ToxiGuard: An Ensemble TF-IDF Framework for Multi-label Toxic Comment Filtering

Zefeng Liu*, Fei Pan
Lingnan University, Hong Kong 999077, China
*Corresponding email: zefengliu@ln.hk

Abstract

Background: Online discussion platforms face persistent risks from toxic and abusive comments, which can degrade community health and require costly manual moderation. In practice, toxicity detection is challenging due to noisy user-generated text, severe class imbalance, and the multi-label nature of harmful behaviors (e.g., insults, threats, and identity attacks may co-occur). This study attempts to build a practical and reproducible toxicity filtering pipeline that remains effective under limited computational budgets. Methods: We propose ToxiGuard, a lightweight multi-label classification framework that combines text normalization, unigram/bigram TF-IDF feature extraction, and heterogeneous classical learners (e.g., Logistic Regression, LinearSVC, Multinomial Naïve Bayes, and gradient-boosted trees). Model selection is performed via 10-fold cross-validation using ROC-AUC as the primary criterion, and an ensemble voting strategy is adopted to improve robustness across labels. For fair evaluation, instances with missing ground-truth labels are excluded from scoring when applicable. Results: Experiments on a large-scale Wikipedia discussion dataset demonstrate that the proposed ensemble consistently outperforms individual base learners across toxicity categories, yielding more stable AUC performance under label imbalance and text noise. Conclusion: ToxiGuard provides an efficient, interpretable, and deployment-friendly baseline for multi-label toxic comment filtering and can serve as a strong foundation for subsequent enhancements such as threshold calibration, cost-sensitive learning, and neural text encoders.

Keywords

Toxic comment filtering, Multi-label classification, Ensemble learning, Voting

Introduction

The health of online discussion communities is increasingly threatened by toxic and abusive comments, which can reduce user participation, distort public discourse, and impose substantial operational burdens on platform moderation teams [1]. In large-scale forums, the volume of user-generated content often exceeds the capacity of manual review, while delayed interventions allow harmful content to propagate and amplify [2]. Consequently, it is necessary to develop automated techniques that can assist moderators in filtering toxic comments more efficiently while reducing the risk of overlooking high-severity cases [3]. This motivation is typically conveyed early and concretely by presenting representative examples of the target problem in a figure, as the reference paper does when it introduces its task and immediately anchors the challenge with an illustrative figure.

In practice, toxic comment detection is challenging for several reasons. First, toxicity is rarely a single, isolated phenomenon: Insults, obscenity, threats, and identity attacks may co-occur, which naturally leads to a multi-label formulation rather than a simple binary decision [4]. Second, the label distribution is typically highly imbalanced. Severe categories occur infrequently but carry higher risk, making model training and thresholding sensitive to sampling noise. Third, user-generated text is noisy and adversarial: It contains misspellings, slang, code-switching, obfuscation, quotations, and context-dependent sarcasm that complicate feature extraction and robust generalization [5]. These issues collectively reduce model stability and can result in inconsistent performance across categories, especially when the goal is not only high average accuracy but also dependable behavior on minority

labels.

A variety of approaches have been explored for toxic content moderation, ranging from feature-based classical machine learning to neural encoders and large pre-trained language models. While neural methods can capture richer semantics, they may incur higher computational cost and operational complexity, and their performance can be sensitive to domain drift and calibration under changing community norms. For many deployment settings - such as resource-constrained systems, rapid iteration cycles, or governance pipelines requiring transparent decision traces - lightweight methods based on sparse lexical representations remain attractive [6]. TF-IDF features combined with linear classifiers are known to provide strong baselines for text classification, and heterogeneous ensembles can further reduce variance and improve robustness when individual learners exhibit complementary error patterns.

Motivated by these considerations, this study proposes ToxiGuard, a practical multi-label toxic comment filtering framework that combines: (1) text normalization tailored to noisy online discourse (2) TF-IDF feature extraction to capture discriminative lexical cues (3) a set of heterogeneous classical learners trained in a one-vs-rest manner for each toxicity label (4) an ensemble voting strategy to enhance stability across categories. Model selection is conducted via cross-validation using ROC-AUC as the primary criterion to better reflect ranking quality under label imbalance. The proposed pipeline is designed to be reproducible, computationally efficient, and suitable as a strong baseline or as the first-stage filter in a human-in-the-loop moderation workflow [7]. This "therefore, this study proposes…" transition and the immediate statement of the proposed solution mirrors the way the reference paper closes its problem framing and introduces its method.

The main contributions of this work are summarized as follows:

(1) We formulate a deployment-oriented pipeline for multi-label toxic comment filtering that explicitly addresses noisy text and class imbalance through robust evaluation and cross-validated model selection.

(2) We conduct a systematic comparison of heterogeneous classical learners under a unified TF-IDF representation and report performance across multiple toxicity categories using ROC-AUC as the principal metric.

(3) We demonstrate that a voting-based ensemble provides more consistent performance than single models across labels, offering an efficient and interpretable solution that can be readily integrated into moderation systems.

Related work

In toxic comment filtering, the practical goal is often to obtain a model that is not only accurate but also stable under noisy user-generated text, highly imbalanced labels, and frequent domain drift. While recent neural and pre-trained language models have shown strong performance, classical machine-learning approaches remain widely used in real-world moderation pipelines due to their favorable trade-offs in efficiency, interpretability, and ease of maintenance. In this work, we focus on the family of classical learners that are commonly paired with sparse lexical representations (e.g., TF-IDF), and we review several representative models that serve as strong baselines or complementary components in an ensemble.

Multinomial naïve bayes

Multinomial Naïve Bayes (MultinomialNB) is a generative classifier that models the class-conditional distribution of token counts. By assuming conditional independence among features given the class label, the model yields a closed-form solution with strong computational efficiency [8]. Although the independence assumption is clearly simplified for natural language, MultinomialNB often performs competitively in high-dimensional sparse text spaces because many toxicity-related cues are lexical and locally informative. In multi-label settings, it is commonly deployed in a one-vs-rest manner, which keeps training and inference straightforward while producing per-label scores that can be thresholded for final decisions.

Logistic Regression

Logistic Regression (LR) is a discriminative linear model that is frequently regarded as a strong baseline for text classification with TF-IDF features. With appropriate regularization (typically L2), LR provides a robust and well-behaved decision function in sparse feature spaces. A practical advantage is that LR naturally outputs probabilistic scores, which makes it convenient for threshold-based filtering policies and for later calibration

when different toxicity labels require different operating points [9]. In moderation scenarios where transparency is relevant, LR also offers a level of interpretability via feature weights, enabling qualitative inspection of which terms contribute most to each label.

LinearSVC

Linear Support Vector Classification (LinearSVC) optimizes a large-margin objective and has historically shown strong performance in high-dimensional text classification. Compared with LR, LinearSVC often provides competitive ranking and decision boundaries in sparse TF-IDF spaces, especially when classes are approximately linearly separable [10]. However, it does not produce calibrated probabilities by default. Therefore, when probability-like outputs are required (e.g., for soft voting or cost-sensitive thresholding), additional calibration steps may be needed. Like other classical learners, multi-label training is usually implemented through one-vs-rest decomposition.

