TAMING TOXICITIY

Utilizing Machine Learning and Deep Learning for Online Content Moderation

Jaemin Han 2024

Abstract

The internet is a significant platform for public discourse.
Toxic behavior online poses challenges for healthy communication.

The Goal

Develop models to classify toxic comments into six specific categories: toxic, severe toxic, obscene, threat, insult, and identity hate.

(part of Kaggle's toxic comment classification challenge by google & jigsaw)

Problem

Goal 01

Build robust machine learning models to detect six types of toxic comments.

Utilize both traditional and deep learning approaches.

Goal 02

Evaluate and compare model performances.

Visualize model training and prediction results.

Data Collection

DATA SOURCE 01

Wikipedia’s talk page edits dataset from Kaggle's Toxic Comment Classification Challenge - a large number of Wikipedia comments which have been labeled by human raters for toxic behavior.

WHY USE DATA 02

The data is specially labeled to detect different types of toxic comments, rather than a binary approach.

(e.g. some platforms may be fine with profanity, but not with other types of toxic content).

PREPROCESSING

Phase 1

Removed any HTML tags, converted text to lowercase, removed special characters and digits.

Phase 2

Tokenized text and applied lemmatization to standardize words.

Phase 3

Removed common English stopwords.

EDA

Word Clouds

We created two sets of word clouds:
Basic Word Cloud:

- Visual representation of the most frequent words for each class.

Filtered Word Cloud:

- Excluded common words to ensure distinct word clouds for each class.

Sentiment Analysis

- Used TextBlob to compute sentiment polarity, which measures the emotional tone of text.
- Visualized sentiment distribution with boxplots to understand the emotional context of toxic comments.

