Toxic Comment Classification Using Machine Learning

IDS 575 – Spring 2024

April 27, 2024

Group 3:

Dhruv Jaiswal

Eamonn Michael McDonagh

Damodar Prabhu

Abilash Sivakumar

Abstract

This project tackles the challenge of moderating toxic comments on online platforms with a high level of confidence. We have explored machine learning models that can classify several types of toxicity, sourced from Wikipedia Talk pages, and applied three models: Logistic Regression, K-Nearest Neighbors, and Linear SVC, each with their own strengths.

Our natural language processing techniques, such as word embeddings and n-grams, have significantly enhanced the models' ability to recognize subtle nuances in language and context. We also employed multiple rounds of feature selection to ensure their accuracy.

After training and evaluating the models, Logistic Regression emerged as the most accurate and precise model, correctly identifying toxic comments with high accuracy.

This work not only advances the field of automated content moderation but also provides a framework for further research on enhancing online discourse. By creating more accurate and efficient models for detecting toxic comments, we are confident of the possibility of fostering a safer and more inclusive online community.

Introduction

The rise of online platforms has made communication more accessible and convenient. However, this increased connectivity also brings the challenge of toxic comments that can negatively impact user experiences and engagement. These comments can range from insults and identity-based hate to severe toxicity, threats, and obscenities. That is why it is crucial to implement an effective moderation system.

Our project aims to tackle this issue by developing a robust machine learning model to categorize comments into different toxicity categories. We will use natural language processing (NLP) techniques and explore multiple machine learning models to train the model to identify and moderate toxic content more effectively.

Our solution goes beyond the typical binary classification of toxic or non-toxic comments. Instead, we will categorize the comments into specific sub-categories to better understand the nuances of toxic comments, allowing for appropriate actions to moderate them. Additionally, we will attempt to ensure that the model is fair and unbiased so that it will not discriminate against any group of users.

By effectively moderating toxic content, we aim to create a safer, more inclusive, and welcoming online community.

Literature Review

Our approach to classifying toxic comments is grounded in extensive prior research and innovative techniques. Specifically, we have considered the Logistic Regression method originally presented by Davidson et al. (2017) for improved accuracy and reliability for multi-class classification. In addition, we have considered BERT (Bidirectional Encoder Representations from Transformers), a feature extraction technique introduced by Devlin et al. (2018). By leveraging these studies and integrating them with our own application, we aim to review machine learning's potential for meaningful contribution to the ongoing discourse around enhancing automated content moderation systems. Our objective is to demonstrate machine learning's ability to provide a safer, more welcoming online environment by enabling platforms to identify and remove toxic comments quickly and effectively.

Methodology

Dataset Description

The study leverages a dataset from the "Toxic Comment Classification Challenge" hosted on Kaggle, encompassing a diverse collection of user-generated comments. The dataset comprises approximately 160,000 examples, each meticulously labeled across a range of toxicity categories including, but not limited to, toxic, severe toxic, obscene, threatening, insulting, and identity-based hate. Following the vectorization process, the data manifests a

rich feature space, anticipated to contain tens of thousands of unique tokens derived through the TF-IDF vectorization technique. Alternatively, when utilizing word embeddings, each comment is represented by a fixed-size feature vector, typically ranging between 50 to 300 dimensions, providing a condensed yet informative representation of the textual data. This expansive and multifaceted dataset forms the backbone of our investigation into automated toxicity detection, offering a comprehensive basis for modeling and analysis.

Origin and Composition:

The dataset utilized for this project originates from the "Toxic Comment Classification Challenge" available on Kaggle. It comprises user-generated comments from Wikipedia Talk pages and public discussion forums associated with specific Wikipedia articles. These forums serve as a platform for Wikipedia editors to discuss content accuracy. This dataset has diverse content to train models that can recognize a spectrum of toxic behaviors.

The dataset, consisting of 159,571 comments split into training and test subsets sourced from Wikipedia Talk pages and each labeled for varying toxicity levels, provides vast language context for the training and testing robust machine-learning models. It includes various user interactions, from constructive to toxic dialogue. It will be analyzed through a descriptive analysis focusing on label distribution, word count statistics, and language features across the different toxicity categories.

Features:

The primary feature of each data entry is the comment text itself, accompanied by several binary labels that categorize the comment across different dimensions of toxicity:

- `comment_text`: Text content of the user comments.
- `toxic`: Binary indicator of general toxicity.
- `severe_toxic`: Binary indicator of severe toxicity levels.

- `obscene`: Binary indicator of obscene content.
- `threat`: Binary indicator of threatening language.
- `insult`: Binary indicator of insulting content.
- `identity_hate`: Binary indicator of comments that promote hate based on identity.

These labels enable the construction of multi-label classification models. The preprocessing of this textual data involves standard NLP techniques, including tokenization, removal of stop words, and vectorization using methods such as TF-IDF or word embeddings to transform the raw text into a structured form suitable for machine learning.

Data Types:

In the dataset utilized for this study, the 'comment_text' column comprises textual data, consisting of free-form comments extracted from user interactions within an online platform. This unstructured text data encapsulates the diverse linguistic expressions characteristic of online discourse. In contrast, the dataset's labeled attributes—'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', and 'identity_hate'—are represented as categorical/binary variables. These variables take on a binary form, designated by 0 or 1, to signify the absence or presence, respectively, of the specific type of toxicity they denote. Each label distinctly marks the relevant type of offensive or inappropriate content, as identified by human raters in accordance with the operational definitions provided for the study.