AdaBoost

AdaBoost constructs an ensemble by iteratively focusing on samples that are misclassified by the current model, thereby combining many weak learners into a stronger predictor [11]. Conceptually, it is appealing for imbalanced and hard-to-classify cases, which are common in toxicity detection (e.g., severe toxic or threat labels). That said, in sparse text settings, AdaBoost can be sensitive to noise and may require careful control of weak learner complexity and iteration count to avoid overfitting. As a result, it is often evaluated as a complementary baseline rather than a default choice.

GBDT and XGBoost

Gradient Boosting Decision Trees (GBDT) extend the boosting idea by performing stage-wise additive modeling with gradient-based optimization of a chosen loss function. In contrast to linear models, tree-based boosting can capture non-linear feature interactions [12]. This can be beneficial when toxicity signals arise from combinations of lexical cues rather than individual tokens. Nevertheless, for extremely sparse and high-dimensional TF-IDF representations, tree-based methods can become computationally heavier and more sensitive to hyperparameters (e.g., tree depth, number of trees, learning rate).

XGBoost is a widely adopted implementation of gradient-boosted trees that integrates regularization and system-level optimizations. In practice, it is frequently used as a competitive boosting baseline and as a heterogeneous component in ensembles. For multi-label toxicity filtering, XGBoost can offer error patterns that differ from linear learners, which is particularly valuable when the goal is robustness across multiple toxicity categories with distinct lexical characteristics.

Voting Classifier and ensemble motivation

Beyond single models, ensemble learning is a pragmatic strategy to improve generalization by leveraging diversity among base learners. Voting Classifier is one of the simplest ensemble mechanisms: In hard voting, each model contributes a discrete prediction, and the final decision is obtained by majority vote. In soft voting, predicted probabilities are averaged before thresholding (when probability outputs are available). For multi-label problems, voting is naturally applied per label, which keeps the system modular and makes it easier to analyze label-specific behavior.

From a practical standpoint, voting ensembles are especially relevant for toxicity filtering because different labels may be dominated by different types of cues: Linear models may excel on lexically explicit insults, while boosted trees may better handle certain interaction patterns [13]. Therefore, combining heterogeneous learners can reduce variance and yield more consistent performance across categories. In this work, we adopt a voting-based combination of classical learners under a unified TF-IDF representation, aiming to obtain a lightweight yet reliable baseline suitable for deployment-oriented moderation settings.

Methods

This study focuses on the multi-label classification of toxic comments in Wikipedia talk-page discussions, where each comment is annotated across six binary categories: toxic, severe_toxic, obscene, threat, insult, and identity_hate [14]. As illustrated in the methodological overview in Figure 1, the proposed approach frames the problem as a series of parallel binary classification tasks following a one-versus-rest strategy. Specifically, an independent classifier is trained for each label, and the final system generates a corresponding set of six binary predictions for each input comment.

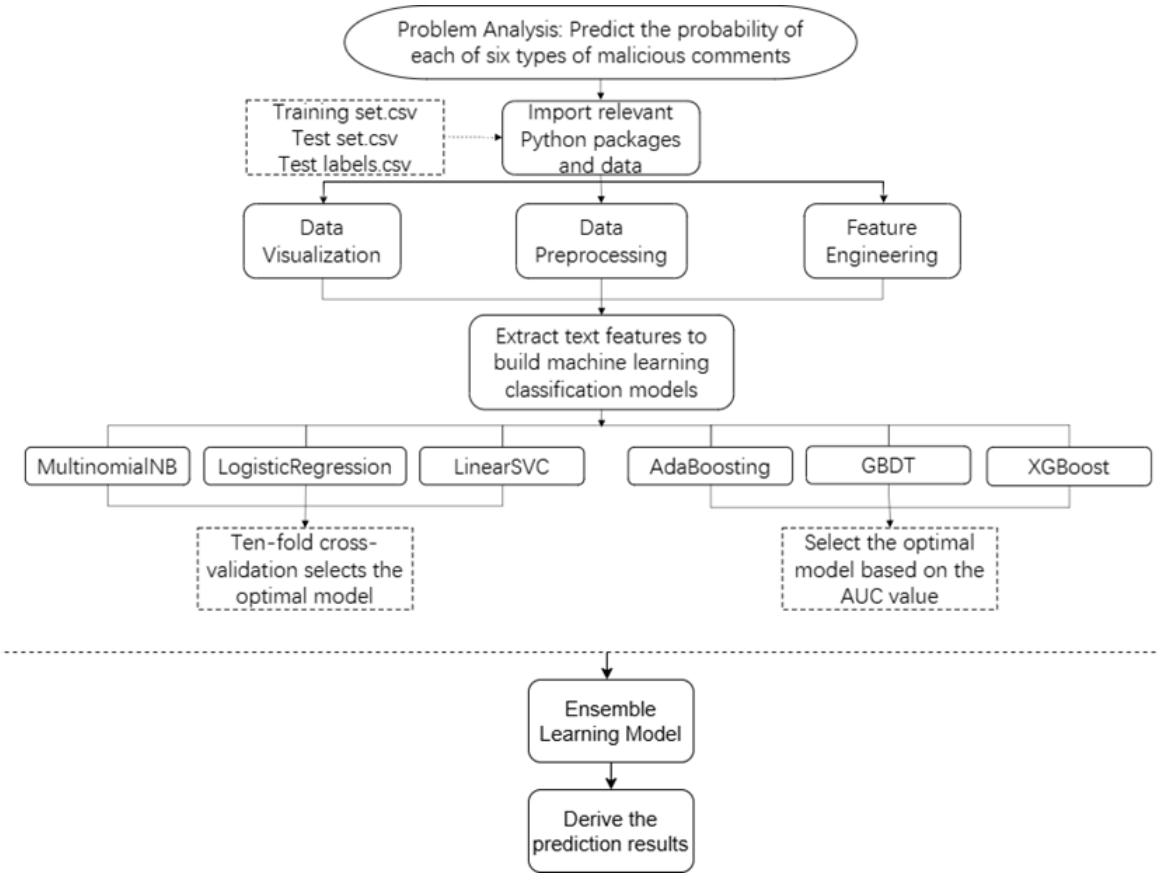

Figure 1. Method overview diagram.

Data loading and quality verification

We first load three files, namely train.csv, test.csv, and test_labels.csv, and conduct basic consistency checks to avoid downstream errors. Concretely, we inspect the first few records of each table (head ()), report dataset shapes (shape), and verify missing values via isnull (). sum (), with particular attention to the comment_text field that is later used for feature extraction.

A critical step concerns the evaluation constraint embedded in test_labels.csv: samples with label value -1 are not scored by the official evaluation protocol. Therefore, we explicitly quantify the number of scorable instances (e.g., by filtering via test_y[test_y.toxic != -1] as implemented) and enforce this masking rule consistently in all test-set metrics reported later. The style of sample -1 in the test label is shown in Table 1.

Table 1. The style of sample -1 in the test label.

Comment_-text	Toxic	Severe_toxic	Obscene	Threat	Insult	Identity_hate
Yo bitch Ja Rule is more succesful then you'll...	-1	-1	-1	-1	-1	-1
== From RfC == \n\n The title is fine as it is...	-1	-1	-1	-1	-1	-1
" \n\n == Sources == \n\n * Zawe Ashton on Lap...	-1	-1	-1	-1	-1	-1
:If you have a look back at the source, the in...	-1	-1	-1	-1	-1	-1
I don't anonymously edit articles at all.	-1	-1	-1	-1	-1	-1

Label distribution and multi-label structure analysis (EDA)

An exploratory analysis is conducted to address the two central challenges of this task: severe class imbalance and label co-occurrence. This step, implemented as per the notebook, precedes feature engineering. We first compute the number of positive samples per label by summing each of the six label columns, and visualize the

distribution using a bar chart, as shown in Figure 2. This analysis systematically quantifies data imbalance and motivates the choice of appropriate evaluation metrics that are robust under skewed labels.