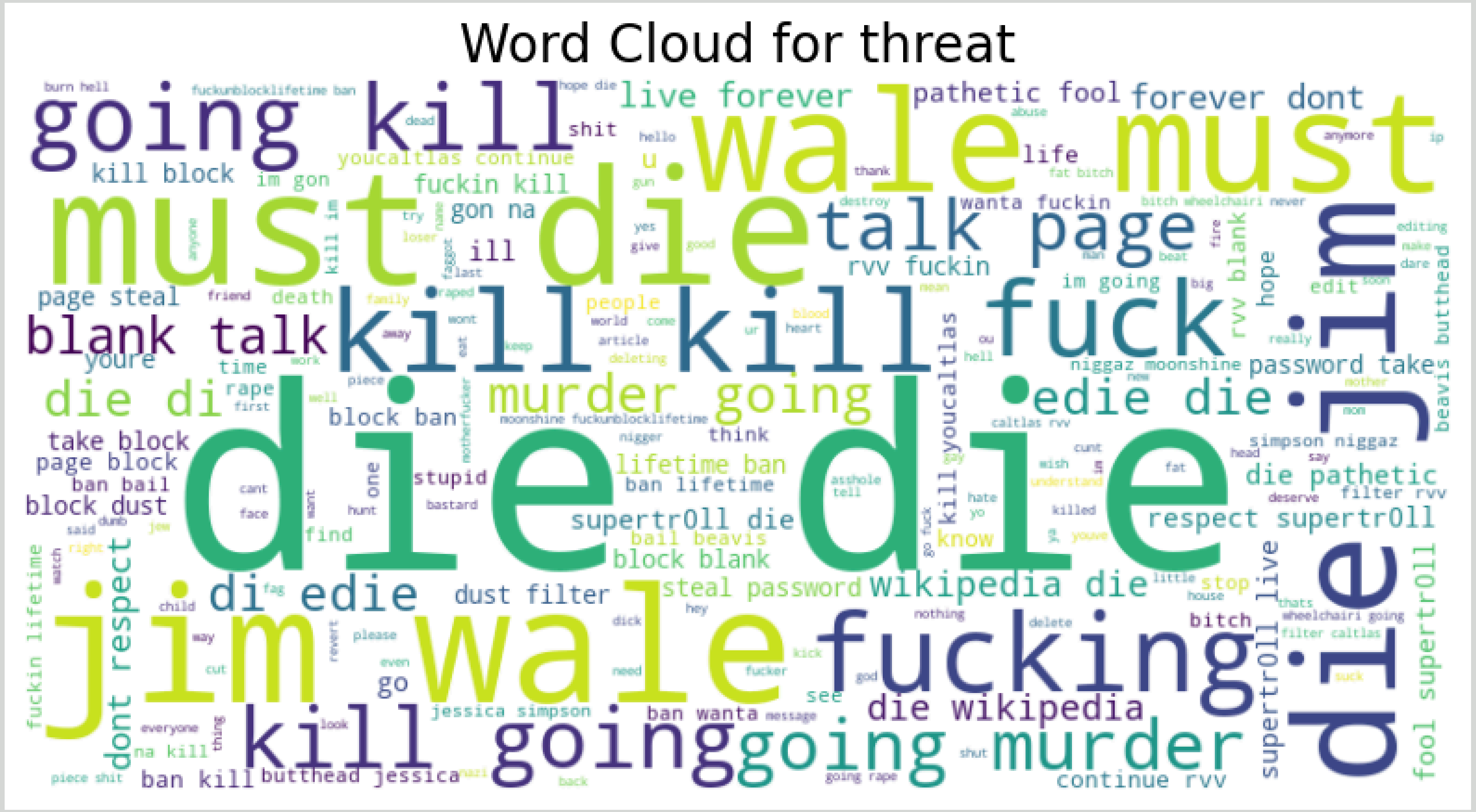
Toxic

Jaemin Han 2024

Severe Toxic

Obscene

Threat

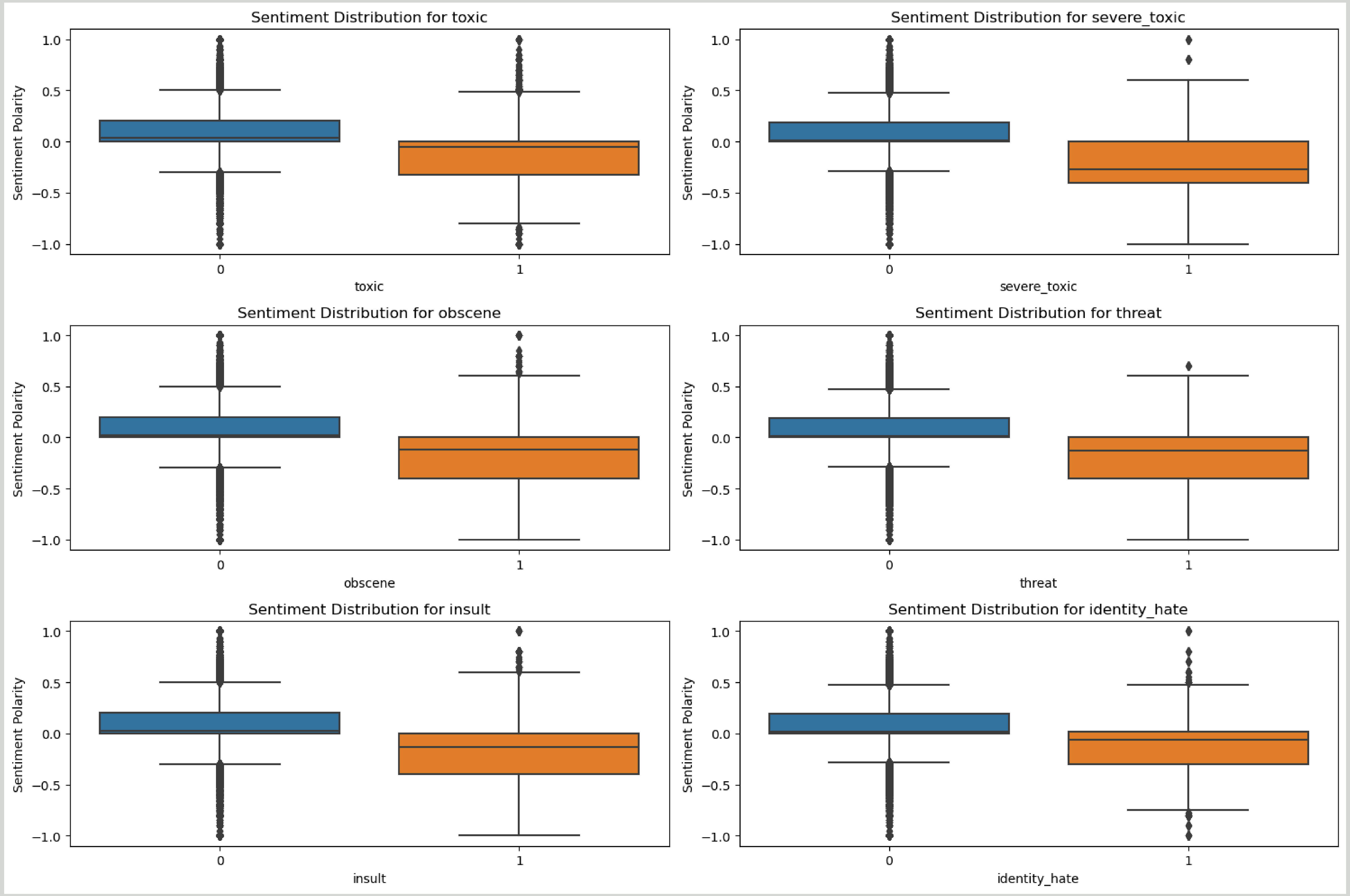


Jaemin Han 2024

Insult

Identity Hate

Sentiment Polarity Distributions



Jaemin Han 2024

ML Models

01 Logistic Regression

A simple and efficient model for binary classification, making it suitable for our task of predicting whether a comment is toxic or not. It serves as a strong baseline due to its interpretability and low computational cost.

02 XGBoost

XGBoost is a powerful gradient boosting model known for its high performance and efficiency. It is well-suited for handling structured data and provides excellent predictive accuracy for our multi-class toxic comment classification task.