Descriptive Analysis of the Dataset:

In the examination of the dataset for the study, the distribution of labels is markedly uneven, a crucial factor to consider when constructing and refining predictive models. Analysis discloses that a substantial 10.2% of the comment corpus is marked as 'toxic,' signifying the presence of clear negative sentiment or intention to harm. Meanwhile, a relatively minuscule 1.1% falls under the 'severe toxic' category, denoting a highly extreme form of harmful content.

Obscenities are present in 5.3% of the comments, and these often intersect with other forms of toxic language. Direct threats, encapsulated by the 'threat' label, are identified in a mere 0.3% of the dataset, indicating their scarcity. Personal attacks or disparaging comments, classified as 'insults,' account for 5.0% of the dataset. Finally, 'identity hate,' which encompasses comments targeting specific groups or identities, comprises 0.9% of the comments. This distribution underscores the challenges inherent in detecting and categorizing toxic online behavior, particularly in developing models sensitive enough to capture less frequent but more severe expressions of toxicity.

Models:

Our approach involves the integration of various predictive models to address the intricacies of classifying toxic comments. Our methodology is designed to optimize accuracy and comprehensiveness, and it represents recent advancements in natural language processing. We have conducted training and testing to assess if our approach is reliable and effective, and we are confident that it offers a promising solution to the challenges associated with toxic comment classification.

Logistic Regression (Baseline Model):

Design of the Model:

Logistic Regression is selected as the baseline model due to its simplicity and effectiveness in binary classification tasks.

It assumes a linear relationship between the log-odds of the dependent variables and the predictors.

Mathematical Description:

The logistic function governs the model's predictions by mapping the input features to probabilities. The logistic function, also known as the sigmoid function, is defined as:

Equation:

$$P(y = 1 \mid X) = \frac{1}{(1 + e\text{^} - z)}$$

where z is the linear combination of the input features and model parameters.

Logistic Function:

The logistic function converts the linear combination of input features into probabilities, providing the probability that a particular instance belongs to the positive class.

Loss Function (Cross Entropy):

To train the logistic regression model, a loss function called cross-entropy is minimized during the optimization process. The cross-entropy loss function is defined as:

J(θ)=−N1∑i=1N[yilog(y^i)+(1−yi)log(1−y^i)]

where N is the number of samples, yi is the true label of the i-th sample, and y^i is the predicted probability of the positive class for the i-th sample.¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

K-Nearest Neighbors Model:

K-Nearest Neighbors (KNN) utilizes the majority voting system among the K-nearest data points

Design of the Model:

KNN is a non-parametric method for classification based on the similarity of instances in the feature space.

It relies on the majority vote of the k nearest neighbors to classify a new instance.

Mathematical Description:

Euclidean distance is commonly used as the distance metric to measure similarity between instances.

Equations:

Euclidean Distance: d(x,y)=∑i=1n (xi −yi)2

Linear SVC:

Linear Support Vector Classification (Linear SVC) is designed to find an optimal hyperplane that separates classes in the feature space with the maximum margin. In a simpler 2D space, this hyperplane can be visualized as a line, but in higher-dimensional spaces, it represents a plane or hyper-surface. The key objective is to separate the classes by the widest possible margin.

Margins and Support Vectors:

The equation defines the separation hyperplane $\omega . x + b = 0$, where:

ω Represents the weight vector, which is normal to the hyperplane.

x Denotes the feature vector.

b is the bias term.

For a correctly classified dataset:

For positive classes $(y_i = 1), \omega . x_i + b \geq 1$

For negative classes $(y_i = -1), \omega . x_i + b \leq -1$

These constraints ensure that each class data point lies on the correct side of the margin. The data points that exactly meet $\omega . x_i + b = 1$ or $\omega . x_i + b = -1$are the "support vectors."

Objective Function:

The objective of Linear SVC is to maximize the margin, which is inversely proportional to $||\omega||$. Hence, the objective function to minimize is $||\omega||^2$ simplified to:

$$min_{\omega,b}\ \frac{1}{2}||\omega||^2$$

This is subject to the constraint for each data point i:

$$y_i(\omega.x_i + b) \geq 1$$

Handling Non-linearly Separable Data:

In scenarios where data is not linearly separable, slack variables ε_i are introduced, allowing some points to violate the margin constraints. The objective function thus becomes:

$$min_{\omega,b}\ \frac{1}{2}||\omega||^2 + \ \boldsymbol{C}\sum_{i=1}^{n}\varepsilon_i$$

With constraints:

$$y_i(\omega.x_i + b) \geq 1 - \varepsilon_i$$

$$\varepsilon_i \geq 0$$

Here, $\boldsymbol{C}$ is a crucial regularization parameter that manages the trade-off between minimizing the training error and maximizing the margin. A higher $\boldsymbol{C}$ values aim for lower training errors at the risk of overfitting, while a lower $\boldsymbol{C}$ encourages a larger margin potentially at the expense of higher training errors.

Linear SVC aims to balance the margin width and classification accuracy by optimizing the weight vector and allowing some flexibility through slack variables. The parameter $\boldsymbol{C}$ plays a pivotal role in this balance, affecting the model's generalization capability. Support vectors

are crucial as they directly influence the positioning of the hyperplane, being the nearest points to the decision boundary.

Effect of OneVsRestClassifier on LinearSVC:

Introduction to OneVsRest Strategy:

The `OneVsRestClassifier` is a strategy for handling multi-label classification problems. In the context of machine learning, multi-label classification involves predicting multiple response variables for each observation, which is a common requirement in domains like text classification where an item can simultaneously belong to multiple categories.