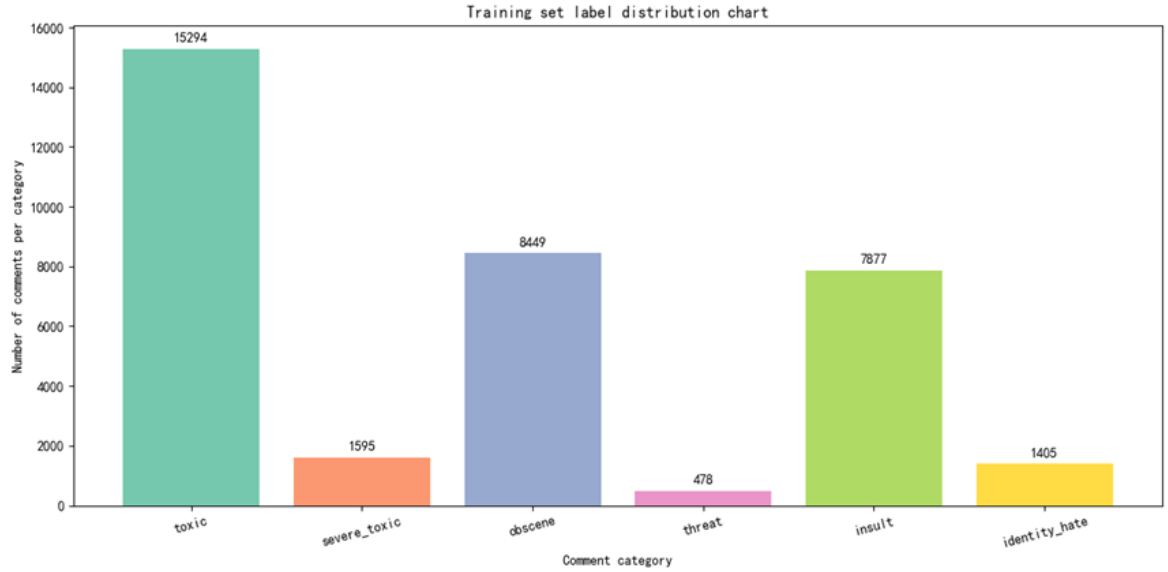

Figure 2. Training set label distribution chart number of comments per category and comment categories.

Next, we extract the six-label submatrix toxic_cols and compute the row sums for each sample label count. Based on these row sums, we explicitly define the "clean" subset as comments with negative values across all six toxicity categories via clean = (rowsums == 0). We report basic statistics including total comments, clean comments, toxic comments (non-clean comments), and the total number of positive labels assigned in the dataset. To further capture multi-label co-occurrence patterns, we compute the frequency of row sums and plot the distribution of label counts per comment, as shown in Figure 3. This clarifies whether toxic expression primarily manifests as single-label or frequently as multi-label patterns.

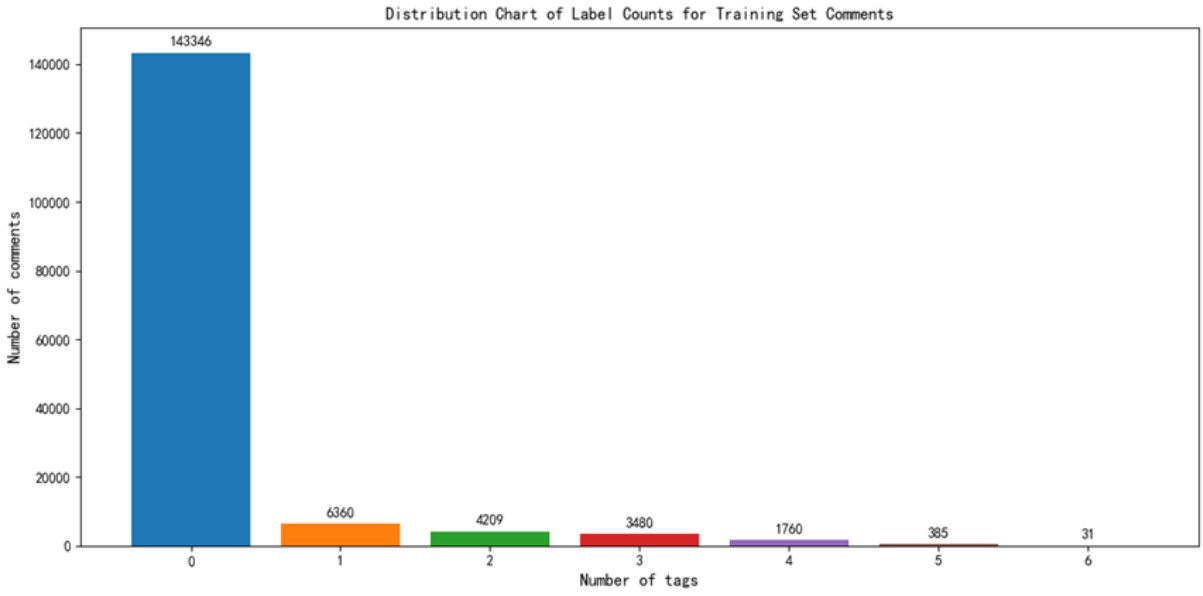

Figure 3. Distribution chart of label counts for training set comments.

Beyond marginal distributions, we examine label-level dependencies by computing the correlation matrix on the subset of samples with at least one positive label, and visualize it as a heatmap, as shown in Figure 4. This correlation analysis provides a principled view of co-occurrence patterns among toxicity types and later supports the interpretation of label-wise performance variability and the potential benefit of heterogeneous ensembles.

Text normalization and tokenization

To accommodate the noisy nature of user-generated comments, we implement a custom tokenizer

tokenize(text) and supply it directly to the TF-IDF vectorizer. The tokenizer follows a deterministic sequence of normalization steps [15]. First, all characters are converted to lowercase to reduce sparsity induced by case variation. Second, regular expressions are used to replace punctuation, digits, and control characters such as carriage returns, tabs, and newlines with whitespace, yielding a standardized token sequence. Third, the text is split by whitespace into tokens. Fourth, each token is encoded in ASCII with errors ignored, which removes non-ASCII artifacts and mitigates certain forms of noise caused by code-switching or special symbols. Fifth, tokens are lemmatized using WordNetLemmatizer to reduce morphological variants. Finally, tokens shorter than three characters are discarded to suppress low-information features [16]. This preprocessing reflects a relatively strict cleaning policy. In exchange for improved feature consistency and reduced dimensional noise, it may discard some informative non-ASCII signals. Given our primary objective of constructing an efficient and maintainable baseline, this reasonable trade-off is generally acceptable. Nevertheless, we later discuss its potential implications thoroughly in the Discussion section.

Figure 4. Correlation coefficient matrix heatmap.