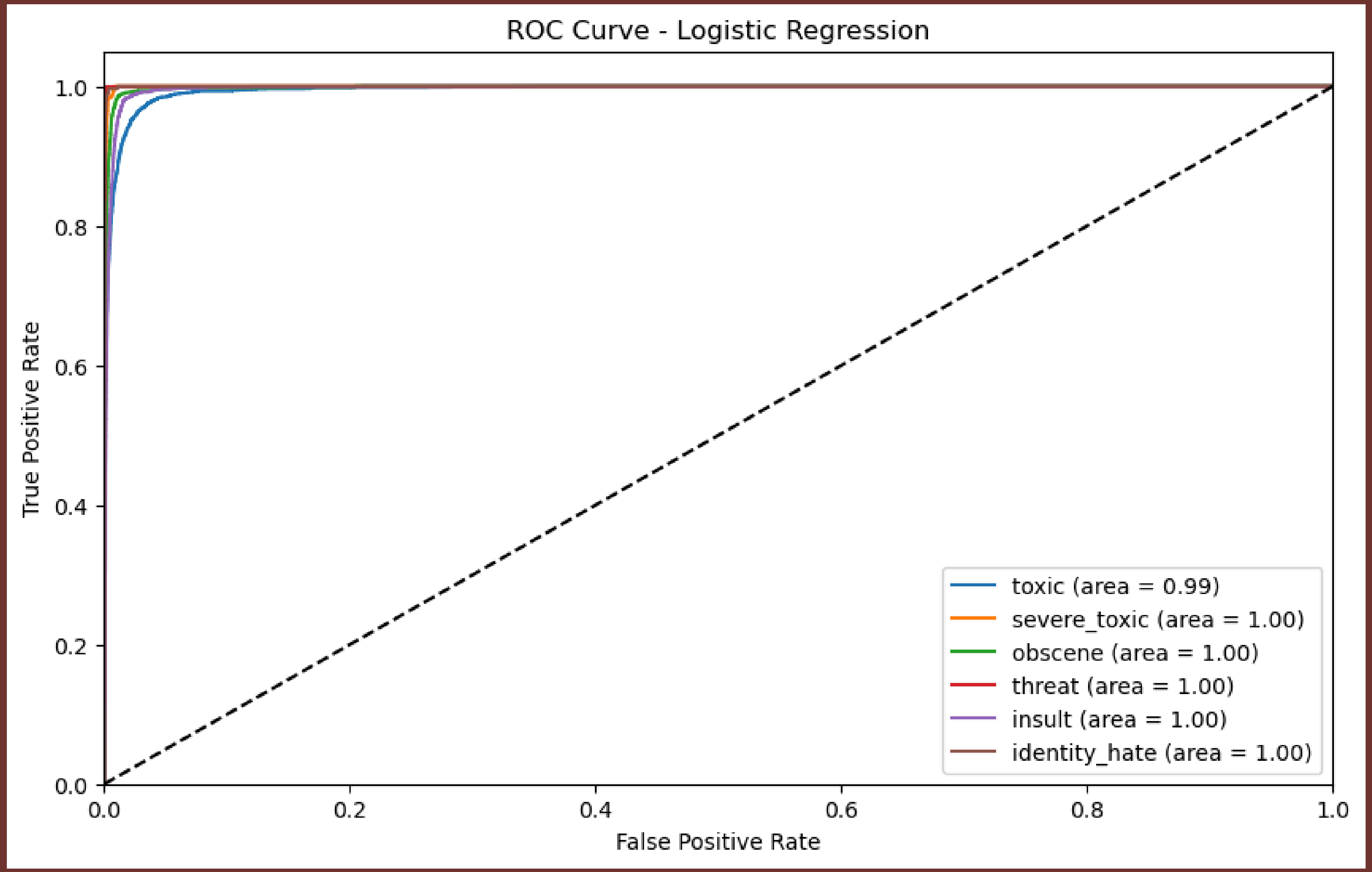
03 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve predictive performance and control overfitting. It is robust and provides feature importance scores, making it valuable for understanding which words contribute most to toxicity.

Logistic Regression

What is Logistic Regression Doing?