Mechanism:

The `OneVsRestClassifier` (also known as One-vs-All) works by fitting one classifier per class. For each classifier, the class it represents is treated as the positive class and all other classes are combined into a single negative class. This method transforms a multi-label classification problem into multiple binary classification problems, thereby extending the applicability of binary classifiers like `LinearSVC`.

Application to LinearSVC:

`LinearSVC` is typically used for binary classification tasks. By integrating `LinearSVC` with `OneVsRestClassifier`, we enhance the classifier to handle multi-label datasets effectively. Each instance of `LinearSVC` within the `OneVsRestClassifier` framework independently predicts the presence or absence of a class. The ensemble of these binary classifiers then allows for the possibility of an observation being tagged with multiple labels, adapting `LinearSVC` to scenarios it would not be able to manage on its own.

Advantages:

1. Simplicity: OneVsRest is straightforward to implement and understand, making it a popular choice for multi-label classification.

2. Scalability: It scales well to large numbers of classes without requiring extensive modification to the underlying binary classification algorithm.

3. Independence: Classifiers for each label are independent, which simplifies the computational complexity and allows for parallel computation.

Limitations:

1. Imbalance Handling: Each classifier deals with a highly imbalanced distribution of the positive and negative classes, especially if the dataset has a skewed class distribution.

2. Correlation Ignorance: The method does not account for correlations between labels since each label is fitted independently.

3. Increased Model Complexity: Fitting multiple classifiers means a linear increase in the number of models that need to be trained, which can increase the computational cost and time, especially with a large number of labels.

Algorithmic Description

The Pseudocode below explains the process done from importing libraries, preprocessing data, vectorization of the comments, using NLP to extract features like tokens, non-stop tokens, lemmas, number of tokens, sentences, punctuations, and GloVe vectors.

```
BEGIN
  IMPORT necessary libraries
  READ training data from CSV
```

READ test data from CSV

PREPROCESS data

Calculate derived columns like 'non-toxic', 'toxicity_type_defined', 'toxic_undefined', and 'soft_toxic'

PERFORM initial data analysis

Compute label counts

Display bar plot of label counts

Generate and display correlation heatmap of target columns

DEFINE additional columns for analysis

BALANCE data according to 'non-toxic' distribution

SHUFFLE balanced data

INITIATE SpaCy natural language processing

PREPROCESS comments to extract features like tokens, non-stop tokens, lemmas, number of tokens, sentences, punctuations, and GloVe vectors

STORE preprocessed features back to DataFrame

DISPLAY histogram of token counts

CALCULATE and display median sentence count for each label

CALCULATE and display median token count for each label

NORMALIZE token counts logarithmically

IDENTIFY soft toxic comments in balanced data

PLOT word frequency analysis for each label

SPLIT data into training and validation sets

INITIALIZE TF-IDF vectorizer

FIT vectorizer on training comment text

TRANSFORM comment text to TF-IDF vectors

Logistic Regression Pseudocode

```
BEGIN
    IMPORT LogisticRegression, OneVsRestClassifier

    INITIALIZE LogisticRegression model named lr_classifier

    WRAP lr_classifier with OneVsRestClassifier named ovr_classifier

    PREPARE training data:
        Set lr_X_train to first part of 'glove_vector' list up to val_border
        Set lr_Y_train to target labels corresponding to lr_X_train

    PREPARE validation data:
        Set lr_X_val to 'glove_vector' list from val_border to end
        Set lr_y_val to target labels corresponding to lr_X_val

    FIT ovr_classifier on lr_X_train and lr_Y_train

    PREDICTIONS can be made using ovr_classifier on validation data lr_X_val
    EVALUATE the model using appropriate metrics such as accuracy, F1-score

END
```

KNN Pseudocode

```
BEGIN
    IMPORT KNeighborsClassifier, OneVsRestClassifier

    INITIALIZE KNeighborsClassifier with n_neighbors set to 3 as knn_classifier
```

```
    WRAP knn_classifier with OneVsRestClassifier named ovr_knn

    PREPARE training data:
      Set knn_X_train to first part of 'glove_vector' list up to val_border
      Set knn_Y_train to target labels corresponding to knn_X_train

    PREPARE validation data:
      Set knn_X_val to 'glove_vector' list from val_border to end
      Set knn_y_val to target labels corresponding to knn_X_val

    FIT ovr_knn on knn_X_train and knn_Y_train

    PREDICT using ovr_knn on validation data knn_X_val
    STORE predictions in y_pred_knn

    (Optional) EVALUATE model performance using y_pred_knn and knn_y_val
      Possible metrics: accuracy, precision, recall, F1 score

  END
```

Linear SVC Pseudocode

```
    SETUP machine learning models (SVM)
  TRAIN models using OneVsRestClassifier with LinearSVC for multi-label classification

  PREDICT on validation set using SVM
  CALCULATE and display classification metrics (Precision, Recall, F1 Score, Accuracy, ROC AUC)
END
```

Experimental Results

What we will attempt to resolve:

Our project aims to resolve a multi-label classification problem within the natural language processing domain. Specifically, to develop machine learning models capable of accurately classifying text comments by their level of toxicity. This classification was to identify multiple categories of toxic behavior, including general toxicity, severe toxicity, obscene language, threats, insults, and identity hate speech. Each comment within the dataset was tagged with one or more of these labels, necessitating a model capable of predicting multiple binary outcomes. The objective was to contribute to automated content moderation by developing a model that could serve as a tool for maintaining the integrity and constructive nature of discussion on digital platforms (e.g., Facebook, Instagram, X/Twitter, etc.).