TF-IDF feature construction and parameterization

Prior to full-scale vectorization, the notebook includes a small toy-corpus demonstration using both Count Vectorizer and TfidfVectorizer to verify the correctness of vocabulary construction and sparse-matrix outputs.

For the main experiments, we adopt TF-IDF to map each comment into a high-dimensional sparse vector space. Given a term tand a document d, the TF-IDF weight is defined as:

$$\mathrm{tfidf}(t,d) = \mathrm{tf}(t,d) \cdot \log \frac{N}{\mathrm{df}(t)+1},$$

where N denotes the total number of documents, and $\mathrm{df}(t)$ denotes the document frequency of term t. The implementation uses smoothing to avoid division-by-zero and overly large weights for rare terms.

In practice, we configure TfidfVectorizer with unigram features (ngram_range= (1,1)), word-level analysis (analyzer= ‘word’), the custom tokenizer defined in

Section 3.3, English stop-word removal (stop_words= 'english'), Unicode accent stripping (strip_accents= 'unicode'), and IDF weighting (use_idf=True). To suppress extremely rare features and reduce overfitting risk, we apply a minimum document-frequency threshold (min_df=10). The vectorizer is fitted on the training corpus to obtain X_{train}via fit_transform and then applied to the test corpus to obtain X_{test}via transform. The final feature dimensionality is reported to confirm successful construction of the feature space.

Base learners and 10-fold cross-validation comparison

Using the TF-IDF features, we first evaluate three classical baselines that are widely adopted for sparse text classification: MultinomialNB, Logistic Regression, and LinearSVC. Since the target is multi-label, we train one binary classifier per label (one-vs-rest) and report results for all six labels. To enable fair model comparisons, we implemented a 10-fold cross-validation program `cross_validation_score (classifier, X_train, y_train) `. This program calculates recall, F1 score, and ROC-AUC values for each label while maintaining consistent fold splits, reporting the median across all 10 folds. Results are stored in a structured table containing columns (Model, Label, Recall, F1, AUC), then concatenated across models to form a unified comparison table. We further visualize the AUC values for each label as grouped bar charts, as shown in Figure 5, enabling direct observation of the strengths and weaknesses of candidate learners across different labels.

Logistic Regression: $\hat{p}(y = 1 \mid x) = \sigma(w^{\top}x + b)$.

Linear SVM (hinge loss): $\min_{w} \frac{1}{2} \| w \|^2 + C \sum_i \max(0, 1 - y_i w^{\top} x_i)$

MultinomialNB posterior (log-form): $\log p(y \mid x) \propto \log p(y) + \sum_j x_j \log p(w_j \mid y)$.

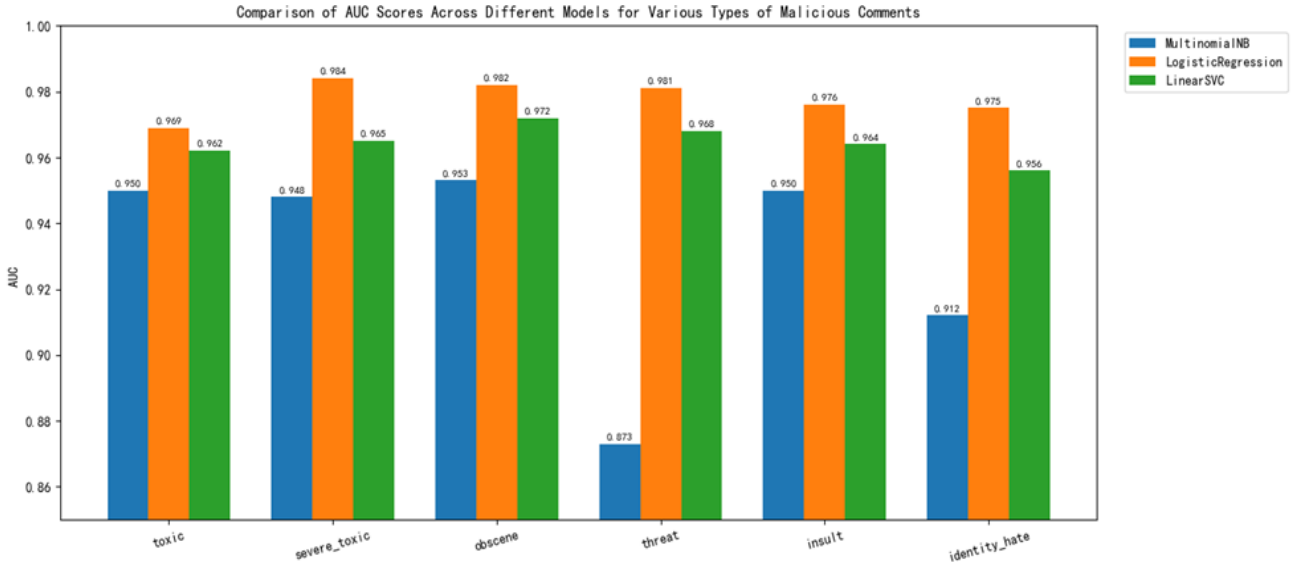

Figure 5. CV-AUC comparison bar chart.

Test-set evaluation with -1 masking and confusion-matrix diagnostics

To evaluate the model on the real test set, the Notebook defines the function score (classifier, X_train, y_train, X_test, y_test). This function sequentially fits the model on the full training set for each label and outputs binary predictions prediction(X_test) on the test set. Subsequently, -1 samples are masked strictly according to the rules in test_labels.csv: All metrics are calculated only on the countable subset where y≠-1. The implementation computes weighted Recall, weighted F1, and ROC-AUC, while also outputting a confusion matrix to observe the structure of false positives and false negatives for the "non-toxic/toxic" categories under this label. Results from three base learners (NB, LR, LinearSVC) are consolidated into a unified evaluation table to support intuitive and quantitative comparison. Box plots further illustrate the AUC distribution across six labels for different models, as shown in Figure 6. This enables a clear "distribution-based perspective" for comparing model stability and analyzing performance variability across different classification tasks. To introduce strong non linear modeling ability and enrich model diversity for later robust ensemble learning, we further evaluate three typical and widely used boosting based classifiers: AdaBoost Classifier, Gradient Boosting Classifier (GBDT), and XGB Classifier (XGBoost).

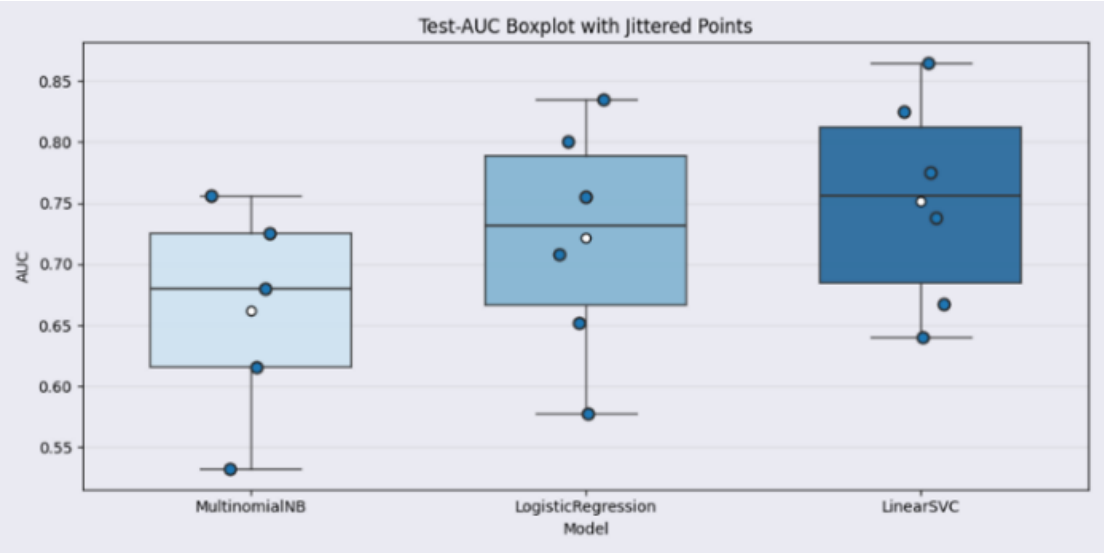

Figure 6. Test-AUC boxplot with jittered points.Boosting-based extensions: AdaBoost, GBDT, and XGBoost.