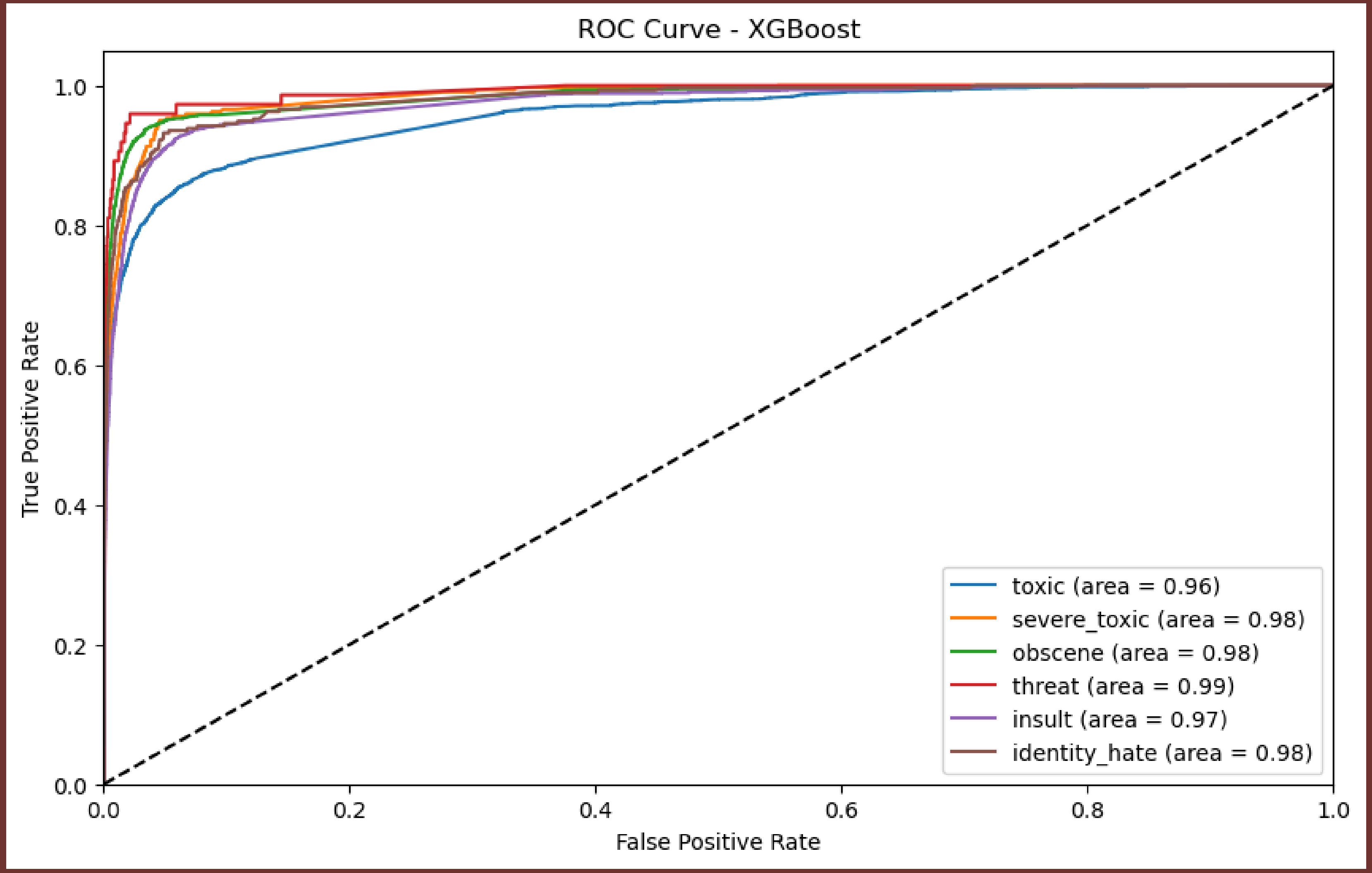
- **Modeling Log-Odds:** We train a separate logistic regression model for each toxic category.
- **Probability Estimation:** It transforms log-odds into probabilities using the logistic function.
- **Binary Classification:** For each toxic category, it fits a separate binary classifier using the TF-IDF features and log-odds ratios of word occurrences.



XGBoost

What is XGBoost Doing?

