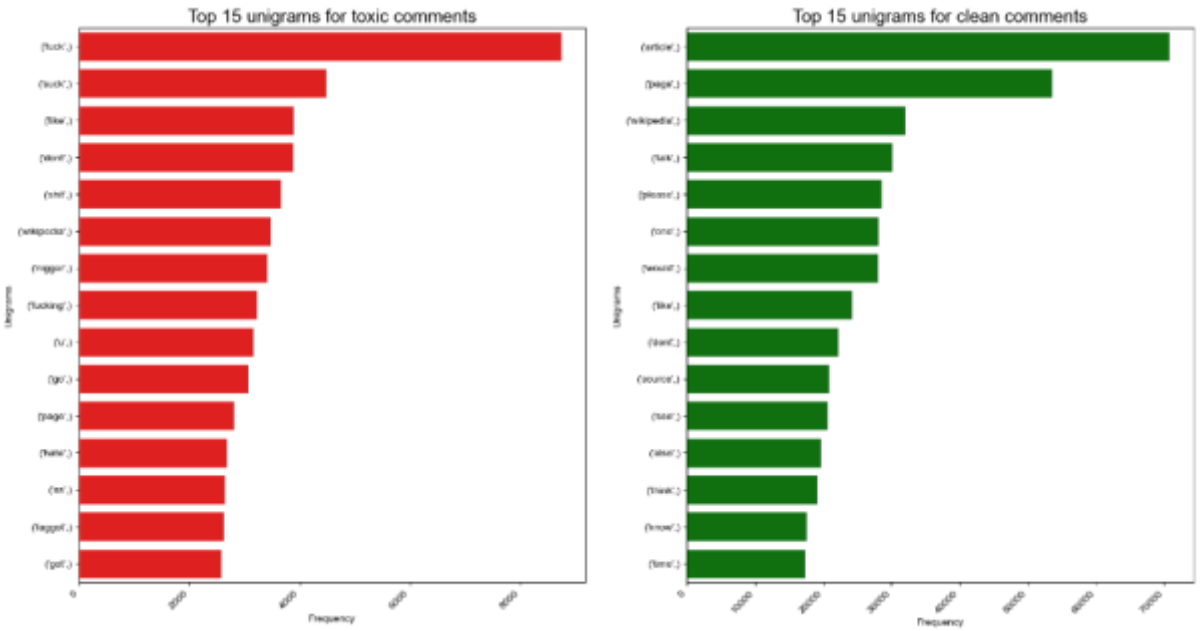
Top 20 TF-IDF Terms for Toxic Comments:



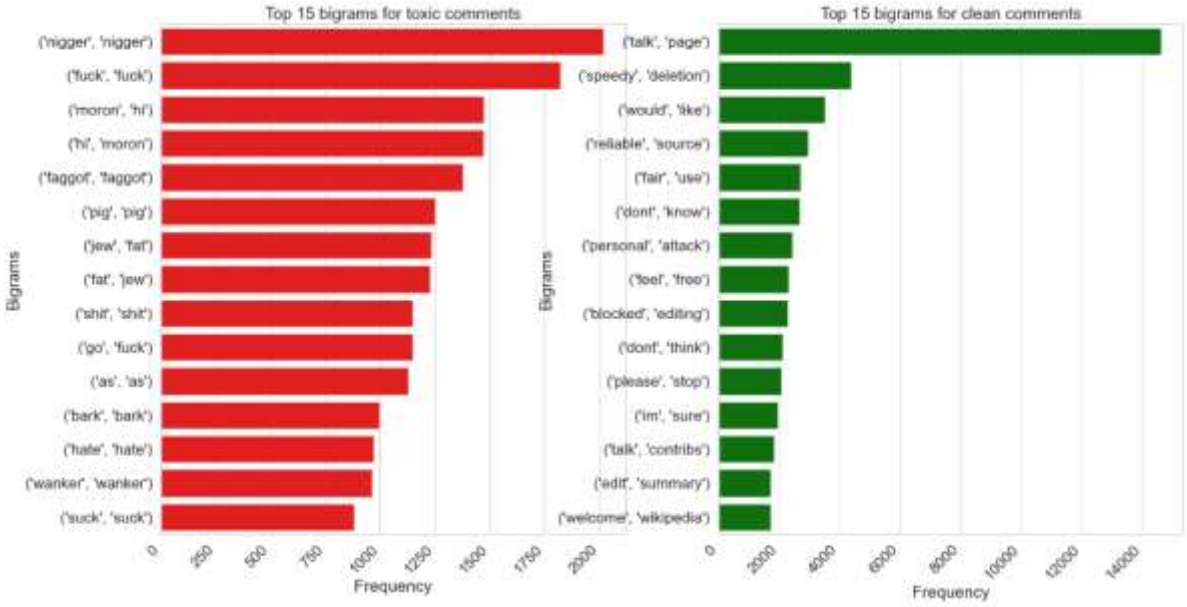
Top 20 Terms by word frequency for Toxic Comments



