Infosys Springboard Internship 5.0
H ATHULKRISHNAN
athulakhil28@gmail.com
AI Internship Batch 2

Hate Speech Detection

4th November 2024 - December 2024

Project Mentor : Dr. N Jagan Mohan

INTRODUCTION

The project **Hate Speech Detection** involves identifying and classifying language that promotes hatred or violence against individuals or groups based on attributes like race, religion, gender, or sexual orientation . This process typically employs natural language processing (NLP) and machine learning techniques to analyze text data from social media, forums, and other online platforms [1] .

DATASET PREPARATION

In this project, four key datasets were utilized:

1. **Hate Speech and Offensive Language Dataset** [2] : A 2.55 MB CSV file with categories for hate speech (0), offensive language (1), and neither (2).
2. **Multilingual Hate Speech Dataset** [3] : A 3.65 GB CSV containing hate speech data across languages, using 0 for non-hate and 1 for hate.
3. **Dynamically Generated Hate Speech Dataset** [4] : A synthetic dataset of labeled hate and non-hate speech text, created from dynamic data collection.
4. **ETHOS Hate Speech Dataset** [5] : Contains 998 comments, with 565 labeled as non-hate and 433 as hate.

Only the text and label fields were extracted and mapped as 0 (hate speech) and 1 (non-hate). The final dataset, ***hate_speech_combined_dataset.csv***, includes **34,621 entries**: 26,627 labeled as hate speech and 7,994 as non-hate.

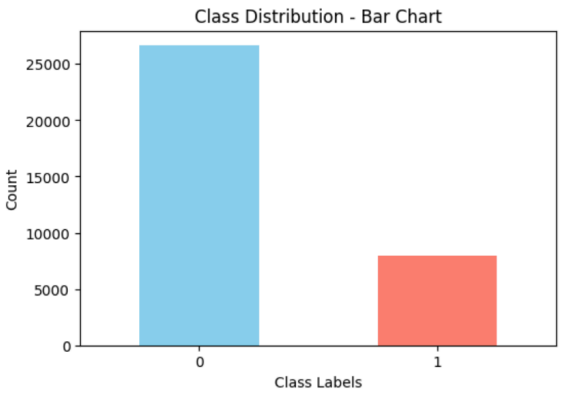
DATASET PREPROCESSING

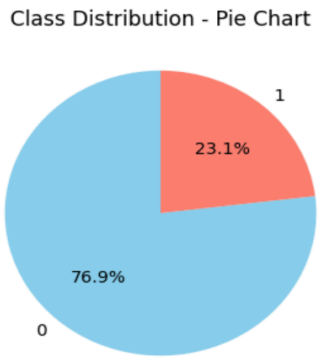
The dataset preprocessing aimed to optimize data for model training by removing irrelevant elements [6] . Key steps included:

1. **URL Removal**: URLs are removed because they often do not provide meaningful content for text analysis and could introduce noise, especially when analyzing hate speech.
2. **Mentions and Hashtags Removal**: Social media mentions (@usernames) and hashtags (#topic) are stripped away because they don't add valuable semantic information for the task at hand and can introduce unnecessary variability.
3. **Non-Alphabetic Characters Removal**: Removing numbers and punctuation helps focus on the core words, reducing distractions in the data.
4. **Text Normalization**: Converting all text to lowercase ensures consistency and prevents the model from treating the same word differently based on case (e.g., "Love" vs. "love").
5. **Duplicates and Whitespace Removal**: Duplicates are removed to prevent data from skewing the model's learning, and excessive spaces are eliminated to maintain uniform formatting.
6. **Stop Words**: Stop words (like "the," "is," "and") are removed because they don't contribute much meaning for text classification tasks.
7. **Stemming**: Reduces words to their root forms (e.g., "running" becomes "run").
8. **Lemmatization**: Ensures that words are reduced to their base forms, taking into account their meaning (e.g., "better" becomes "good").

Final dataset shape after cleaning: **(34621 , 2),** distributed as follows:

- **Hate Speech (0)**: 26,627 entries
- **Non-Hate Speech (1)**: 7,994 entries





TEXT VECTORIZATION USING TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used to convert text data into numerical values for machine learning models . It evaluates the importance of a word in a document relative to a collection of documents [7] .