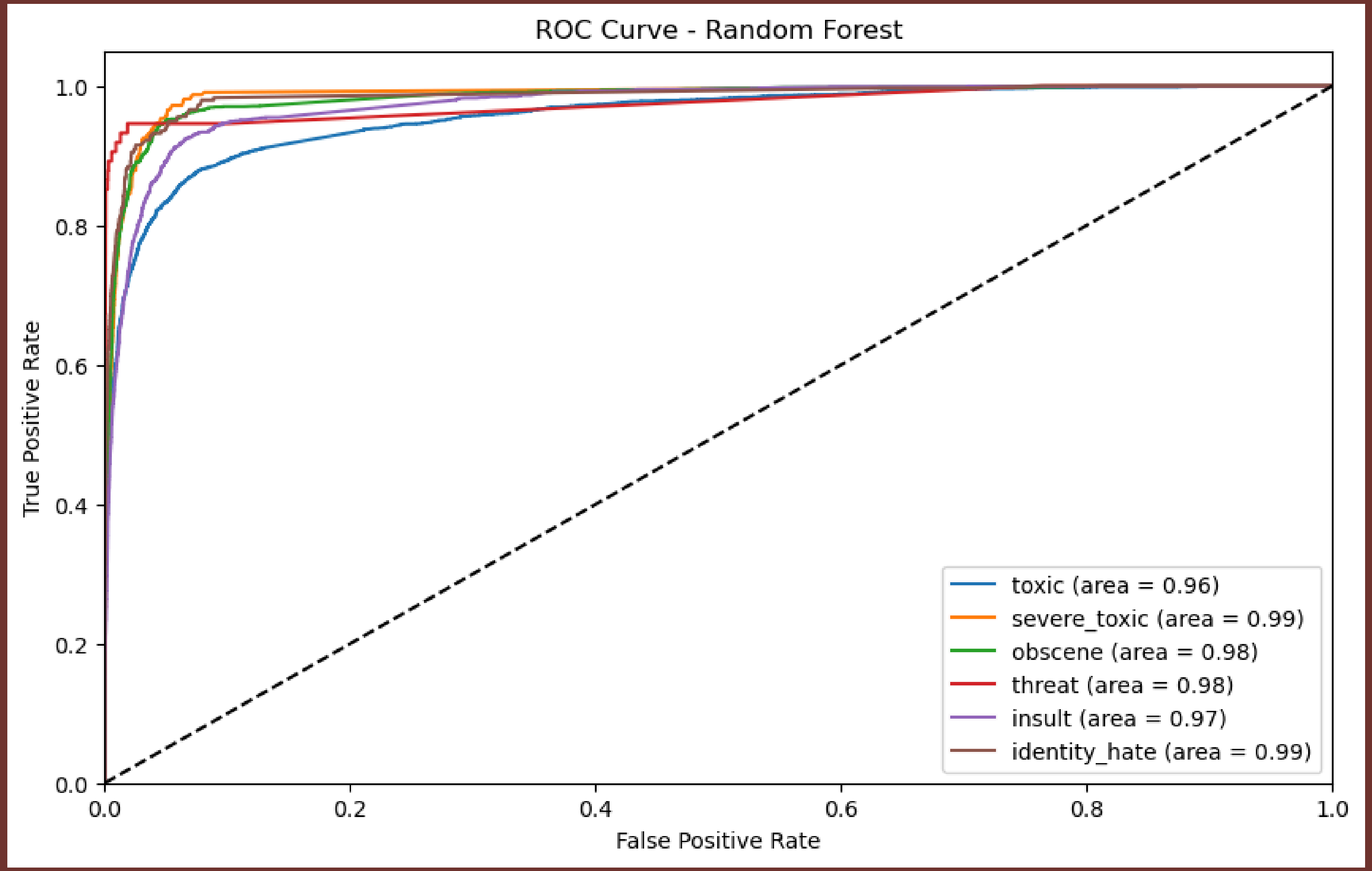
- **Ensemble Learning**: XGBoost builds an ensemble of decision trees, each correcting the errors of the previous ones.
- **Gradient Boosting**: It minimizes the loss function by sequentially adding trees that focus on the residuals.
- **Regularization**: XGBoost includes L1 and L2 regularization to prevent overfitting, enhancing the model's robustness.