To incorporate learners with non-linear capacity and to increase model diversity for subsequent ensemble learning, we further evaluate three boosting-based classifiers: AdaBoost Classifier, Gradient Boosting Classifier (GBDT), and XGB Classifier (XGBoost). Each boosting model is trained and evaluated under the same label-wise one-vs-rest protocol and the same score () routine with -1 masking. The primary motivation here is not to replace linear baselines, but to provide complementary error patterns - particularly for labels whose lexical signals may interact in ways that a pure linear separator cannot capture [17]. We summarize per-label metrics and also compute grouped average AUC values to obtain an overall comparison among boosting methods.

Additive boosting model (GBDT): $F_m(x) = F_{m-1}(x) + \eta h_m(x)$.

AdaBoost reweighting (conceptual): $w_i^{(m+1)} \propto w_i^{(m)} \exp(\alpha_m \mathbb{I}[y_i \neq h_m(x_i)])$.

Compared to standard GBDT, XGBoost explicitly incorporates a tree complexity regularization term into its objective function and employs second-order Taylor expansion to introduce Hessian information. This approach enhances numerical stability and generalization performance during training while preserving the expressive power of the boosting trees. Given the significant noise and class imbalance in the TF-IDF sparse feature space for this task, XGBoost, as a nonlinear complementary learner, provides distinct error patterns from linear models, offering a source of diversity for subsequent heterogeneous voting ensembles.

XGBoost: $L = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{t=1}^{T} (\gamma |L_t| + \frac{\lambda}{2} \sum_{j \in L_t} w_{t,j}^2), \hat{y}_i = \sum_{t=1}^{T} f_t(x_i)$.

Heterogeneous voting ensemble and submission generation

Finally, we construct a heterogeneous ensemble using Voting Classifier with hard voting, combining Logistic Regression, LinearSVC, and XGBoost. This design follows the intuition that linear and boosted-tree learners may capture complementary aspects of toxicity-related lexical cues. To simultaneously perform evaluation and generate the final prediction file, the notebook defines score_predict (), which trains the ensemble per label, predicts on the test set, writes predictions into a unified predicted_df table (aligned by id), and computes Recall, F1-score, ROC-AUC, and confusion matrices under the same -1 masking protocol. The resulting predictions are exported to submission.csv with the required schema (id plus six label columns), enabling direct submission or integration into downstream moderation workflows.

For hard voting, the decision rule can be expressed as:

$$\hat{y} = \mathbb{I}\left(\sum_{k=1}^{K} \mathbb{I}(\hat{y}_k = 1) \geq \left\lceil \frac{K}{2} \right\rceil\right),$$

Experiments and results

This section reports the empirical performance of the proposed ToxiGuard pipeline on the Wikipedia toxic comment benchmark.

Experimental setup

We use the standard Wikipedia discussion dataset released for multi-label toxicity detection. Each sample

contains an identifier id, a raw comment field comment_text, and up to six binary toxicity annotations (toxic, severe_toxic, obscene, threat, insult, identity_hate). Following the notebook, the learning problem is implemented as six independent one-vs-rest binary classification tasks, one per label, and the final system outputs a six-dimensional label vector per comment.

Text is normalized and tokenized using the deterministic tokenizer described in Section 3 and then transformed into a sparse TF-IDF representation. The TF-IDF extractor is configured with unigram features, English stop-word removal, Unicode accent stripping, and a minimum document frequency threshold (min_df=10) to reduce the influence of extremely rare tokens [18]. The same fitted vectorizer is applied to both training and test comments.

Two evaluation protocols are used, corresponding to two stages in the notebook. First, for model selection on the training data, we conduct 10-fold cross-validation for each label and report Recall, F1-score, and ROC-AUC. In this setting, roc_auc scoring leverages the estimator's internal decision scores (probabilities or margins), which aligns with standard AUC practice for imbalanced classification.

Second, we evaluate trained models on the provided test split using test_labels.csv. Importantly, this file contains samples with missing labels encoded as -1; consistent with the official rule, all metrics are computed only on the subset where the corresponding label is not -1. In the notebook's test-time evaluation function, ROC-AUC is computed using binary predictions (0/1) rather than continuous decision scores. Consequently, test AUC should be interpreted as a conservative diagnostic of discriminative behavior under hard decisions, rather than as a direct substitute for probability-based AUC.

Cross-validation results of classical baselines

We first compare three classical baselines commonly paired with sparse text representations: MultinomialNB, Logistic Regression, and LinearSVC. Table 4 reports detailed and reliable 10-fold cross-validation results for each label. Overall, Logistic Regression achieves the strongest and most consistent AUC across all six categories, reaching AUC values of 0.969 (toxic), 0.984 (severe_toxic), 0.982 (obscene), 0.981 (threat), 0.976 (insult), and 0.975 (identity_hate). LinearSVC follows closely, while MultinomialNB remains competitive on frequent labels but obviously degrades on the rarer "threat" category due to data imbalance.

Table 2. Reports 10-fold cross-validation results for each label.

ID	Model	Label	Recall	F1	AUC
0	MultinomialNB	Toxic	0.483	0.637	0.95
1	MultinomialNB	Severe_toxic	0.022	0.042	0.948
2	MultinomialNB	Obscene	0.469	0.622	0.953
3	MultinomialNB	Threat	0.000	0.000	0.873
4	MultinomialNB	Insult	0.367	0.511	0.95
5	MultinomialNB	Identity_hate	0.008	0.015	0.912
6	Logistic Regression	Toxic	0.611	0.731	0.969
7	Logistic Regression	Severe_toxic	0.256	0.352	0.984
8	Logistic Regression	Obscene	0.637	0.747	0.982
9	Logistic Regression	Threat	0.123	0.207	0.981
10	Logistic Regression	Insult	0.524	0.638	0.976
11	Logistic Regression	Identity_hate	0.201	0.31	0.975
12	LinearSVC	Toxic	0.681	0.759	0.962
13	LinearSVC	Severe_toxic	0.266	0.354	0.965
14	LinearSVC	Obscene	0.695	0.774	0.972
15	LinearSVC	Threat	0.22	0.321	0.968
16	LinearSVC	Insult	0.576	0.663	0.964

ID	Model	Label	Recall	F1	AUC
17	LinearSVC	Identity_hate	0.275	0.384	0.956

Test-set results under the -1-masking rule

We next evaluate the same baselines on the test split using the official masking rule. Table 4-2 summarizes per-label performance and the macro-average AUC by model. When AUC is computed from hard predictions (as in the notebook), LinearSVC attains the best overall mean AUC (0.7376), followed by Logistic Regression (0.7080) and MultinomialNB (0.6156). At the label level, LinearSVC yields AUC values of 0.8596 (toxic), 0.6646 (severe_toxic), 0.8255 (obscene), 0.6368 (threat), 0.7736 (insult), and 0.6654 (identity_hate). Logistic Regression performs competitively on frequent labels (e.g., toxic AUC 0.8348, obscene AUC 0.7988) but is weaker on rare categories such as threat (AUC 0.5779). MultinomialNB exhibits a pronounced collapse on rare labels, with AUC close to 0.5 for threat (0.5000) and identity_hate (0.5014), suggesting near-constant predictions under hard decision rules.