- **Term Frequency (TF)** measures how often a word appears in a document.
- **Inverse Document Frequency (IDF)** measures how common or rare a word is across all documents.
- **TF-IDF Score** is the product of TF and IDF, which helps identify words that are significant in a particular document but rare across the entire dataset.

What TF-IDF Does:

- Transforms text into numerical features for machine learning.
- Highlights important words by assigning higher scores to those that are frequent in a document but rare in the overall corpus.
- Reduces the weight of common, uninformative words like "the," "is," and "and."

In the context of my internship project on **hate speech detection**, TF-IDF helps the model focus on relevant terms while reducing the influence of common words, improving the model's ability to distinguish between hate speech and non-hate speech .

DATA AUGMENTATION

Data Augmentation for Imbalanced Datasets

In machine learning, class imbalance is a common issue that can lead to model bias, where the model favors the majority class and underperforms on the minority class. In this project, the dataset was imbalanced, with a significantly larger number of **Hate Speech** instances compared to **Non-Hate Speech** entries. To address this imbalance and ensure a more equitable model performance, data augmentation techniques were employed .

Dataset Overview

After preprocessing, the final dataset had the following distribution:

- **Hate Speech (0)**: 26,627 entries
- **Non-Hate Speech (1)**: 7,994 entries

This disparity in the dataset's class distribution could lead to model bias, where the model may be overly influenced by the majority class. To prevent this, data augmentation techniques were applied to balance the dataset.

Data Augmentation Techniques

Data augmentation artificially increases the size of the underrepresented class, allowing the model to learn more varied patterns and improving overall performance. In the context of imbalanced datasets, it helps prevent the model from favoring the majority class, thus improving fairness and classification accuracy [8] .

Common data augmentation strategies for balancing datasets [8] include :

1. **Oversampling the Minority Class**: This approach increases the number of samples in the minority class by duplicating existing instances. While effective, it may sometimes lead to overfitting as the model might memorize duplicate instances.
    a. **Random Oversampling**: Duplicates random samples from the minority class.
    b. **SMOTE (Synthetic Minority Over-sampling Technique)**: Instead of simple duplication, SMOTE generates synthetic samples by interpolating between existing samples. This method enhances generalization by introducing variations and reducing the risk of overfitting.
2. **Undersampling the Majority Class**: This technique reduces the number of majority class samples to match the minority class. While this leads to a balanced dataset, it can result in the loss of valuable information from the majority class.

Selected Approach: SMOTE

For this project, **SMOTE** was chosen as the preferred data augmentation method. SMOTE helps generate synthetic samples for the minority class by interpolating between existing data points, thus creating a more balanced dataset. This technique introduces greater variability in the minority class, which not only helps prevent overfitting but also improves the model's ability to generalize [8] .

After applying SMOTE, the dataset shape was modified to **(53254 , 2)**, with the following distribution:

- **Hate Speech (0)**: 26,627 entries
- **Non-Hate Speech (1)**: 26,627 entries

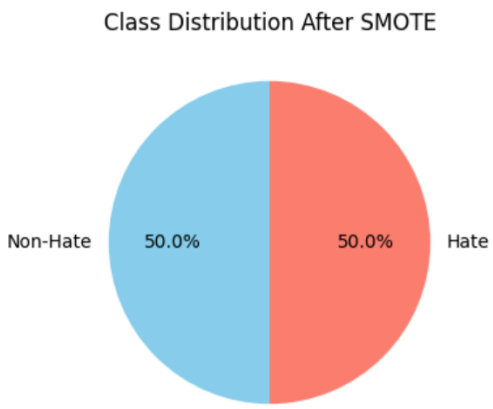


Fig 2 : Data visualization after SMOTE

ALGORITHMS EMPLOYED IN MODEL BUILDING

1. **Logistic Regression**

   Logistic Regression is a statistical method primarily used for binary classification tasks. It models the probability of a binary outcome based on one or more predictor variables, using the logistic function to produce a value between 0 and 1. It is widely used for its simplicity, interpretability, and effectiveness in binary classification problems, making it a good baseline model [9] .