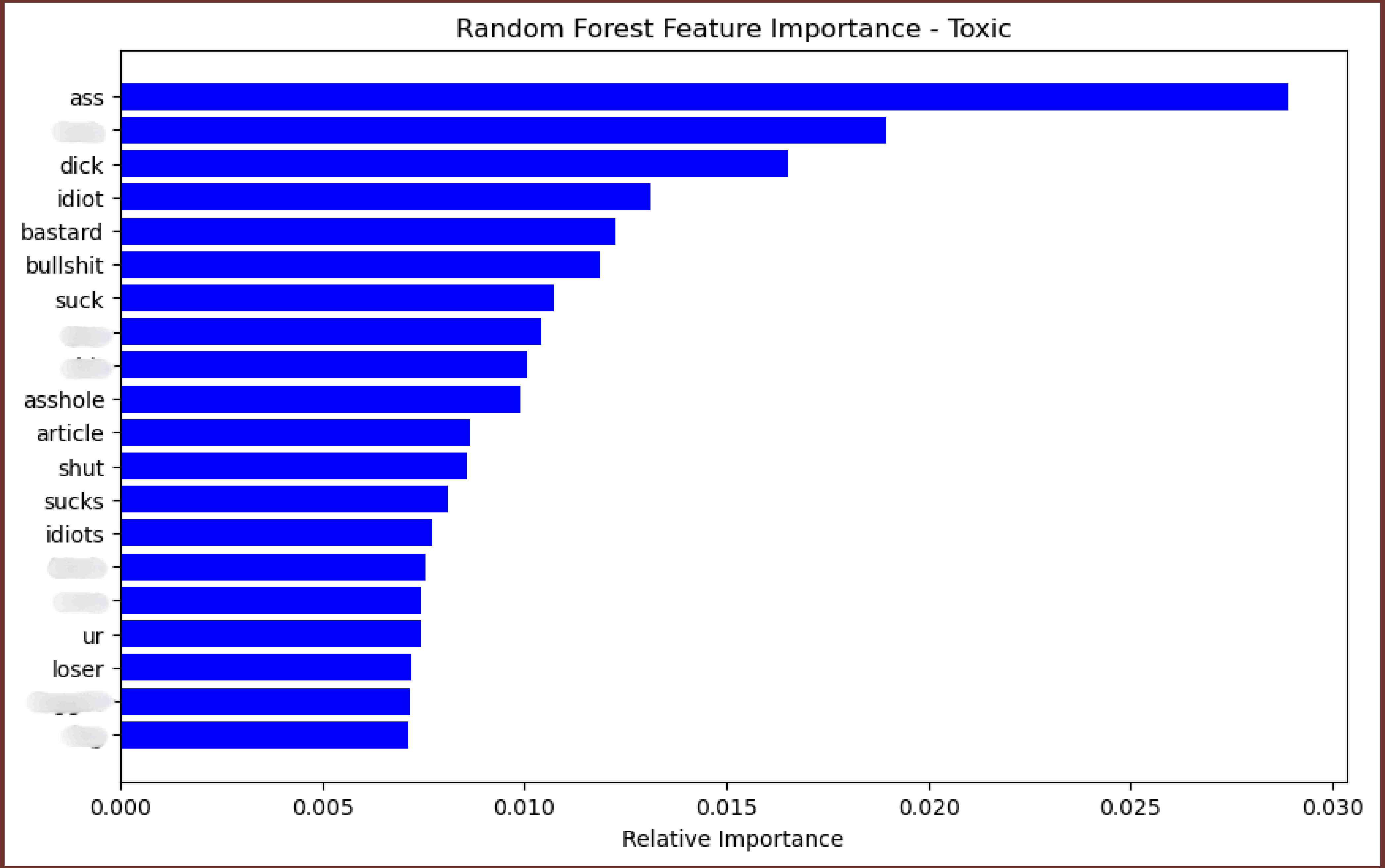
Random Forest

What is Random Forest Doing?

- **Bootstrap Aggregation (Bagging)**: Random Forest trains each tree on a random subset of the data to reduce variance.
- **Random Feature Selection**: It considers a random subset of features for splitting at each node, introducing diversity among trees.
- **Ensemble Averaging**: The final prediction is the average of the probabilities predicted by all trees, enhancing stability and accuracy.



Feature Importance



LSTM (Long Short-Term Memory)

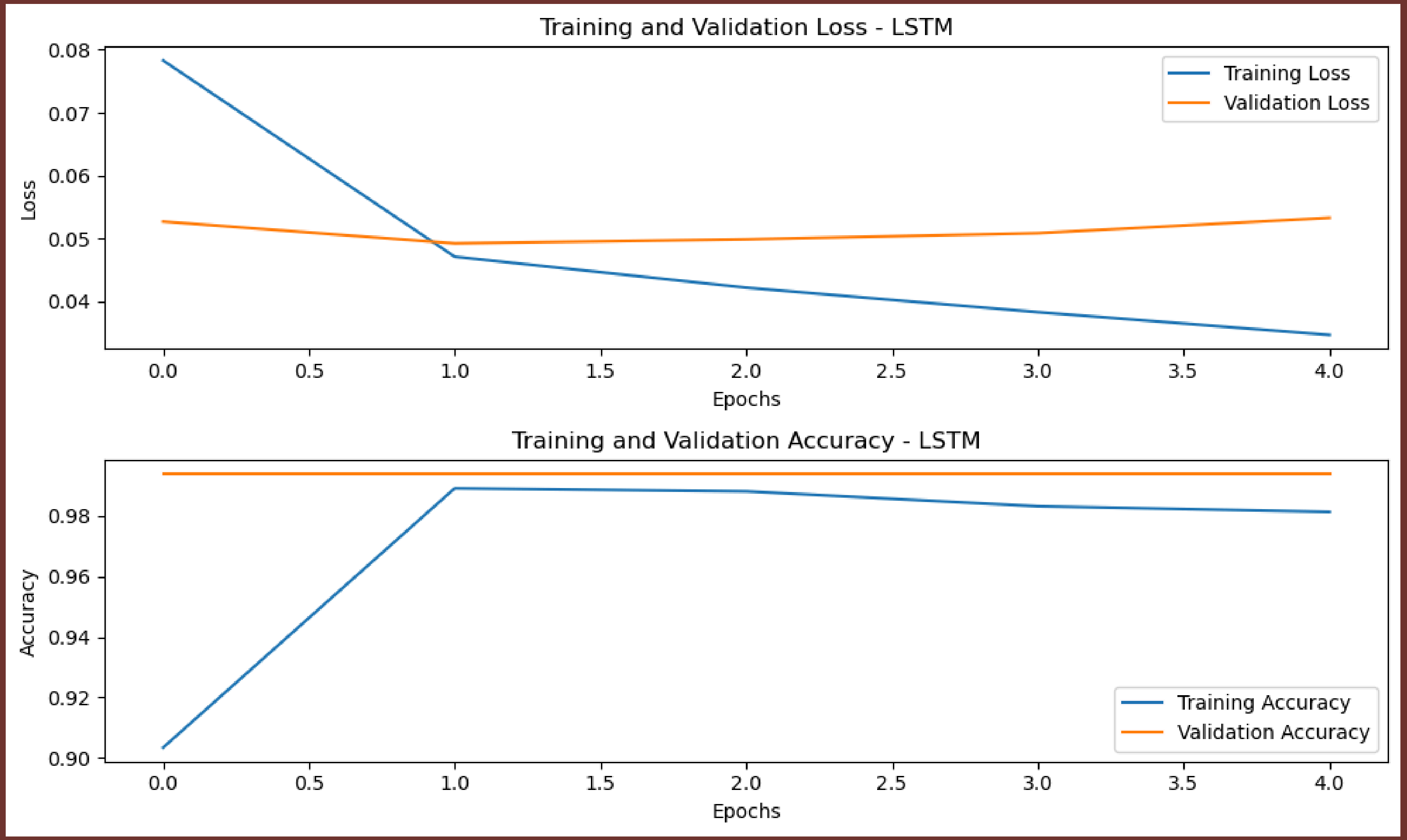
Why LSTM?