This behavior is fully consistent with what is observed in the confusion matrices: under severe class imbalance, a classifier may achieve high weighted Recall/F1 while still failing to meaningfully separate positives from negatives for rare labels, especially when thresholding produces mostly negative outputs [19]. Thus, the confusion matrix provides a detailed and granular breakdown of model performance, serving as an essential diagnostic tool that effectively complements aggregate evaluation metrics.

Table 3. Summarizes per-label performance and the macro-average AUC by model.

	Model	Label	Recall	F1	AUC
0	MultinomialNB	Toxic	0.9352	0.9309	0.7552
1	MultinomialNB	Severe_toxic	0.9944	0.9921	0.5312
2	MultinomialNB	Obscene	0.9630	0.9579	0.7247
3	MultinomialNB	Threat	0.9967	0.9951	0.5000
4	MultinomialNB	Insult	0.9602	0.9535	0.6810
5	MultinomialNB	Identity_hate	0.9889	0.9834	0.5014
6	Logistic Regression	Toxic	0.9357	0.9370	0.8348
7	Logistic Regression	Severe_toxic	0.9931	0.9928	0.6511
8	Logistic Regression	Obscene	0.9660	0.9643	0.7988
9	Logistic Regression	Threat	0.9965	0.9957	0.5779
10	Logistic Regression	Insult	0.9642	0.9612	0.7533
11	Logistic Regression	Identity_hate	0.9905	0.9884	0.6320
12	LinearSVC	Toxic	0.9250	0.9299	0.8596
13	LinearSVC	Severe_toxic	0.9930	0.9928	0.6646
14	LinearSVC	Obscene	0.9628	0.9627	0.8255
15	LinearSVC	Threat	0.9964	0.9960	0.6368
16	LinearSVC	Insult	0.9614	0.9599	0.7736
17	LinearSVC	Identity_hate	0.9905	0.9890	0.6654

Boosting models: AdaBoost, GBDT, and XGBoost

To introduce more sophisticated non-linear decision functions and further enhance the diversity of our model pool for the subsequent ensemble framework, we conduct a comprehensive evaluation of three representative boosting-based classifiers: AdaBoost Classifier, Gradient Boosting Classifier (GBDT), and XGB Classifier (XGBoost). A detailed breakdown of their predictive performance across all six toxicity labels, including key metrics such as Accuracy, Recall, F1-score, and AUC, is systematically presented in Table 4. This table not only consolidates the quantitative outcomes for each label but also allows for a granular comparison of how each boosting model effectively adapts to the unique challenges posed by different toxicity categories, from the more frequent "toxic" and "obscene" labels to the rarer "threat" and "identity_hate" categories.

Table 4. Boosting model test set results.

ID	Model	Accuracy	Recall	F1	AUC
0	AdaBoost Classifier	Toxic	0.9243	0.9237	0.7718
1	AdaBoost Classifier	Severe_toxic	0.9928	0.9926	0.6659
2	AdaBoost Classifier	Obscene	0.9597	0.9566	0.7503
3	AdaBoost Classifier	Threat	0.9960	0.9957	0.6272
4	AdaBoost Classifier	Insult	0.9567	0.9493	0.6630
5	AdaBoost Classifier	Identity_hate	0.9892	0.9877	0.6460
6	Gradient Boosting Classifier	Toxic	0.9319	0.9265	0.7358
7	Gradient Boosting Classifier	Severe_toxic	0.9929	0.9922	0.6104
8	Gradient Boosting Classifier	Obscene	0.9584	0.9558	0.7531
9	Gradient Boosting Classifier	Threat	0.9959	0.9951	0.5468
10	Gradient Boosting Classifier	Insult	0.9603	0.9554	0.7061
11	Gradient Boosting Classifier	Identity_hate	0.9898	0.9880	0.6400
12	XGB Classifier	Toxic	0.9331	0.9345	0.8272
13	XGB Classifier	Severe_toxic	0.9934	0.9930	0.6540
14	XGB Classifier	Obscene	0.9621	0.9620	0.8221
15	XGB Classifier	Threat	0.9965	0.9961	0.6369
16	XGB Classifier	Insult	0.9626	0.9608	0.7708
17	XGB Classifier	Identity_hate	0.9908	0.9896	0.6843

Employing the identical rigorous test protocol and consistent 1-making rule applied in previous experiments, we calculate the mean AUC scores to facilitate a fair and direct comparison within the boosting algorithm family. The results clearly demonstrate that XGBoost achieves the highest mean AUC score of 0.7325, thereby establishing a superior overall performance benchmark compared to the other boosting methods. In contrast, the mean AUC values for AdaBoost and GBDT are observed to be 0.6874 and 0.6654, respectively. This significant performance hierarchy and the quantitative differences between the three models are visually summarized and clearly illustrated in the descending-order bar chart provided in Figure 7.

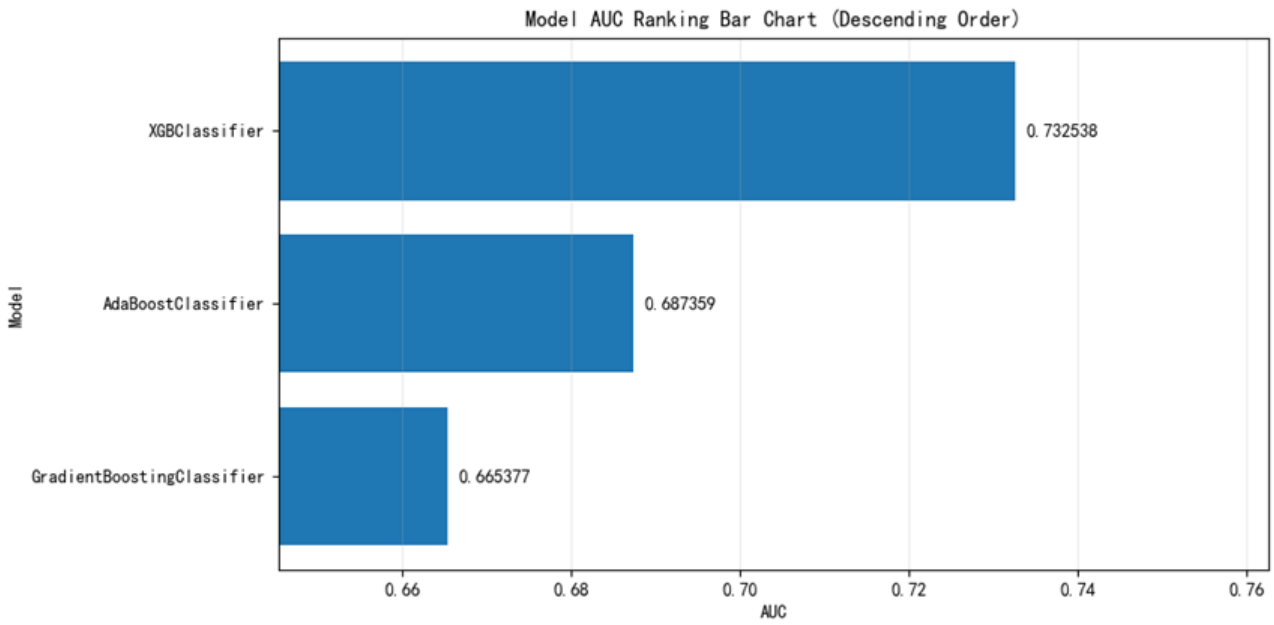

Figure 7. Model AUC ranking bar chart (descending order).