Toxic comments can undermine the sense of community and safety on online platforms, deterring user engagement and contributing to negative experiences. An automated tool to efficiently and accurately classify the toxicity level of comments can enhance moderation efforts, creating a safer online environment.

Statistical Overview:

Missing Values: There are no missing values in the dataset, as preprocessing ensured the completeness of all records.

Class Distribution: The dataset is imbalanced with a higher proportion of non-toxic comments. Approximately 10% of the comments are labeled as toxic in one or more categories.

Statistical Overview – The dataset in question comprises 159,571 comments sourced from Wikipedia Talk pages. These comments have been classified and labeled based on their toxicity, with five distinct categories used to sub-classify them. It should be noted, however,

that a substantial proportion of the comments are classified as non-toxic, leading to a significant class imbalance within the dataset.

Word Count and Comment Length – The average word count per comment is approximately 68 words, with a median of 42, suggesting moderate detail in most discussions. Comments labeled as toxic tend to be longer, averaging around 78 words, possibly reflecting longer thoughts expressing negativity. The distribution of comment lengths shows a right-skewed pattern, typical of online text data, where most comments are relatively short, but a few are lengthy.

Common Words and Phrases – Analyzing the frequency of words and phrases provides insights into the nature of the discussion within each category:

Non-Toxic Comments: Common terms include 'article,' 'page,' 'please,' 'thank you,' and 'help,' which indicate neutral or collaborative expression

Toxic Comments: These frequently contain terms such as 'idiot,' 'stupid,' and profanities. Phrases often involve insults or aggression

Severe Toxic and Obscene: These comments use highly offensive language, which is less common in other categories

Word Clouds – Visual representations such as word clouds have been used to illustrate the prevalence of specific terms. The word clouds for non-toxic comments are polite and technical, reflecting Wikipedia’s purpose as an information-sharing platform. The clouds for toxic and severely toxic comments are full of derogatory and obscene terms, visually highlighting the differences in word use across categories.

Label Imbalance:

A significant challenge with the dataset is the imbalance among the labels. Most comments are non-toxic, leading to a disproportionate representation of the non-toxic class compared to toxic classes. Within the toxic labels, categories like 'threat' and 'identity hate' are notably less frequent than 'toxic' or 'obscene,' which introduces complexity in training classification models.

Data Acquisition:

The dataset, sourced from an online platform's comment sections, consists of numerous user-generated comments. These comments are labeled for various types of toxicity, such as general toxicity, severe toxicity, obscenities, threats, insults, and identity-based hate. The data was loaded from CSV files for both training and testing purposes.

Preprocessing and Initial Data Analysis:

The preprocessing steps involved generating derived columns to better define the nature of toxicity within the comments. These include:

- Non-Toxic: A derived binary label indicating the absence of any toxic traits.
- Toxicity Type Defined: A binary indicator summarizing whether any specific type of toxicity (insult, obscene, identity hate, threat) was present.
- Toxic Undefined: An indicator for comments labeled toxic but not fitting the defined types.
- Soft Toxic: Marks comments not labeled toxic despite having defined toxic traits.

Initial data analysis included computing and visualizing the counts of each label to understand the distribution within the dataset. A correlation heatmap was also generated to identify any relationships between different types of toxicity.

Balancing the Dataset:

Given the imbalance observed in the initial analysis, the dataset was balanced by adjusting the proportion of non-toxic comments relative to toxic comments. The balanced dataset was then shuffled to randomize the order of data points to prevent any order bias during model training. These methodologies could improve the sensitivity and specificity of the predictive models, improving their performance across all categories to help moderate online content.

Feature Engineering Using SpaCy:

Natural Language Processing (NLP) was employed to further preprocess and extract meaningful features from the comments:

Tokenization: Breaking down text into individual words or terms.

Non-Stop Tokens: Extracting words excluding common stop words.

Lemmatization: Reducing words to their base or root form.

Numerical Features: Counting the total number of tokens, sentences, and punctuation marks in each comment.

GloVe Vectors: Utilizing pretrained GloVe vectors to capture semantic meanings of words in a dense vector form.

These features were compiled into the main DataFrame for subsequent analyses and model training.

Visualization and Further Analysis:

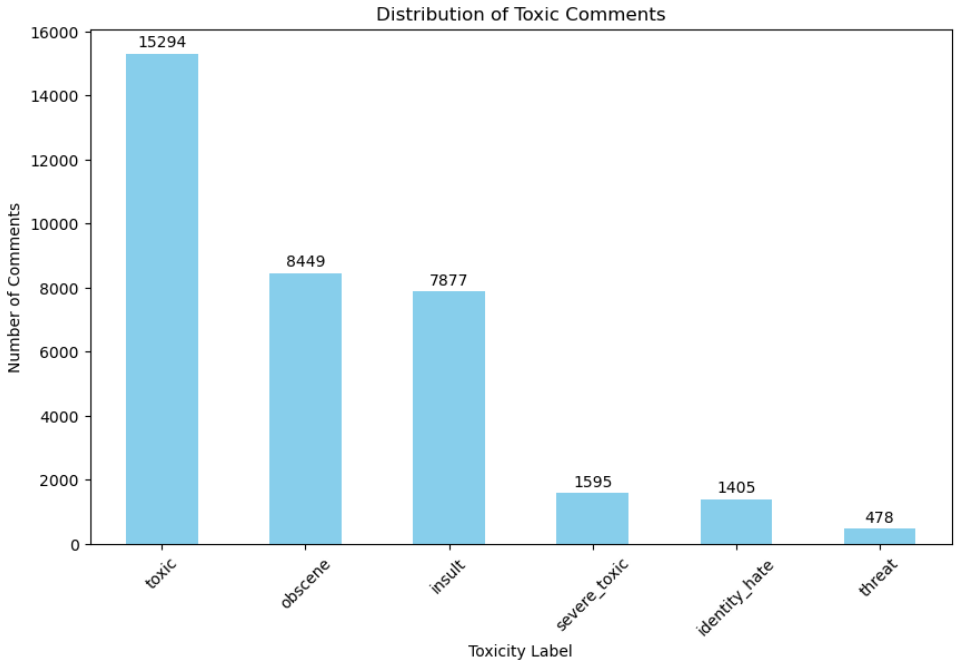
Visual explorations included histograms of token counts and bar plots to compare the median sentence and token counts across different labels. These visualizations helped in understanding the textual characteristics associated with each category of toxicity.

Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) is the cornerstone for understanding the dataset's structure, patterns, and characteristics. This section delves into the key insights gained from exploring the data.

Distribution of Toxicity Labels:

Understanding the distribution of toxicity labels is paramount for comprehending the prevalence of different types of toxicity in the dataset. A detailed analysis revealed that most comments are non-toxic, accounting for approximately 89% of the dataset. However, a smaller proportion exhibits various degrees of toxicity such as 'toxic' (9.58%), 'obscene' (5.29%), and 'insult' (4.99%). Interestingly, certain toxicity labels, such as 'severe_toxic' and 'threat', are relatively rare, constituting less than 1% of the dataset each.



Feature Engineering:

Feature Engineering plays a pivotal role in preparing the dataset for modelling. In our analysis, we explored the creation of new features derived from the existing data to enhance the performance of machine learning models. For instance, we generated additional columns to capture nuanced aspects of toxicity, such as the presence of soft toxicity (comments exhibiting toxicity characteristics without being explicitly labelled as such) and the classification of comments with undefined toxicity types. These engineered features provide valuable insights into the complexity and diversity of toxic language.

Model Training Preparation:

Training and Validation Split: The data was divided into training and validation sets to ensure the model could be objectively evaluated.

TF-IDF Vectorization: The text data was transformed using a TF-IDF vectorizer, preparing it for input into machine learning models. This step converts text into a numerical format that machine learning algorithms can process, emphasizing words that are more interesting, i.e., frequent in a few documents but rare in the corpus.