LSTM networks are a type of recurrent neural network (RNN) that excel at capturing long-term dependencies in sequential data. They are particularly well-suited for text data, where the context of words can span many tokens.

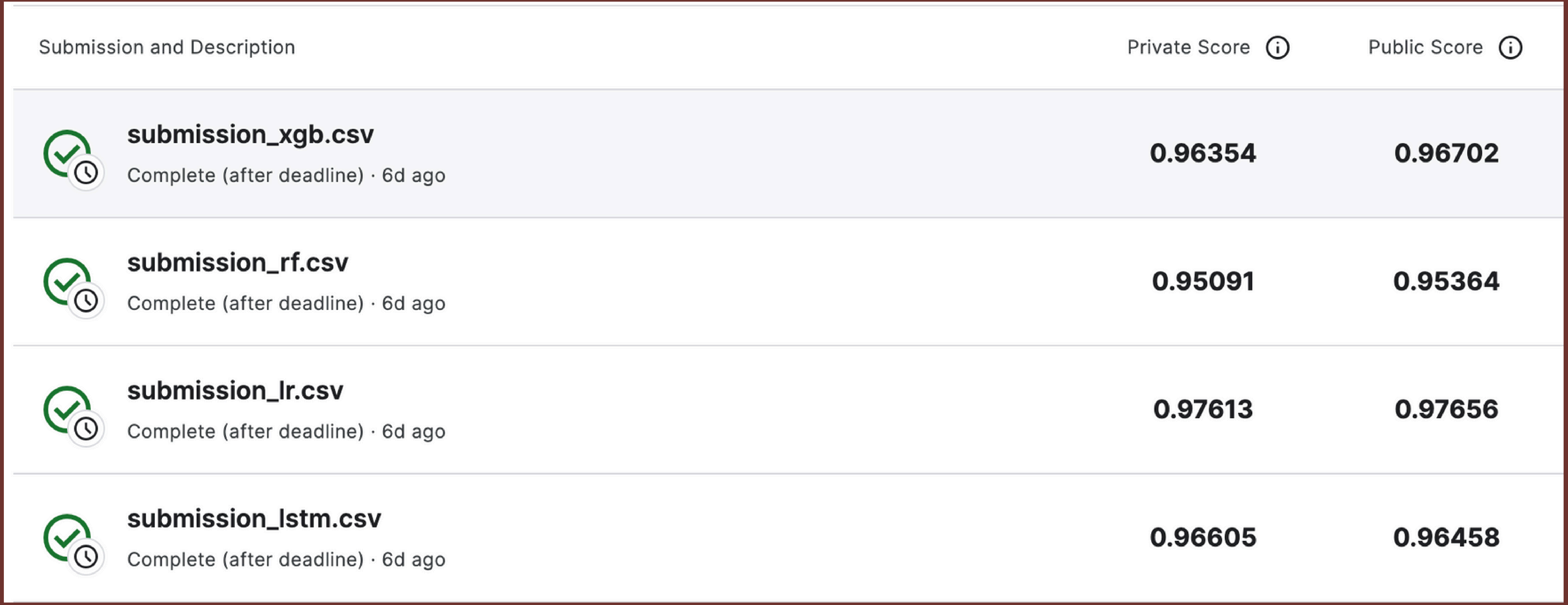
LSTM

What is LSTM Doing?

- Memory Cells: LSTM units have memory cells that store information over long periods, controlled by input, forget, and output gates.
- Sequential Learning: LSTM networks process input sequences one token at a time, maintaining a hidden state that captures context.
- Long-Term Dependencies: LSTMs can capture the context of words across long sequences, making them ideal for understanding the nuanced patterns in toxic comments.



The Results



Submission and Description	Private Score	Public Score
submission_xgb.csv Complete (after deadline) · 6d ago	**0.96354**	**0.96702**
submission_rf.csv Complete (after deadline) · 6d ago	**0.95091**	**0.95364**
submission_lr.csv Complete (after deadline) · 6d ago	**0.97613**	**0.97656**
submission_lstm.csv Complete (after deadline) · 6d ago	**0.96605**	**0.96458**

The Future & Plans

Applications

- Deploy models in real-time comment moderation systems.
- Extend to other languages and platforms.
- Incorporate more advanced NLP techniques like transformers.