2. **Random Forest Classifier**

   Random Forest is an ensemble learning method that builds multiple decision trees during training and combines their results to make predictions. Each tree in the forest is trained on a random subset of the data, and their outputs are aggregated (usually by majority voting for classification tasks). This method improves accuracy and reduces overfitting compared to a single decision tree, making it a robust and highly effective model for classification [9] .

3. **K-Nearest Neighbors (KNN)**

   K-Nearest Neighbors (KNN) is a simple algorithm that classifies a data point based on the majority class of its nearest neighbors in the feature space. The number of neighbors (k) is a user-defined parameter. KNN is easy to understand and implement, but it can be computationally expensive for large datasets as it requires calculating the distance to all other points during prediction [9] .

4. **XGBoost**

   XGBoost is an optimized implementation of gradient boosting, a technique that builds an

ensemble of decision trees in a sequential manner where each tree corrects the errors made by the previous one. XGBoost is known for its efficiency, scalability, and high performance in many machine learning tasks, particularly when handling large datasets and complex problems [9] .

5. **AdaBoost**

   AdaBoost (Adaptive Boosting) is an ensemble technique that combines multiple weak classifiers to create a strong classifier. It works by adjusting the weights of misclassified points and iteratively adding weak models to correct errors. AdaBoost can significantly improve model accuracy, especially when the base learners (weak classifiers) are simple models, such as decision trees [9] .

MODEL PERFORMANCE RESULTS

Model	Training Accuracy	Testing Accuracy	Precision	Recall	F1 Score
Logistic Regression	96.22%	94.11%	93.03%	95.36%	94.18%
Random Forest	92.57%	90.37%	89.79%	91.07%	90.43%
K-Nearest Neighbors	99.49%	67.59%	60.73%	99.44%	75.40%
XGBoost	89.36%	88.80%	95.34%	81.56%	87.92%
AdaBoost	88.28%	88.47%	95.13%	81.07%	87.54%

Fig 3 : Results 1

Model	Training Accuracy	Testing Accuracy
Logistic Regression	96.22%	94.11%
Random Forest	92.57%	90.37%
K-Nearest Neighbors	71.42%	66.32%
XGBoost	89.36%	88.80%
AdaBoost	88.28%	88.47%