Unigrams



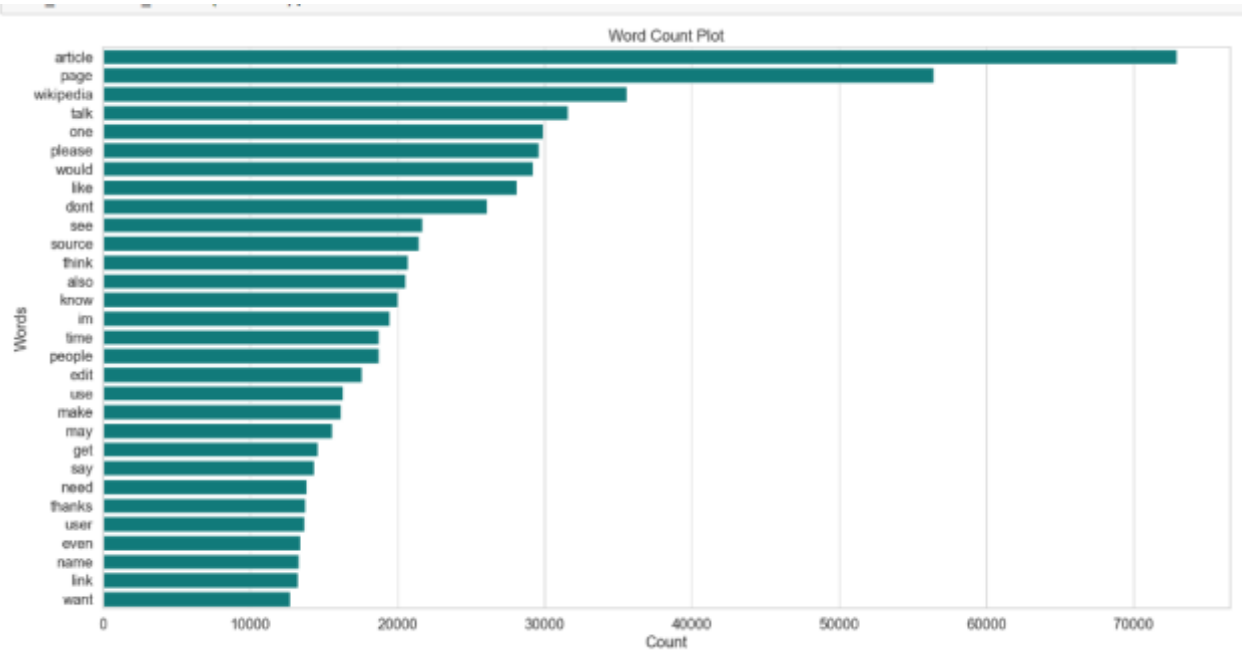
BiGrams



- Some prohibited words are among the top unigrams in poisonous comments, whereas 'article,' 'page,' and 'wikipedia' rank among the top unigrams in clean comments.BIgrams and trigrams provide more background information.The bigram "talk, page" indicates concentrated dialogue and involvement within certain platform parts.

Notably, the combination of "source, reliable" denotes a strong emphasis on information accuracy and reliability.

Word Count Plot:



- The visualisation shows that the words 'article,' 'page,' and 'wikipedia' are among the top words in terms of word count, showing their prominence in the analysed text data.

Feature Engineering:

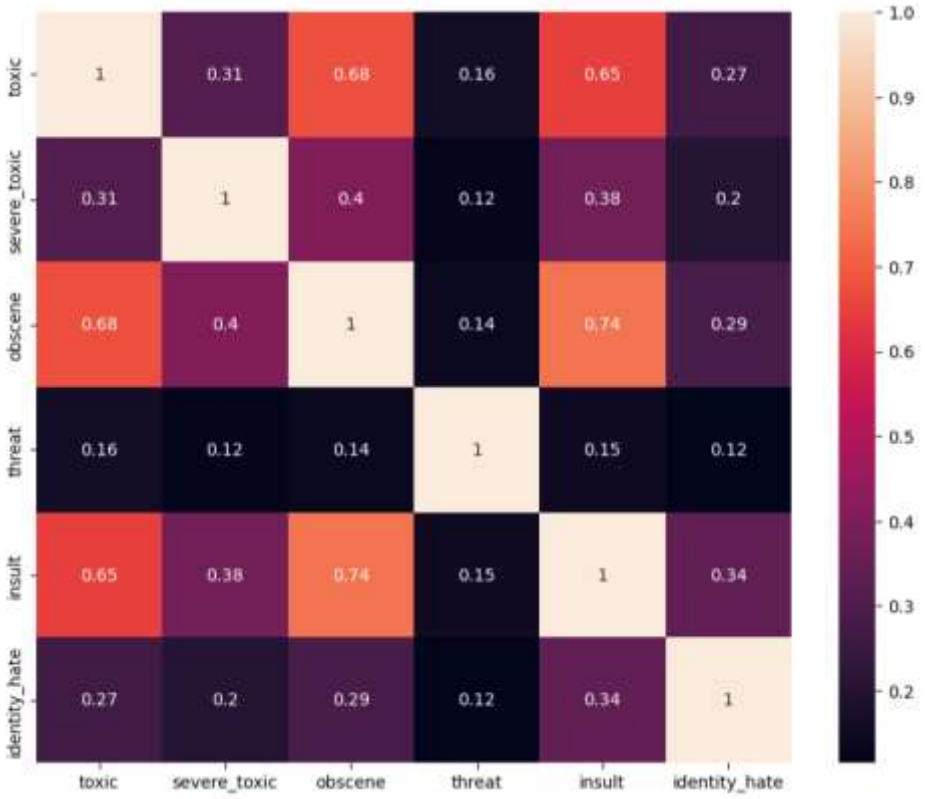
Description of DataFrame Feature Engineering Columns:

- **Counting Sentences**: Calculates the number of sentences in each comment. Creates a new column "count_sent" in the DataFrame.
- **Counting Words**:Computes the total number of words in each comment.Introduces a new column "count_word" in the DataFrame.
- **Counting Unique Words**:Determines the count of distinct words in each comment.Appends a new column "count_unique_word" to the DataFrame.
- **Counting Letters**:Measures the total number of letters (including spaces) in each comment.Adds a new column "count_letters" to the DataFrame.
- **Calculating Word Uniqueness Percentage**:Computes the percentage of unique words compared to the total word count in each comment.Generates a new column "word_unique_percent" in the DataFrame.
- **Counting Punctuations**:Evaluates the number of punctuation characters in each comment.Establishes a new column "count_punctuations" within the DataFrame.
- **Counting Uppercase Words**:Determines the count of words in uppercase (all-caps) in each comment.Adds a new column "count_words_upper" to the DataFrame.
- **Counting Title Case Words**:Identifies the count of words in title case (capitalized) in each comment.Introduces a new column "count_words_title" in the DataFrame.

- **Counting Stopwords**:Computes the number of common stopwords in each comment (e.g., "the," "is," "and").Appends a new column "count_stopwords" to the DataFrame.
- **Calculating Mean Word Length**:Computes the average length of words in each comment.Creates a new column "mean_word_len" in the DataFrame.

	ID	drugName	condition	review	rating	date	usefulCount	unprocessed_review	count_word	count_unique_word	count_letters	count_punctuat
61		Valsartan	Left Ventricular Dysfunction	[side, effect, take, combination, bystolic, 5,...	9	2012-05-20	27	"It has no side effect, I take it in combinati...	9	9	79	
60		Guanfacine	ADHD	[son, halfway, fourth, week, intuniv, became, ...	8	2010-04-27	192	"My son is halfway through his fourth week of ...	66	56	621	
03		Lybrel	Birth Control	[used, take, another, oral, contraceptive, 21,...	5	2009-12-14	17	"I used to take another oral contraceptive, wh...	73	54	685	
00		Ortho Evra	Birth Control	[first, time, using, form, birth, control, i03...	8	2015-11-03	10	"This is my first time using any form of birth...	43	33	380	
96		Buprenorphine / naloxone	Opiate Dependence	[suboxone, completely, turned, life, around, f...	9	2016-11-27	37	"Suboxone has completely turned my life around...	62	56	647	

Correlation Plot:



TF-IDF Vectorization

- The Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer is used to convert textual reviews into numerical representations. The TfidfVectorizer, initialised with max_features=500, is used in this operation. This option considers the top 500 terms with the highest term frequency, significantly reducing the dataset's dimensionality. This strategy reduces noise and superfluous expressions in the corpus by focusing on important words. The resultant matrix is then constructed into the review_df DataFrame, which is then joined with the original df (without the'review' column) to generate data_final. This combination produces a full data frame that is enhanced with the TF-IDF properties of the reviews.
- By selecting a feature count of 500, a careful balance is created between obtaining key insights from reviews and controlling computational complexities and memory usage. This is especially important when dealing with large datasets, as it helps to preserve efficiency while preserving important information from the reviews.