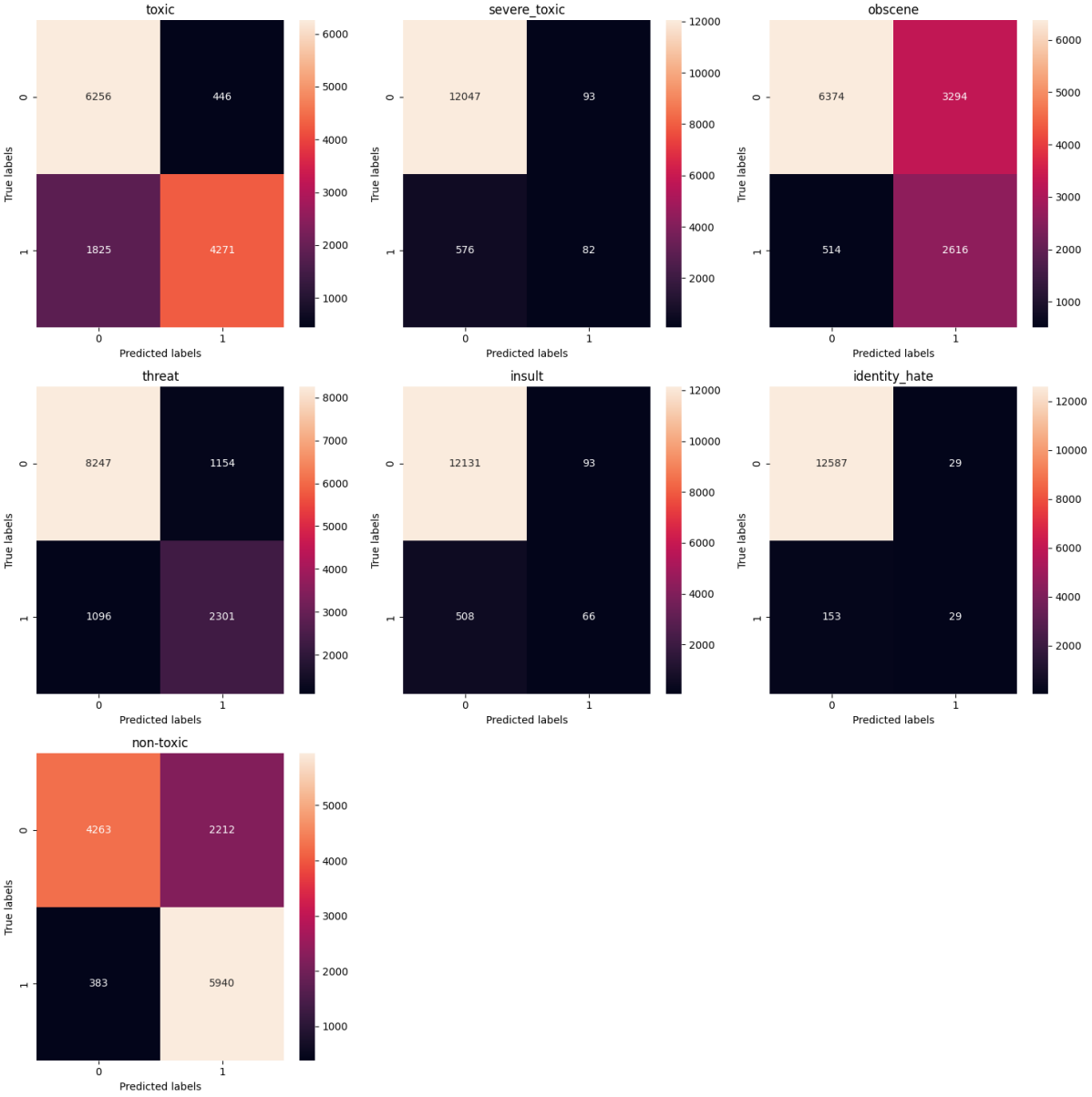
Results

SVC Model: Exhibits the highest precision especially for the 'toxic' category, indicating it is the most conservative model in predicting a comment as toxic. However, it may not capture as many toxic comments as the Logistic Regression model, which is reflected in its lower recall.

	precision	recall	f1-score	support
toxic	0.91	0.70	0.79	6096
severe_toxic	0.47	0.12	0.20	658
insult	0.44	0.84	0.58	3130
obscene	0.67	0.68	0.67	3397
identity_hate	0.42	0.11	0.18	574
threat	0.50	0.16	0.24	182
non-toxic	0.73	0.94	0.82	6323
micro avg	0.68	0.75	0.71	20360
macro avg	0.59	0.51	0.50	20360
weighted avg	0.71	0.75	0.71	20360
samples avg	0.69	0.78	0.71	20360

AUC: 0.690209896214827
Overall accuracy: 0.86

Confusion Matrix:



For 'toxic' and 'obscene' labels, the model demonstrates a substantial ability to recognize toxic content, as indicated by the higher counts in the TP sections.

The 'severe_toxic', 'threat', and 'identity_hate' categories show a tendency for the model to be conservative, resulting in a higher number of FNs, suggesting potential areas for improvement in sensitivity.

'Insult' and 'non-toxic' classifications show balanced performance with high counts of both TNs and TPs, reflecting the model's competence in these areas.

The number of FPs across most categories is contained, indicating the model's specificity is generally well-calibrated.

Logistic Regression Model: Demonstrates the best overall performance with the highest recall, F1-score, and AUC. It strikes a balance between correctly identifying toxic comments and maintaining precision. Its robust performance across various metrics suggests that it is a strong model for practical applications where both false positives and false negatives have significant implications.

Logistic Regression

	precision	recall	f1-score	support
toxic	0.87	0.82	0.85	6111
severe_toxic	0.48	0.20	0.29	632
insult	0.72	0.50	0.59	3154
obscene	0.78	0.61	0.69	3314
identity_hate	0.45	0.11	0.17	563
threat	0.41	0.21	0.28	197
non-toxic	0.86	0.90	0.88	6317
micro avg	0.82	0.72	0.77	20288
macro avg	0.65	0.48	0.53	20288
weighted avg	0.80	0.72	0.75	20288
samples avg	0.80	0.76	0.76	20288

AUC: 0.7107678394047582
Overall accuracy: 0.90

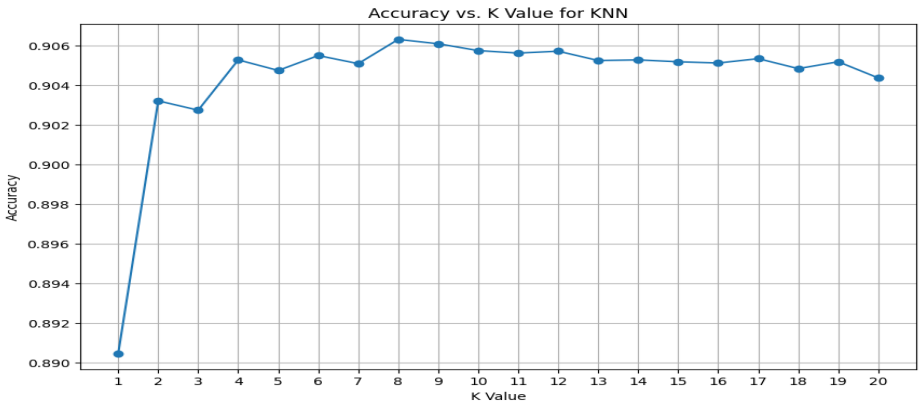
KNN Model: Shows reasonable performance in terms of accuracy and recall in 'toxic' but falls behind the other models in most metrics. KNN's lower macro and weighted averages indicate that it might struggle with class imbalance and the complex decision boundary intrinsic to textual data.

KNN

	precision	recall	f1-score	support
toxic	0.75	0.77	0.76	6111
severe_toxic	0.30	0.18	0.22	632
insult	0.54	0.50	0.52	3154
obscene	0.55	0.55	0.55	3314
identity_hate	0.20	0.07	0.11	563
threat	0.31	0.08	0.12	197
non-toxic	0.79	0.74	0.76	6317
micro avg	0.68	0.64	0.66	20288
macro avg	0.49	0.41	0.44	20288
weighted avg	0.66	0.64	0.65	20288
samples avg	0.68	0.67	0.65	20288

AUC: 0.6520337183732038
Overall accuracy: 0.85

The below graph shows the various accuracies for different K value.