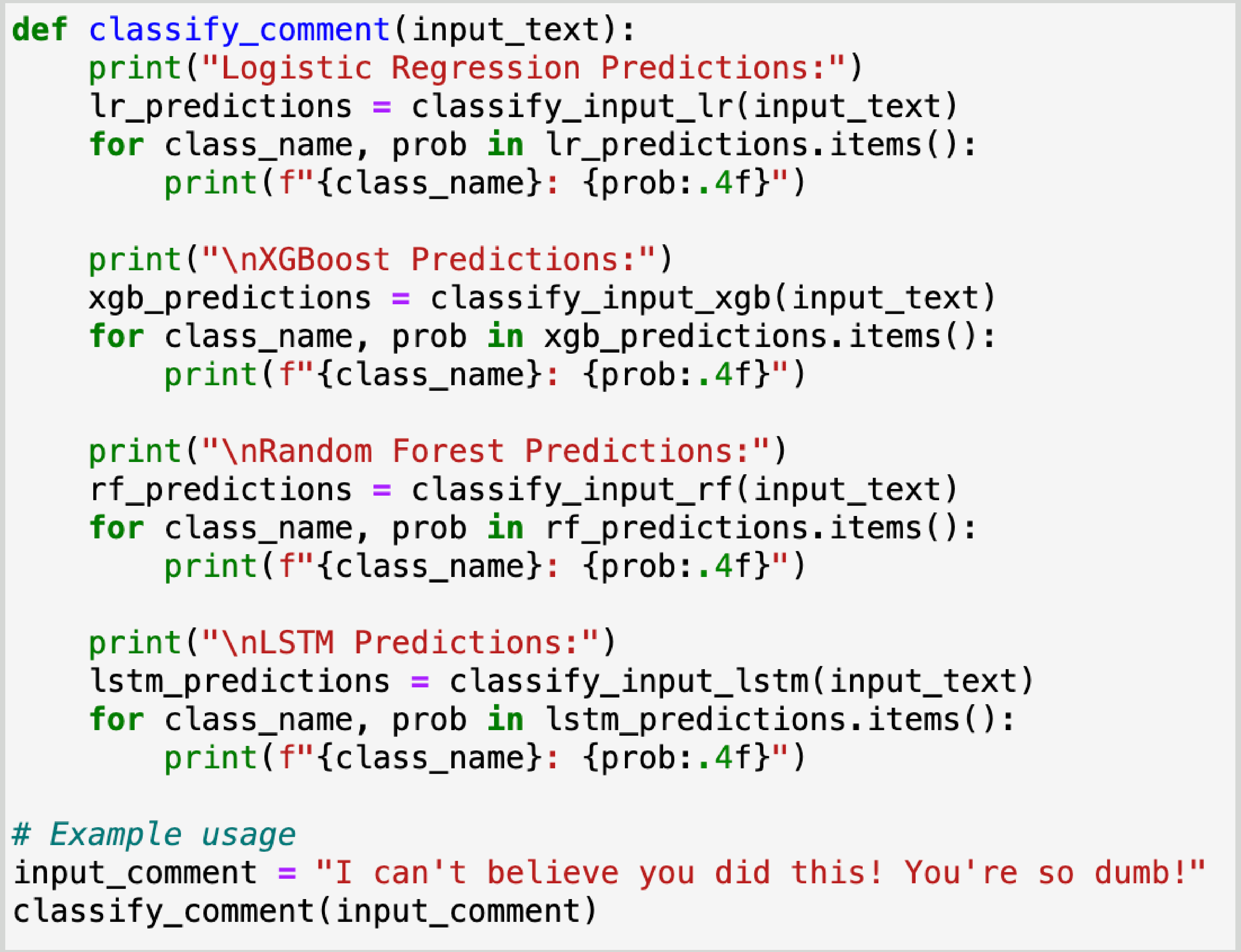
Potential

- Fine-tune models with larger datasets.
- Experiment with ensemble methods combining multiple models.
- Continuous model training with new data to adapt to evolving language.

Real-Time Classification

Objective: Allow users to input a comment and classify it into six toxic categories: toxic, severe toxic, obscene, threat, insult, and identity hate.

Applications:
Content Moderation: Real-time detection of toxic comments.
User Interaction: Immediate feedback on comment toxicity.



```python
def classify_comment(input_text):
    print("Logistic Regression Predictions:")
    lr_predictions = classify_input_lr(input_text)
    for class_name, prob in lr_predictions.items():
        print(f"{class_name}: {prob:.4f}")

    print("\nXGBoost Predictions:")
    xgb_predictions = classify_input_xgb(input_text)
    for class_name, prob in xgb_predictions.items():
        print(f"{class_name}: {prob:.4f}")

    print("\nRandom Forest Predictions:")
    rf_predictions = classify_input_rf(input_text)
    for class_name, prob in rf_predictions.items():
        print(f"{class_name}: {prob:.4f}")

    print("\nLSTM Predictions:")
    lstm_predictions = classify_input_lstm(input_text)
    for class_name, prob in lstm_predictions.items():
        print(f"{class_name}: {prob:.4f}")

# Example usage
input_comment = "I can't believe you did this! You're so dumb!"
classify_comment(input_comment)
```

Summary

Summary:

Successfully developed and evaluated multiple models for toxic comment classification.

Combined traditional machine learning with deep learning for robust performance.

Visualizations provided insights into model behavior and feature importance.

THANK YOU

Jaemin Han 2024
For WCD
✉ j4emin.han@gmail.com