Modeling:

MultiNomial NB classifier:

The Naive Bayes model maintains a steady level of performance across training and testing phases, with scores of around 0.435 and 0.434 for training and testing, respectively. This implies a lack of significant overfitting in the model.Although the overall accuracy is relatively modest at 0.4335, the model presents diverse precision and recall scores across distinct categories of toxic comments. Remarkably, it demonstrates heightened recall rates for categories such as 'toxic,' 'obscene,' and 'insult.' This indicates the model's capability to accurately identify a substantial portion of instances within these categories, despite the corresponding precision being low.

Random Forest Classifier:

The Random Forest model showcases an impressive training score of 0.9999, signifying a robust fit to the training dataset. Simultaneously, it maintains a notable test score of 0.9095, underscoring its capacity to generalize effectively to previously unseen data.While achieving a commendable accuracy score of 0.9095, the model's precision and recall values display variations across distinct toxic comment categories. Noteworthy is its notable precision and recall for classifications like 'toxic' and 'obscene,' highlighting its adeptness in accurately identifying and categorizing instances within these classes. Nevertheless, its performance is comparatively lower in categories such as 'severe_toxic,' 'threat,' and 'identity_hate.'

Decision Tree Classifier:

The Decision Tree model has a significant training score of 0.9999, indicating that it is well aligned with the training dataset. At the same time, it achieves a decent test score of 0.8431, showing a reasonable capacity to generalise efficiently to new and previously unknown data.When applied to distinct harmful comment categories, the Decision Tree model demonstrates variances in precision and recall despite reaching a very acceptable accuracy score of 0.8758. Its performance stands out in areas such as 'obscene' and 'toxic,' displaying well-matched accuracy and recall. However, it has difficulty properly categorising occurrences in classes such as'severe_toxic,' 'threat,' and 'identity_hate,' where accuracy and recall values decline.

Gradient Boosting classifier:

The achieved test score of 0.9092 and a train score of 0.9151 reflect a strong generalization capability during training, ensuring effective performance on unseen data. The model's precision and recall metrics exhibited variations across toxic comment categories, excelling notably in categories like 'obscene' and 'toxic,' while encountering challenges in categories such as 'severe_toxic,' 'threat,' and 'identity_hate,' highlighting the nuanced nature of its predictive prowess.

Confusion Matrix:

Confusion Matrix Observation:

1. Multinomial NB:

- The Naive Bayes model did not yield higher accuracy due to its challenge in effectively capturing the complexities and nuances present in the dataset.

2. Decision Tree

- The diagonal elements, which signify accurate predictions, are more uniformly distributed in the Decision Tree model compared to [another model]. This suggests that the Decision Tree maintains better differentiation across various classes, although it still exhibits a bias towards predicting the last type.

3. Random Forest Confusion :

- The higher values along the diagonal indicate that this model surpasses the performance of the Decision Tree. Nevertheless, to the Decision Tree model, there exists a notable inclination towards predicting the last class.

4. Gradient Boosting:

- The diagonal elements, which indicate accurate predictions, are more uniformly spread out compared to . This suggests that the Gradient boosting preserves greater distinctions among classes, albeit with a tendency to favor predicting the last type.

In the model comparisons, the Naive Bayes approach exhibited limited accuracy due to its inability to capture intricate dataset intricacies. Conversely, the Decision Tree model demonstrated improved class differentiation through evenly distributed accurate predictions, albeit with a preference for predicting the last class. The Random Forest model outperformed the Decision Tree, displaying higher diagonal values, yet still exhibited a tendency to favor the final class. Similarly, Gradient Boosting maintained balanced accurate predictions, resembling the Decision Tree's behavior in showing a bias toward predicting the last class.

Improvements :

In the pursuit of optimizing model performance, K-Fold cross-validation was adeptly employed. This methodology partitions the dataset into multiple subsets or "folds," where each fold serves as both a training and validation set. By iteratively rotating through these folds, each time selecting one as the validation set while using the rest for training, the model's efficacy is rigorously assessed across various data distributions. This holistic approach mitigates the risk of overfitting and offers a more comprehensive evaluation of the model's generalization capability. In the case of the presented model, the average cross-validation score of 0.9103 attests to its remarkable adaptability and performance consistency across diverse folds. This further solidifies the model's credibility and enhances our confidence in its predictive accuracy for unseen data.