The table below shows the overall comparison of the models:

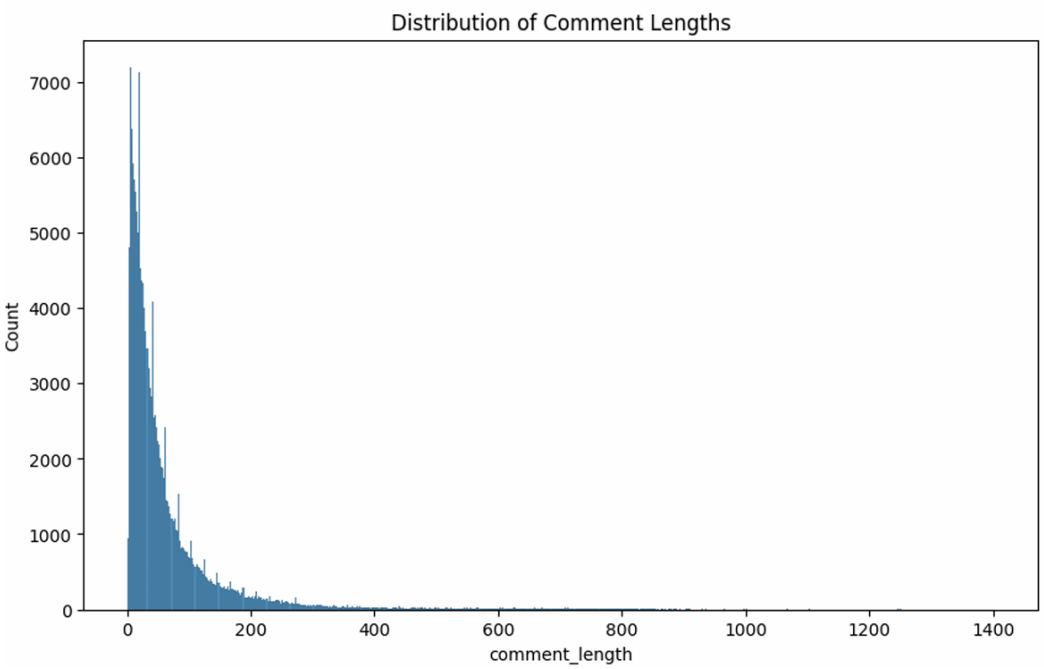
Metric	SVC Model	Logistic Regression Model	KNN Model
Precision	High, best in 'toxic'	Lower than SVC, but still high	Lowest of the three, fair in 'toxic'
Recall	Good in 'toxic', lower in other categories	Higher in 'toxic' and 'insult'	Higher in 'toxic', lower in other categories
F1-Score	High in 'toxic', lower in other categories	Highest in 'toxic', 'insult', and 'obscene'	Generally moderate, lower in 'identity_hate' and 'threat'
Support	Balanced class distribution	Balanced class distribution	Balanced class distribution
Micro Avg	Good performance	Best performance	Moderate performance
Macro Avg	Moderate performance	Best performance	Lower performance compared to other models
Weighted Avg	Good performance	Best performance	Least effective among the three
AUC	0.69	0.71	0.65
Accuracy	0.86	0.9	0.85

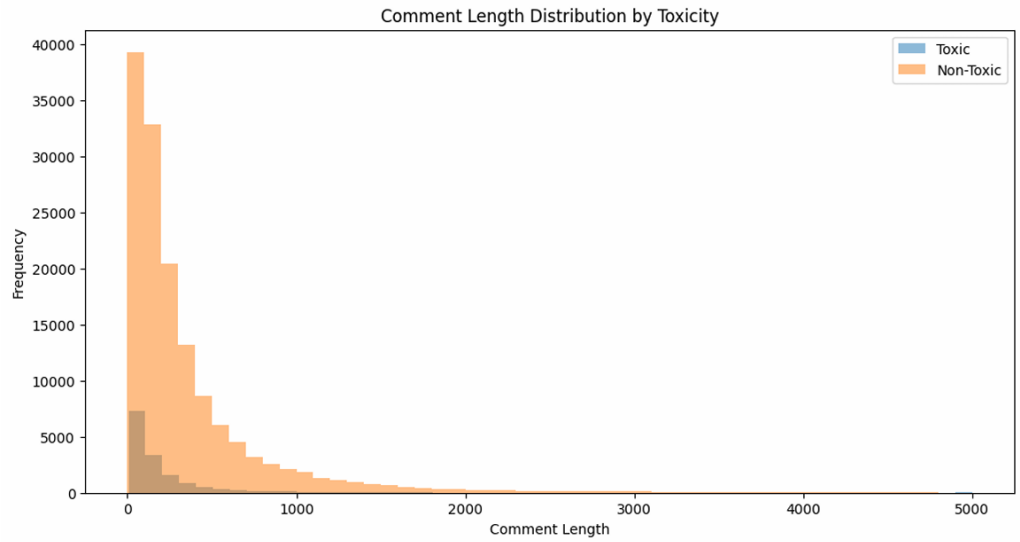
Text Analysis

Text analysis is a cornerstone for understanding the language used in toxic comments. We used TF-IDF (Term Frequency-Inverse Document Frequency) techniques to analyze the most common words and phrases associated with each toxicity category. This analysis revealed distinctive language patterns and provided valuable information for feature selection and model training. For example, words such as 'hate', 'kill', and 'stupid' were frequently associated with toxic comments, highlighting prevalent themes and sentiments.