Fig 4 : Results 2

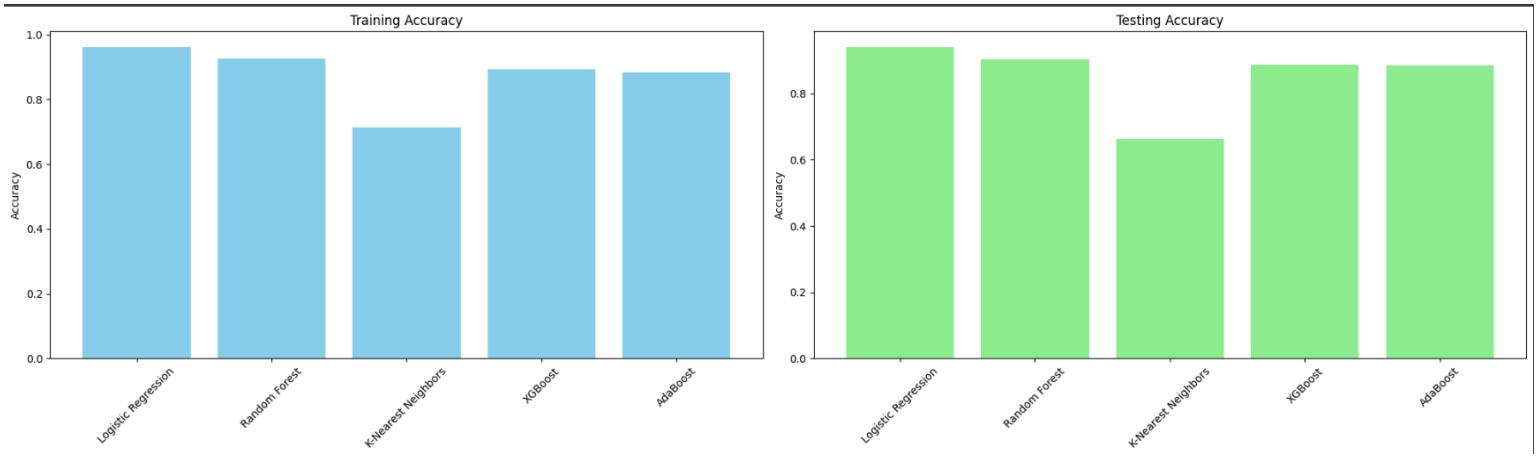


Fig 5 : Results Visualization

Best Model Based on Testing Accuracy

Model: Logistic Regression

- **Training Accuracy:** 96.22%
- **Testing Accuracy:** 94.11%
- **Precision:** 93.03%
- **Recall:** 95.36%
- **F1 Score:** 94.18%

Logistic Regression emerged as the best-performing model in this project, achieving the highest testing accuracy and a well-balanced F1 score. This indicates its robustness and effectiveness in identifying hate speech while maintaining low error rates for both false positives and false negatives.

Confusion Matrix And Analysis

A confusion matrix is a performance measurement tool used to evaluate classification models. It provides a detailed breakdown of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), offering insights into the model's performance [10] .

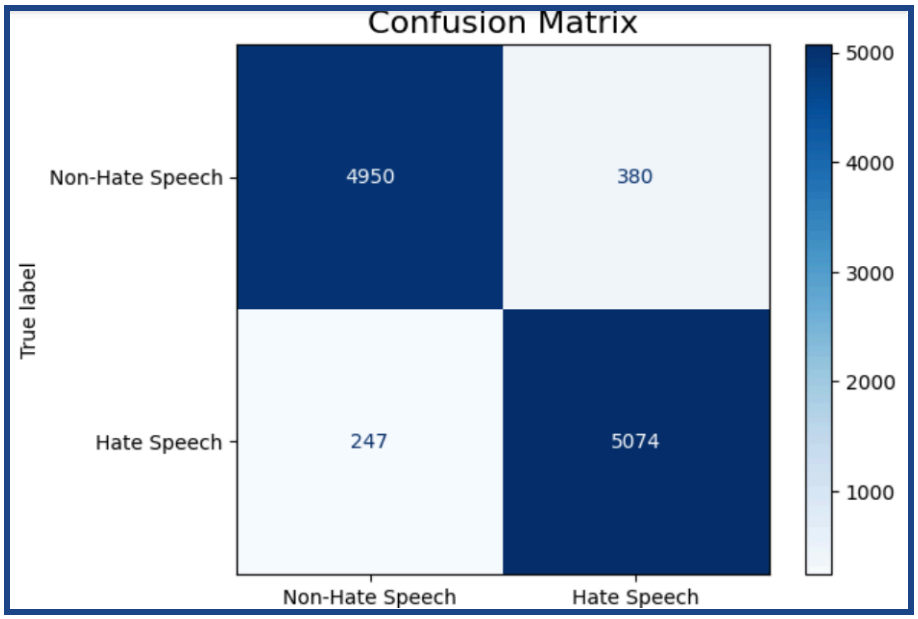


Fig 6 : Confusion Matrix

Confusion Matrix Breakdown

- **True Negatives (TN):** 4950 - Correctly identified "Non-Hate Speech."
- **False Positives (FP):** 380 - Incorrectly predicted "Hate Speech" for "Non-Hate Speech."
- **False Negatives (FN):** 247 - Missed some "Hate Speech" instances.
- **True Positives (TP):** 5074 - Correctly identified "Hate Speech."

Analysis

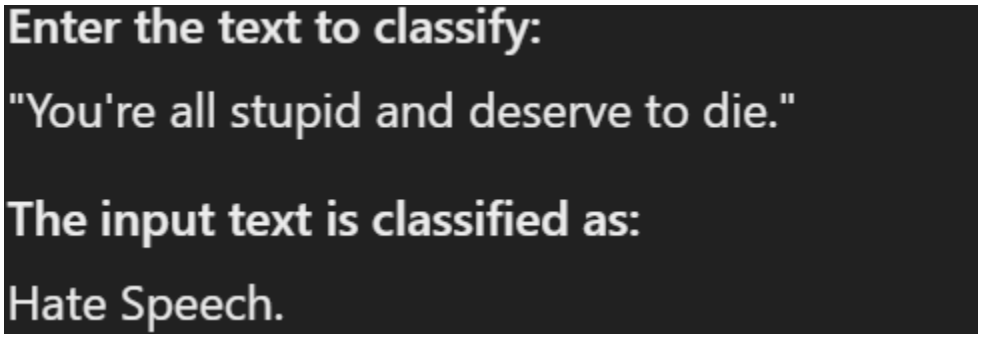
The confusion matrix highlights the following:

1. **High True Positives and True Negatives:** Indicates the model effectively distinguishes between "Hate Speech" and "Non-Hate Speech."
2. **Low False Positives:** Demonstrates the model minimizes misclassifying "Non-Hate Speech" as "Hate Speech."
3. **Low False Negatives:** Shows that the model accurately captures most "Hate Speech" instances.

These results affirm the model's reliability in hate speech detection, with a strong ability to balance false positives and false negatives.

SAMPLE OUTPUT

```
Enter the text to classify: hello world
The input text is classified as Non-Hate Speech.
```



SUMMARY OF ACHIEVEMENTS

1. **Best Model:** Logistic Regression with 94.11% testing accuracy and 94.18% F1 Score.
2. **Best Precision:** XGBoost with 95.34% precision.
3. **Best F1 Score:** Logistic Regression with 94.18%.
4. **Overfitting:** K-Nearest Neighbors exhibited 71.42% training accuracy but significantly lower testing accuracy of 66.32%, indicating overfitting.
5. **Balanced Performance:** AdaBoost achieved a precision of 95.13% and recall of 81.07%, demonstrating a well-balanced approach.

FUTURE SCOPE

1. **Hyperparameter Tuning:** Experimenting with various hyperparameter configurations to further enhance model performance and accuracy.
2. **Deep Learning Models:** Exploring advanced techniques such as LSTMs or Transformers to better handle large datasets and capture complex patterns in textual data.
3. **Real-time Deployment:** Implementing the model in live systems for continuous and real-time detection of hate speech on online platforms.

CONCLUSION

The project successfully developed a machine learning model capable of classifying hate speech from text data. By utilizing effective preprocessing techniques, TF-IDF for feature extraction, and SMOTE for balancing the dataset, the model achieved strong performance metrics. Logistic Regression emerged as the best-performing algorithm, delivering high accuracy and F1 score.

This initiative provided valuable insights into text classification and showcased the potential of machine learning in addressing real-world challenges like online content moderation.

REFERENCES

1. **Jahan, M. S., & Oussalah, M.** (2023). A systematic review of hate speech automatic detection using natural language processing. *University of Oulu, CMVS*. Available online 9 May 2023, Version of Record 19 May 2023.
2. **Samoshyn, Andrii.** "Hate Speech and Offensive Language Dataset." *Kaggle*, [Dataset Link] , accessed 6-11-2024.
3. **Moosa, Wajid Hassan.** "Multilingual Hate Speech Dataset." *Kaggle*, [Dataset Link] , accessed 6-11-2024.
4. **Tensor Girl.** "Dynamically Generated Hate Speech Dataset." *Kaggle*, [Dataset Link] , accessed 6-11-2024.
5. **Mollas, Ioannis, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumaka.** "*Ethos-Hate-Speech-Dataset*" ,Paper : "ETHOS: An Online Hate Speech Detection Dataset." *GitHub*,[Dataset Link], accessed 6-11-2024.
6. DataStud, "Preprocessing Text Data for Machine Learning," *DataStud* . https://datastud.dev/posts/nlp-preprocess. [Accessed: Nov. 14, 2024]
7. GeeksforGeeks. "Understanding TF-IDF (Term Frequency – Inverse Document Frequency)."*GeeksforGeeks*,https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/. Accessed 26 Dec. 2024.
8. Dhiraj Bembade. "*How to Balance the Data in NLP"*. Kaggle, [Article Link] . Accessed [12-11-2024].
9. GeeksforGeeks. "Machine Learning Algorithms." *GeeksforGeeks*,, https://www.geeksforgeeks.org/machine-learning-algorithms/. Accessed 26 Dec. 2024.
10. Scikit-learn Developers. "Confusion Matrix." *Scikit-learn*,, https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html. Accessed 26 Dec. 2024.