At the label level, XGBoost exhibits **significant and balanced improvements** across several critical and challenging categories, including the highly prevalent toxic (AUC 0.8272) and obscene (0.8221) labels, as well as the notoriously scarce threat (0.6369) and identity_hate (0.6843) categories. These prominent results strongly suggest that gradient-boosted decision trees can effectively complement traditional linear learners by successfully capturing intricate non-linear interaction patterns and complex latent dependencies embedded within sparse lexical features. Nevertheless, it is important to note that the overall performance gains remain highly label-dependent, and the magnitude of improvement is partially constrained by the inherent limitation of using hard class predictions rather than probability scores during the AUC computation process.

Heterogeneous voting ensemble

Finally, we construct a robust heterogeneous Voting Classifier employing the hard voting strategy, which strategically integrates the three top-performing base models identified in our experiments: Logistic Regression, LinearSVC, and XGBoost. Empirical results demonstrate that this ensemble framework achieves a competitive mean AUC score of 0.7211 on the reserved test set, representing a clear and measurable improvement of approximately 1.3% over the standalone Logistic Regression baseline (0.7080). However, while the ensemble strategy effectively leverages model diversity, its performance remains slightly below that of the strongest single model, LinearSVC (0.7376), and is marginally comparable to the high-performing XGBoost model (0.7325). A comprehensive summary of the Voting Classifier's quantitative performance across all six toxicity labels, including Accuracy, Recall, F1-score, and AUC metrics, is systematically presented in Table 4, and the relative distribution of these key performance indicators is visually visualized and summarized in the detailed metrics heatmap illustrated in Figure 8.

Table 4. Voting integration test set results.

ID	Model	Accuracy	Recall	F1	AUC
0	Voting Classifier	Toxic	0.9338	0.9359	0.8435
1	Voting Classifier	Severe_toxic	0.9931	0.9927	0.6525
2	Voting Classifier	Obscene	0.9652	0.9642	0.8140
3	Voting Classifier	Threat	0.9965	0.9959	0.6038
4	Voting Classifier	Insult	0.9643	0.9617	0.7619
5	Voting Classifier	Identity_hate	0.9908	0.9890	0.6510

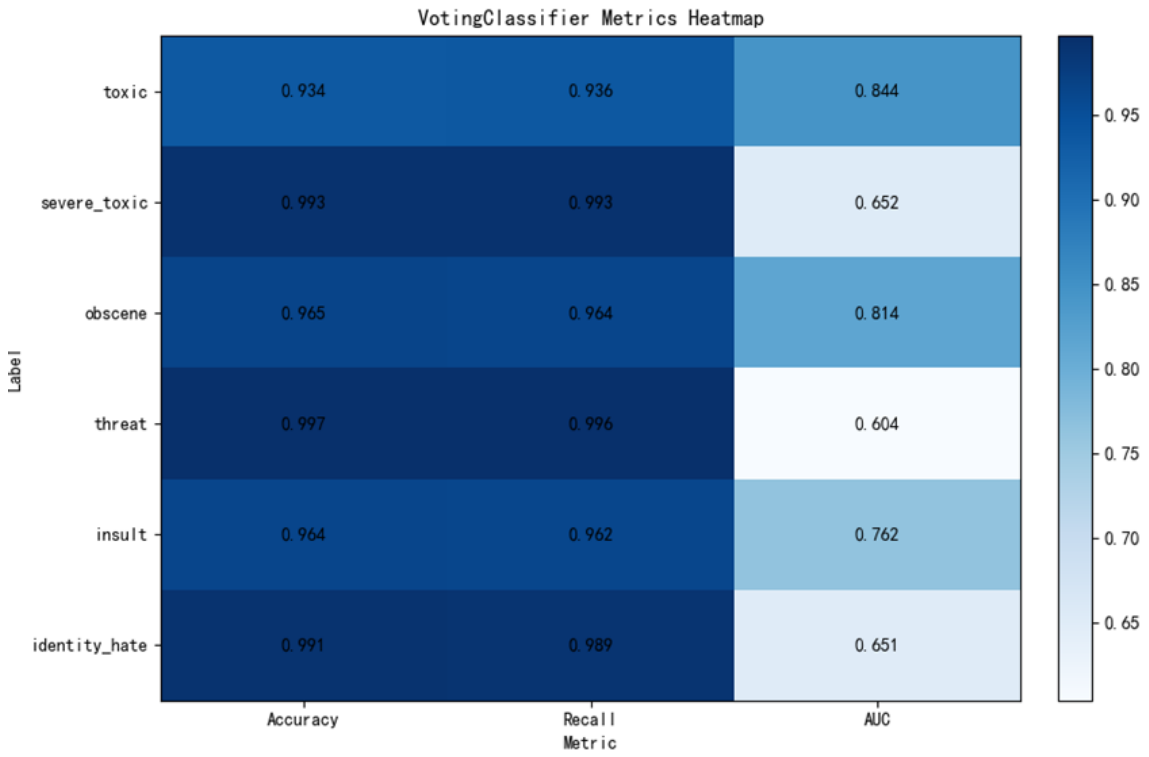

Figure 8. Voting integration test set results in metrics heatmap.

Importantly, the ensemble provides balanced performance across labels without the severe collapse observed for MultinomialNB on rare categories. For example, the ensemble reaches AUC 0.6038 on threat and 0.6510 on identity_hate, both higher than Logistic Regression on the same labels (0.5779 and 0.6320, respectively), though still below LinearSVC on threat (0.6368) and XGBoost on identity_hate (0.6843).

This pattern is consistent with the motivation of heterogeneous voting: when individual learners exhibit complementary error profiles, voting can mitigate worst-case behavior on difficult labels. However, because hard voting discards confidence information, it may not fully exploit the ranking quality reflected in probability-based AUC; this is a plausible explanation for why the ensemble does not consistently surpass the strongest single learner in the current evaluation setting

Summary of findings

Across evaluation stages, LogisticRegression is the strongest baseline under cross-validated probability-based AUC, while LinearSVC and XGBoost exhibit strong robustness under the notebook's hard-decision test AUC evaluation, particularly on rare categories such as threat and identity_hate. The heterogeneous voting ensemble improves label-wise balance relative to single linear learners, but its gains are moderated by the use of hard voting and by the current AUC computation based on binary outputs. These results collectively support the central premise of ToxiGuard: A lightweight TF-IDF pipeline with carefully chosen classical learners - and, where appropriate, heterogeneous ensembling - can deliver reliable multi-label toxicity filtering without requiring heavy neural infrastructure.

