

Hate Speech Detection Using Machine Learning

Project Report

Submitted By: Vipul Pareek [2201233]

Date: 18/11/2024

Introduction

Hate speech detection has become an essential area of research due to the increasing use of social media platforms and the proliferation of harmful content. This project aims to create a machine learning-based system that accurately identifies hate speech in text data. By leveraging natural language processing (NLP) and advanced machine learning techniques, the project attempts to mitigate the spread of hate speech online.

Objectives

- Develop a machine learning model capable of detecting hate speech with high accuracy.
- Explore the effectiveness of **Logistic Regression** and **Random Forest classifiers** for this task.
- Utilize advanced text preprocessing techniques to clean and standardize the data.
- Evaluate the model's performance on unseen data to ensure robustness.

Methodology

3.1 Dataset

- Source: The dataset used for this project was sourced from Kaggle/Twitter Sentiment Analysis [<https://www.kaggle.com/datasets/arkhoshghalb/twitter-sentiment-analysis-hatred-speech?select=train.csv>].
- Size: [39,900].
- Structure: The dataset contains three columns: Unique Id, tweet (text data) and label (binary label).

3.2 Data Preprocessing

Key preprocessing steps included converting text to lowercase, removing URLs, tokenization, and lemmatization.

3.3 Model Selection

Two machine learning models were implemented:

1. **Logistic Regression with hyperparameter tuning using GridSearchCV.**
2. **Random Forest with hyperparameter tuning using GridSearchCV.**

3.4 Data Split

Training Set: 60%

Validation Set: 20%

Test Set: 20%

3.5 Performance Metrics

Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.

Implementation

4.1 Tools and Libraries

Python, Pandas, NumPy, NLTK, Scikit-learn, Seaborn, and Matplotlib.

4.2 Preprocessing Pipeline

Example of tweet before and after processing.

4.3 Hyperparameter Tuning Results

Logistic Regression: C=1.0, Solver=liblinear

Random Forest: n_estimators=100, max_depth=None.

4.4 Model Training and Evaluation

The models were trained on the training set, tuned using the validation set, and tested on the test set.

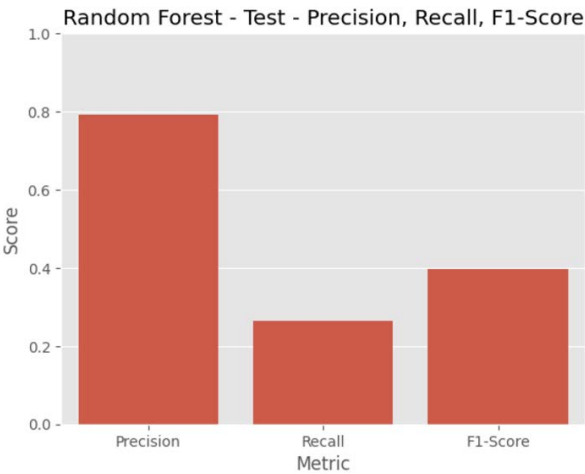
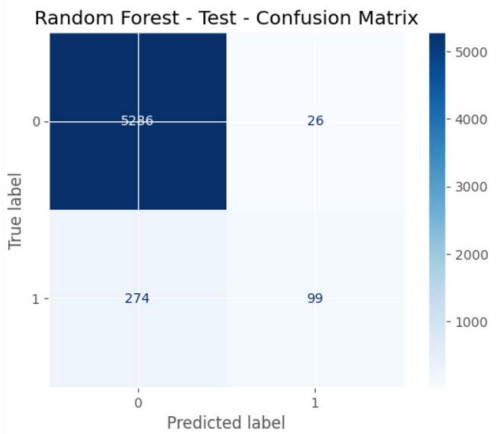
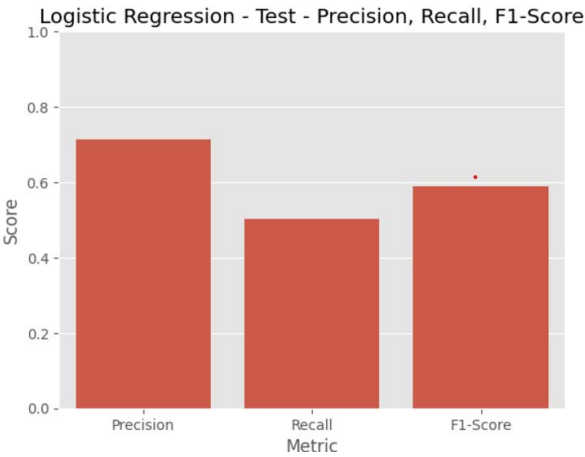
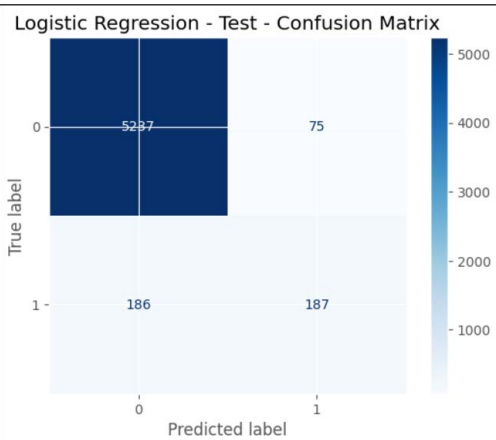
Results and Discussion

5.1 Model Performance

- **Logistic Regression**
 - **Hyper parameters Used:**
 - C (Inverse Regularization Strength): 100
 - Solver: lbfgs
 - **Test Accuracy:** 95.41%
- **Random Forest Classifier**
 - **Hyper Parameters Used:**
 - Number of Estimators (Trees): 50
 - Maximum Depth of Trees: None (unlimited)
 - **Test Accuracy:** 94.72%

5.2 Visualization

Confusion Matrix and Precision, Recall, and F1-Score Bar Charts.



Conclusion and Future Work

6.1 Conclusion

The project developed a machine learning pipeline for hate speech detection with commendable results.


6.2 Future Work

Exploring deep learning models like LSTMs or Transformers, expanding the dataset for better generalization.

7.1 Literature Survey

Reference Paper

Title: Automated Hate Speech Detection and the Problem of Offensive Language
Authors: Thomas Davidson, Dana Warmasley, Michael Macy, Ingmar Weber
Published In: Proceedings of the 11th International Conference on Web and Social Media (ICWSM), 2017

Key Comparison with the Reference Paper		
Aspect	Reference Paper	Our Project
Dataset	Twitter dataset with three classes: hate speech, offensive language, and neither	Twitter dataset with binary labels: hate (1) and non-hate (0)
Preprocessing	Tokenization, stopword removal, and basic noise reduction	Extended preprocessing: lemmatization, trigram vectorization, advanced noise filtering
Model	Logistic Regression and other models (SVM, Random Forest) as baselines	Logistic Regression with hyperparameter tuning (GridSearchCV)
Feature Representation	TF-IDF with default parameters	TF-IDF with trigram-based feature extraction
Metrics	Accuracy, precision, recall, and F1-score	Accuracy, precision, recall, F1-score, and confusion matrix visualization
Visualization	Minimal data visualization	Extensive visualization: class distribution, word clouds, and model performance plots
Hyperparameter Tuning	None	Extensive tuning using GridSearchCV to optimize parameters like <code>C</code> and <code>solver</code>
Model Complexity	Includes deep learning models like LSTMs, which require more computational resources 	Focused on Logistic Regression for a lightweight and interpretable solution