- Employed K-Fold cross-validation to refine the Random Forest model's capabilities. K-Fold validation involves dividing the dataset into multiple subsets (folds) for training and testing, leading to improved model robustness and generalization.
- Achieved an impressive average cross-validation score of 0.9103. This score signifies the model's enhanced performance across different folds and indicates its ability to generalize well to diverse data subsets.
- The Random Forest model demonstrated its effectiveness by yielding a test score of 0.9095. This score underscores the model's capacity to accurately predict outcomes for previously unseen data instances, validating its real-world utility..

- The model's precision and recall values exhibited significant variations across different classes. Notably, some categories saw substantial precision and recall rates, indicating the model's strength in identifying instances accurately. However, challenges were observed in other categories, where precision and recall were comparatively lower.

- The model displayed an overall micro-average F1-score of 0.55. This metric provides a comprehensive view of the model's performance by considering both precision and recall. The score suggests that the model strikes a balance between accurately identifying positive instances and capturing the relevant proportion of actual positive instances in the dataset.

Results

The process of modeling on the provided real life comments data involved a systematic approach encompassing multiple phases. The initial stage encompassed Exploratory Data Analysis (EDA), where an array of visualization techniques like histograms, bar graphs, and word clouds were harnessed to comprehend the data's characteristics comprehensively. These visualizations facilitated pattern recognition, and a deeper understanding of the dataset.

Subsequently, during the data preprocessing stage, efforts were directed towards cleaning and preparing the data for modeling. A significant departure from the conventional approach was the utilization of K-Fold cross-validation to tackle class imbalance. This technique involved partitioning the dataset into distinct folds, effectively using each fold as a validation set while the remaining folds were utilized for training. By iteratively cycling through these folds, the model's performance was more accurately assessed, and overfitting risks were mitigated. This alternative approach helped maintain data integrity and provided a robust framework for model evaluation and enhancement.

Conclusion

Our journey through the realm of data science unfolded within our Jupyter Notebook, commencing with a thorough exploration via exploratory data analysis (EDA) and visualization. This initial phase provided a comprehensive understanding of the dataset's architecture, distribution patterns, and potential anomalies.

The substantial presence of textual data within the dataset necessitated meticulous preparation. Techniques like tokenization, lemmatization, stopword elimination, and

punctuation removal were judiciously applied to ready the text data for subsequent analysis.

Given the categorical nature of certain variables, we leveraged label encoding to transform them into numerical representations, ensuring compatibility with machine learning methodologies.

To enable machine learning, we harnessed TF-IDF vectorization, which adeptly transformed the preprocessed text into a numerical format. This technique not only converted the text into actionable numerical vectors but also accorded significance to lesser-seen yet more informative expressions.

As we transitioned into the modeling phase, a recurrent obstacle emerged in the form of class imbalance. This discrepancy in class distribution carried the potential to introduce biases, particularly favoring the dominant class within our models. To navigate this challenge, we opted for a different approach—employing K-Fold cross-validation. This technique partitions the dataset into multiple subsets, or "folds," allowing each fold to serve as both a validation set and a training set iteratively. This method effectively mitigates biases while facilitating a robust assessment of model performance across various data distributions.

In subsequent stages, our focus remained steadfast on training and evaluating a range of classification models. While these models exhibited promise, they also unveiled areas of refinement, particularly in effectively handling the intricate nature of textual data. Notably, the adoption of K-Fold cross-validation demonstrated its significance, ensuring that our models maintained their integrity and generalization capabilities without overfitting.

To wrap up, this notebook stands as a vivid portrayal of the multifaceted nature inherent in data science ventures, particularly when dealing with unstructured data like text. While substantial strides were taken, the expedition itself emphasized the pivotal role of continuous refinement and iterative processes. To further enhance performance, future endeavors might involve delving deeper into the intricacies of hyperparameter tuning, embarking on explorations into advanced text representation strategies, or delving into the possibilities presented by deep learning models.

Future Work

Numerous potential avenues for future endeavors present themselves. To enhance performance, there is the possibility of delving further into the fine-tuning of hyperparameters, exploring advanced methods of text representation, and even delving into the realm of deep learning models.

References

Toxic Comment Classification Challenge | Kaggle.
(n.d.). https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge

Towards Data Science. (n.d.). Text Preprocessing in Natural Language Processing using Python. https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8

3.1. Cross-validation: evaluating estimator performance. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/cross_validation.html

Deepanshi. (2023). Text Preprocessing in NLP with Python Codes. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/