Discussion

This study suggests that a TF-IDF-based pipeline with classical machine-learning models can still serve as a cost-effective and deployment-friendly baseline for multi-label toxic comment filtering. A key reason is that the task contains substantial discriminative lexical cues: With sparse high-dimensional representations, linear decision functions can separate toxic versus non-toxic patterns reliably for the majority of categories. Empirically, LinearSVC provides the most robust overall performance across the six labels (Macro-AUC = 0.737585), followed by LogisticRegression (0.707988). In contrast, MultinomialNB degrades markedly on rare categories (e.g., AUC $\approx$ 0.5 on threat and identity_hate), indicating that under severe imbalance, simplified generative assumptions combined with hard decisions

may fail to capture minority-class separability.

Importantly, evaluation should not rely on a single metric. In our test-time results, Recall and F1 are generally high, while AUC varies substantially on the rarer labels. This observation highlights a common pitfall in highly imbalanced multi-label settings: Appearing to "make few mistakes overall" does not necessarily imply effective discrimination for minority positives. Accordingly, we treat AUC as a primary comparative metric and recommend interpreting it together with confusion-matrix patterns to understand false-positive/false-negative structures. In addition, since the current notebook computes test AUC from binary (0/1) predictions, the reported AUC should be viewed as a diagnostic under hard-threshold decisions rather than a strict probability-ranking AUC. A more standard analysis would compute AUC from continuous scores (e.g., predict_proba or decision_function) and further apply calibration and threshold tuning.

With respect to ensembling, heterogeneous voting is mainly valuable for reducing worst-case behavior and improving cross-label consistency, yet hard voting discards confidence information and therefore may not consistently surpass the strongest single learner. This naturally motivates future work that preserves the lightweight, deployable nature of the framework while improving minority-label utility: (1) adopting soft voting with probability calibration, and (2) performing label-wise threshold optimization or cost-sensitive learning, particularly for rare categories such as threat and identity_hate. In addition, incorporating more robust lexical features (e.g., character-level n-grams) may improve resilience to spelling variants and adversarial obfuscation.

Conclusion

In this work, we present ToxiGuard, a lightweight and reproducible pipeline for multi-label toxic comment filtering on Wikipedia discussion data. The proposed approach integrates deterministic text normalization, TF-IDF feature extraction, and a set of classical machine-learning learners trained in a one-vs-rest manner, with optional heterogeneous ensembling via voting. Across six toxicity categories, our experiments demonstrate that classical linear models remain strong and reliable baselines under sparse lexical representations and that performance varies substantially across labels under severe class imbalance. LinearSVC achieves the most robust overall discrimination on the test set (Macro-AUC = 0.737585), while Logistic Regression provides competitive performance and MultinomialNB exhibits pronounced degradation on rare labels.

Funding

This work was not supported by any funds.

Acknowledgements

The authors would like to show sincere thanks to those techniques who have contributed to this research.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Yang, Y., Lv, H., Chen, N. (2023) A survey on ensemble learning under the era of deep learning. *Artificial Intelligence Review*, 56(6), 5545-5589.

[2] Sawicki, J., Ganzha, M., Paprzycki, M. (2023) The state of the art of natural language processing - a systematic automated review of NLP literature using NLP techniques. *Data Intelligence*, 5(3), 707-749.

[3] Jim, J. R., Talukder, M. A. R., Malakar, P., Kabir, M. M., Nur, K., Mridha, M. F. (2024) Recent advancements and challenges of NLP-based sentiment analysis: a state-of-the-art review. *Natural Language Processing Journal*, 6, 100059.

[4] Olujimi, P. A., Ade-Ibijola, A. (2023) NLP techniques for automating responses to customer queries: a systematic review. *Discover Artificial Intelligence*, 3(1), 20.

[5] Mienye, I. D., Sun, Y. (2022) A survey of ensemble learning: concepts, algorithms, applications, and prospects. *IEEE Access*, 10, 99129-99149.

[6] Rupapara, V., Rustam, F., Shahzad, H. F., Mehmood, A., Ashraf, I., Choi, G. S. (2021) Impact of SMOTE on imbalanced text features for toxic comments classification using RVVC model. *IEEE Access*, 9, 78621-78634.

[7] Cahyani, D. E., Patasik, I. (2021) Performance comparison of tf-idf and word2vec models for emotion text classification. *Bulletin of Electrical Engineering and Informatics*, 10(5), 2780-2788.

[8] Bailly, A., Blanc, C., Francis, É., Guillotin, T.,

Jamal, F., Wakim, B., Roy, P. (2022) Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models. *Computer Methods and Programs in Biomedicine*, 213, 106504.

[9] Gulati, K., Kumar, S. S., Boddu, R. S. K., Sarvakar, K., Sharma, D. K., Nomani, M. Z. M. (2022) Comparative analysis of machine learning-based classification models using sentiment classification of tweets related to COVID-19 pandemic. *Materials Today: Proceedings*, 51, 38-41.

[10] Salih, N., Ksantini, M., Hussein, N., Ben Halima, D., Abdul Razzaq, A., Ahmed, S. (2023) Prediction of ROP zones using deep learning algorithms and voting classifier technique. *International Journal of Computational Intelligence Systems*, 16(1), 86.

[11] Hupkes, D., Giulianelli, M., Dankers, V., Artetxe, M., Elazar, Y., Pimentel, T., Jin, Z. (2023) A taxonomy and review of generalization research in NLP. *Nature Machine Intelligence*, 5(10), 1161-1174.

[12] Goyal, S., Doddapaneni, S., Khapra, M. M., Ravindran, B. (2023) A survey of adversarial defenses and robustness in nlp. *ACM Computing Surveys*, 55(14s), 1-39.

[13] Dogra, V., Verma, S., Kavita, Chatterjee, P., Shafi, J., Choi, J., Ijaz, M. F. (2022) A complete process of text classification system using state - of - the - art NLP models. *Computational Intelligence and Neuroscience*, 2022(1), 1883698.

[14] Haque, R., Islam, N., Islam, M., Ahsan, M. M. (2022) A comparative analysis on suicidal ideation detection using NLP, machine, and deep learning. *Technologies*, 10(3), 57.

[15] Lauriola, I., Lavelli, A., Aiolli, F. (2022) An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470, 443-456.

[16] Li, Q., Wang, Y., Shao, Y., Li, L., Hao, H. (2023) A comparative study on the most effective machine learning model for blast loading prediction: From GBDT to Transformer. Engineering Structures, 276, 115310.

[17] Zini, J. E., Awad, M. (2022) On the explainability of natural language processing deep models. *ACM Computing Surveys*, 55(5), 1-31.

[18] Kastrati, Z., Dalipi, F., Imran, A. S., Pireva Nuci, K., Wani, M. A. (2021) Sentiment analysis of students' feedback with NLP and deep learning: A systematic mapping study. *Applied Sciences*, 11(9), 3986.

[19] Qorib, M., Oladunni, T., Denis, M., Ososanya, E., Cotae, P. (2023) Covid-19 vaccine hesitancy: text mining, sentiment analysis and machine learning on COVID-19 vaccination Twitter dataset. *Expert Systems with Applications*, 212, 118715.