Visualization of Comment Lengths

Visualizing the distribution of comment lengths provides insights into the structure and complexity of comments in the dataset. We plotted histograms and box plots to visualize the distribution of comment lengths across different toxicity categories. These visualizations highlighted differences in comment lengths between toxic and non-toxic comments, which could indicate different writing styles or content characteristics. Moreover, they underscored the importance of considering comment length as a potential feature in modelling tasks.

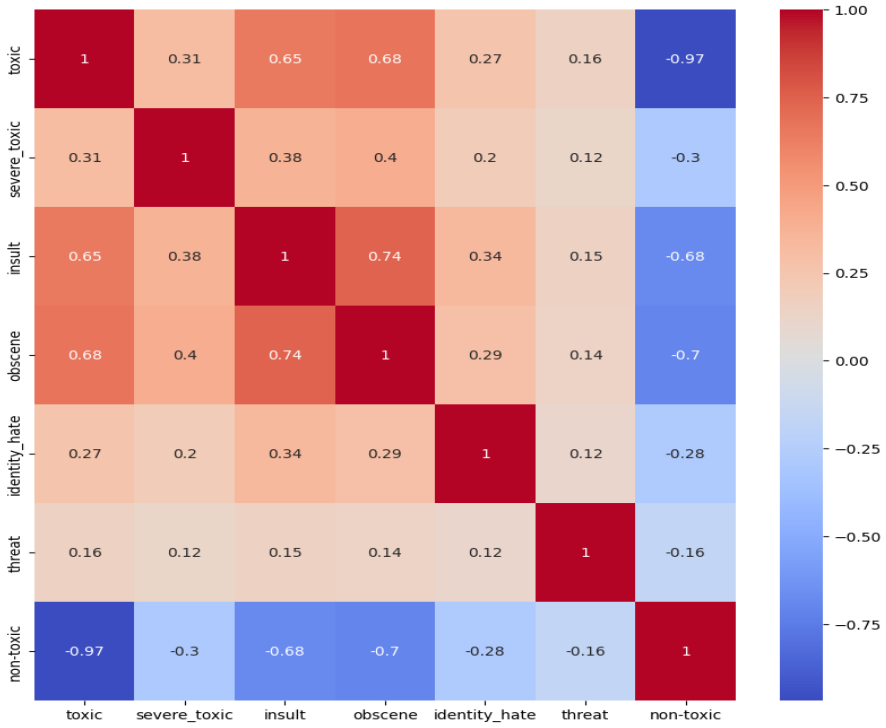




So, the toxic comment tends to be shorter than the nontoxic one.

Correlation Analysis

Understanding the relationships between toxicity labels can inform the design of multi-label classification models. We conducted correlation analysis to examine the correlations between pairs of toxicity labels. This analysis revealed both positive and negative correlations between certain toxicity labels, indicating potential dependencies and interactions between different types of toxicity. For instance, 'toxic' comments were found to exhibit strong positive



correlations with 'obscene' and 'insult', suggesting overlapping characteristics and themes.

Interpretation of Key Relationships

Positive Correlations:

'Toxic' and 'Insult' (0.65): There is a substantial positive correlation, suggesting that comments labeled as toxic are often also considered insulting.

'Insult' and 'Obscene' (0.74): This strong correlation indicates that obscene comments are frequently perceived as insulting.

'Toxic' and 'Obscene' (0.68): A high positive correlation reflects that many comments categorized as toxic contain obscene content.

Negative Correlations:

'Toxic' and 'Non-Toxic' (-0.97): An almost perfect negative correlation, as expected, since comments cannot be both toxic and non-toxic.

'Insult' and 'Non-Toxic' (-0.68), 'Obscene' and 'Non-Toxic' (-0.7): These strong negative correlations further validate the mutual exclusivity of these labels with the 'non-toxic' category.

Weak/No Correlations:

'Threat' shows weak correlations with all other toxic categories, suggesting that threats in comments are less frequently associated with other forms of toxicity.

'Identity_Hate' also shows relatively weak correlations with other toxic labels, which could imply that identity-based hate possesses distinct characteristics that do not always overlap with other toxicity types.

Discussion

Our team has used machine learning algorithms to identify and categorize toxic comments in online discussions. After analysing various models, we found that Logistic Regression is well-suited for linear separations in high-dimensional spaces. This was demonstrated by its exceptional performance metrics, which outperformed other models. On the other hand, KNN displayed lower F1 scores, indicating potential difficulties in handling sparse and imbalanced data. However, we believe these concerns can be addressed with more sophisticated sampling techniques.

In conclusion, we have determined that Logistic Regression is the most effective model for detecting and categorizing toxic comments. Its dependable performance metrics hold great promise for future applications. Moving forward, we will focus on integrating deep learning techniques to enhance the accuracy of our models even further. Additionally, we will explore cost-sensitive algorithms to address the challenges of rare toxic categories. Our goal is to leverage the potential of machine learning to improve the quality of online discussions and foster a safer and more positive online environment.

References

Bishop, C. M. (2006). "Pattern Recognition and Machine Learning." Springer, New York.

Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks." Machine Learning, 20(3), 273-297.

Cristianini, N., & Shawe-Taylor, J. (2000). "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods." Cambridge University Press.

Davidson, T., et al. "Automated Hate Speech Detection and the Problem of Offensive Language." Proceedings of ICLR, 2017.

Devlin, J., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL, 2018.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." Springer Series in Statistics, Springer New York.

Scholkopf, B., & Smola, A. J. (2002). "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond." MIT Press.

Wikipedia Talk Labels: Toxicity. (2017). Toxic Comment Classification Challenge [Data set]. Kaggle. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge