

HATE SHIELD

Submitted by

2448047: SANDHIKA G

2448056: SUJILIN JESICA S V

2448059: VARUN DSOUZA

For the course

CourseCode: MDS372

CourseName: Machine Learning

March – 2025

ABSTRACT

Hate speech has become a significant issue in the digital age, posing risks to individuals and communities. To address this challenge, machine learning algorithms can be leveraged to automatically detect and flag hate speech in both text and speech-based data. This project explores hate speech detection using a combination of **supervised learning models and deep learning techniques**, trained on a **labeled dataset** and a **generated dataset**. The model classifies text into three categories: **Hate Speech, Offensive Language, and Neutral Speech**. Additionally, **audio-to-text processing** is integrated, enabling the system to transcribe and analyze spoken content for hate speech detection.

The detection process involves **data preprocessing, feature extraction, and deep learning-based classification** using models such as **LSTM and Bidirectional LSTM**. Various linguistic features, including word usage, grammar, and contextual meaning, are analyzed to improve classification accuracy. While machine learning provides a scalable approach to hate speech detection, challenges such as **bias in training data, ambiguity in defining hate speech, and model interpretability** remain critical areas of concern. This report discusses these challenges and presents strategies to mitigate them.

Overall, this project demonstrates the potential of machine learning in **automated hate speech detection across text and speech formats**. The findings contribute to ongoing research efforts aimed at improving the **accuracy, fairness, and reliability** of hate speech detection systems.

TABLE OF CONTENTS

Sl.no	Content	Page.no
I	Introduction	1
II	Global and contemporary Competencies	2
2.1	LRNG (Learning and Research in the Next Generation)	2
2.2	Cross-Cutting Issues	2
2.3	Sustainable Development Goals (SDGs)	3
2.4	21st Century Skills	3
III	Methodology	4
3.1	Data Collection	4
3.2	Preprocessing	4
3.3	EDA	6
3.4	Feature Engineering	8
3.5	Model Description	10
3.6	Libraries	13
IV	Results and Discussion	14
4.1	About Dataset	14
4.2	Performance of Model and Comparison	14
4.3	Screenshots of Output	15
V	Conclusion	19
VI	References	20

I.INTRODUCTION

Hate speech detection using machine learning is a critical area of research in today's digital world, where the prevalence of online hate speech and harassment continues to rise. Hate speech includes any language or behavior that expresses prejudice, discrimination, or hostility toward individuals or groups based on characteristics such as race, ethnicity, gender, religion, or sexual orientation. The widespread dissemination of hate speech on social media and other online platforms poses significant challenges, making it essential to develop automated detection methods to mitigate its impact.

Machine learning provides a powerful approach to hate speech detection by analyzing large volumes of data and identifying linguistic patterns that distinguish hate speech from neutral or offensive content. In this project, a labeled dataset and a generated dataset were utilized to train a model capable of classifying text into three categories: **Hate Speech, Offensive Language, and Neutral Speech**. Additionally, the project integrates an **audio-to-text processing component**, allowing spoken language to be transcribed and analyzed for hate speech detection. This multimodal approach ensures broader applicability, extending beyond text-based content to audio-based speech analysis.

This report explores the methodologies used, including **data preprocessing, feature engineering, supervised learning techniques, deep learning models, and natural language processing (NLP) methods**. Furthermore, it discusses the challenges of hate speech detection, such as defining hate speech, dataset limitations, model biases, and ethical considerations in automated content moderation. The goal of this project is to contribute to the development of more robust and accurate hate speech detection systems that can be applied across different forms of communication, including text and speech.

II. GLOBAL AND CONTEMPORARY COMPETENCIES

Hate speech detection is a crucial area of research in today's digital age, addressing the rising concerns of **online safety, responsible AI, and ethical technology use**. The project enhances competencies in **data science, artificial intelligence, and natural language processing (NLP)** while promoting social responsibility. By developing a **machine learning model** that detects hate speech from text and audio inputs, this project contributes to digital literacy, ethical AI development, and global efforts to mitigate online harassment and cyberbullying.

2.1 LRNG (Learning and Research in the Next Generation)

The project integrates **cutting-edge AI techniques, including deep learning (LSTM) and NLP**, fostering innovation in hate speech detection. It promotes **interdisciplinary learning**, combining machine learning, linguistics, and ethical considerations. The inclusion of **audio-to-text processing** expands its applicability, making it relevant in real-world scenarios such as **social media monitoring, automated content moderation, and digital forensics**.

2.2 Cross-Cutting Issues

This project addresses several cross-cutting issues, including:

- **Ethical AI and Bias Mitigation:** By training the model on diverse datasets and applying **class balancing techniques**, the project ensures fairness in classification, reducing algorithmic biases.
- **Cybersecurity and Online Safety:** Hate speech detection helps create **safer online environments**, reducing the spread of harmful content.
- **Freedom of Speech vs. Harmful Speech:** The project emphasizes the **responsible use of AI**, ensuring that detection models do not unfairly censor legitimate discussions while tackling harmful language.
- **Digital Citizenship and Media Literacy:** The research fosters **critical awareness** of online communication, encouraging responsible digital interactions.

2.3 Sustainable Development Goals (SDGs)

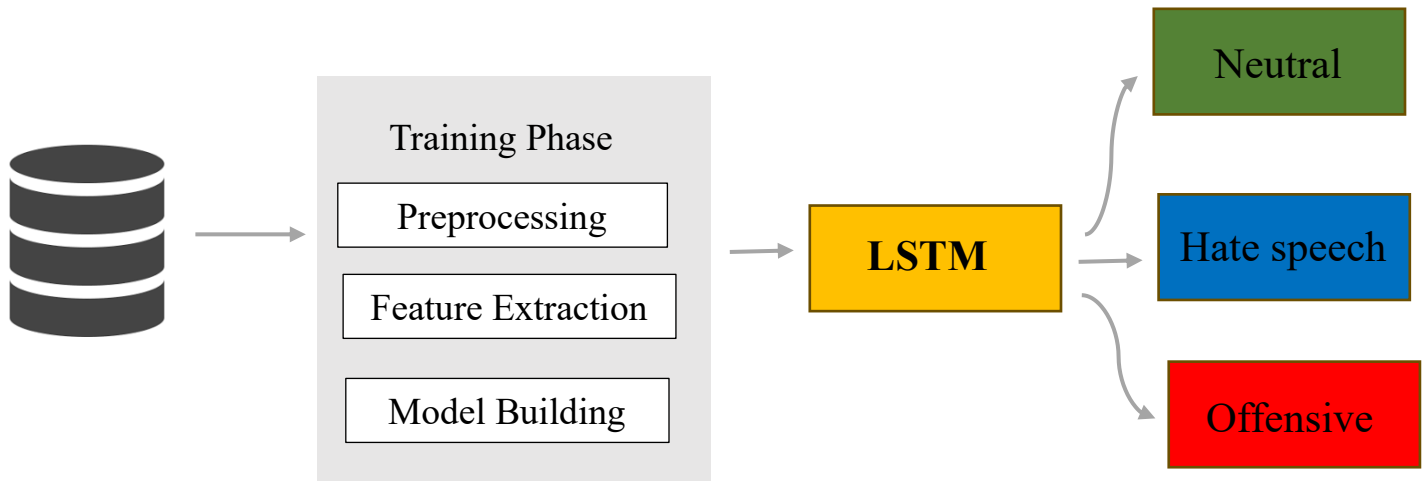
This project directly aligns with **SDG 16: Peace, Justice, and Strong Institutions**, which aims to promote peaceful and inclusive societies, provide access to justice, and build accountable institutions.

- **Reducing Online Hate Speech** – The project enhances digital safety by detecting and mitigating hate speech, contributing to a safer and more inclusive online environment.
- **Promoting Ethical AI Use** – By ensuring fairness and transparency in machine learning models, the project supports responsible AI development and trust in digital communication.

2.4 21st Century Skills

- **Critical Thinking & Problem-Solving** – Developing an **AI model** that accurately distinguishes between **hate speech, offensive speech, and neutral speech**.
- **Collaboration & Interdisciplinary Learning** – Integrating **machine learning, linguistics, and social sciences**.
- **Digital Literacy & Technological Proficiency** – Working with **deep learning, NLP, and AI ethics**.
- **Communication & Ethical AI Awareness** – Encouraging discussions on **freedom of speech, digital ethics, and bias in AI**.
- **Innovation & Research** – Applying **cutting-edge AI techniques** to solve a pressing global issue.

III. METHODOLOGY



3.1. DATA COLLECTION

A labelled dataset was utilized for training the hate speech detection model, along with an additional generated dataset to enhance the diversity and robustness of the training data. The labelled dataset consisted of pre-classified examples categorized into three classes: Hate Speech, Offensive Language, and Neutral. The generated dataset was created to supplement the existing data, ensuring a more balanced distribution across categories and improving model generalization.

3.2. PREPROCESSING

The preprocessing pipeline ensures that raw text data is cleaned, tokenized, and converted into a numerical format suitable for training the hate speech detection model. The following steps were applied to both the labeled dataset and the generated dataset:

1. Data Cleaning:

- ✓ All text was converted to lowercase to maintain consistency.
- ✓ URLs and special characters were removed to eliminate unnecessary noise.
- ✓ Stopwords were removed using the NLTK stopwords list to reduce irrelevant words.

- ✓ Lemmatization was applied using the WordNet lemmatizer to normalize words and reduce their variations.

2. Tokenization and Sequencing:

- ✓ A Tokenizer was trained on the cleaned text to convert words into numerical sequences.
- ✓ The vocabulary size was limited to 20,000 words, and out-of-vocabulary tokens were handled using a placeholder (<OOV>).
- ✓ The text sequences were padded to a maximum length of 100 tokens to ensure uniform input size.

3. Label Encoding:

- ✓ The dataset had three categories: Hate Speech, Offensive Language, and Neutral.
- ✓ Labels were converted into numerical form for model compatibility.

4. Handling Imbalanced Data:

- ✓ Class weights were computed using the `compute_class_weight` function to ensure balanced learning across all categories.

Processed Data (First 10 rows):

	clean_text	class
0	rt mayasolovely woman shouldnt complain cleani...	2
1	rt mleew boy dat coldtyga dwn bad cuffin dat h...	1
2	rt urkindofbrand dawg rt sbabylife ever fuck b...	1
3	rt cganderson vivabased look like tranny	1
4	rt shenikaroberts shit hear might true might f...	1
5	tmadisonx shit blow meclaim faithful somebody ...	1
6	brighterdays sit hate another bitch got much s...	1
7	selfiequeenbri cause im tired big bitch coming...	1
8	amp might get ya bitch back amp thats	1
9	rhythmixx hobby include fighting mariam bitch	1

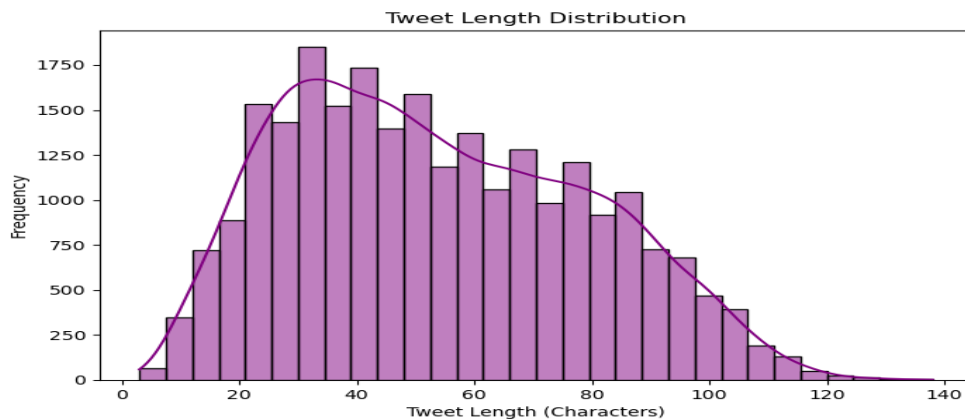
3.3.EDA

1.Handling Class Imbalance Methods:

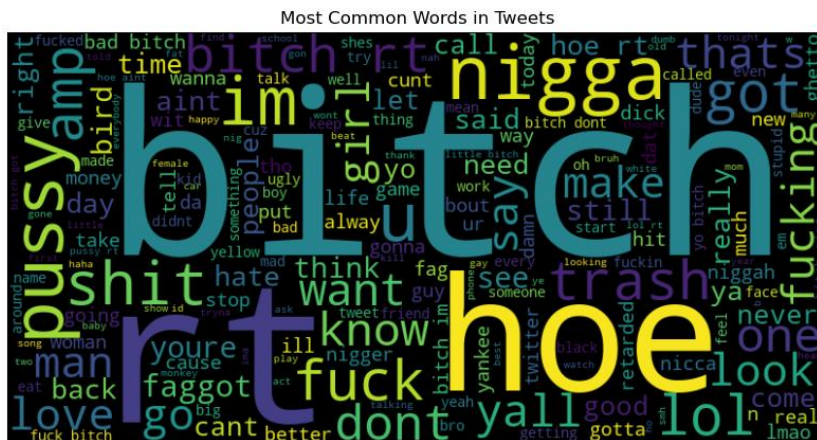


The class distribution shows a significant imbalance, with the "Offensive" class (1) having the highest count, while "Hate Speech" (0) is underrepresented. This imbalance can affect model performance, causing it to favour the majority class. To address this, methods like **oversampling** (increasing minority class data), **undersampling** (reducing majority class data), and **class weighting** (penalizing misclassification of minority classes) can be used. Hybrid approaches and advanced techniques like **SMOTE** are also effective for improving model accuracy across all classes.

2. Analysis of Tweet Length Distribution

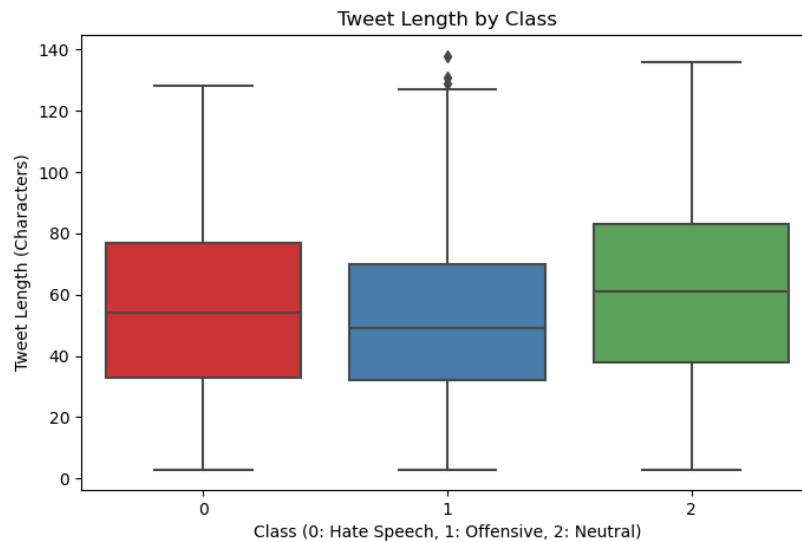


3. Word Frequency Analysis



- **Presence of Offensive Language:** A significant portion of the words are offensive, derogatory, or explicit, indicating the nature of language used in these tweets.
- **High Usage of Slang and Informal Terms:** Many words in the word cloud, such as "lol," "yall," and "gon," suggest the prevalence of informal and conversational language in the dataset.
- **Social Media Sentiment Indicator:** The word distribution indicates a mix of negative and neutral sentiments, reflecting common online interactions, disputes, or expressions.
- **Potential Need for Content Moderation:** The dominance of offensive terms highlights the necessity for social media platforms to implement better filtering and moderation mechanisms.

4. Tweet Length Distribution



- **Median Tweet Length Consistency:** The median tweet length remains fairly consistent across all three classes, suggesting that tweet length alone may not be a strong differentiator.
- **Outliers in Offensive Tweets:** The offensive category (1) exhibits a few outliers with significantly higher character counts, indicating occasional longer tweets.
- **Overall Spread Similarity:** All three categories show a similar interquartile range (IQR), meaning the variation in tweet lengths is comparable across classes.
- **Potential for Further Analysis:** While tweet length alone may not distinguish between hate speech, offensive, and neutral tweets, combining it with other features could enhance classification accuracy.

3.4. FEATURE ENGINEERING

1. Text Preprocessing (Feature Transformation)

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'https?://\S+|www\.\S+', '', text) # Remove URLs
    text = re.sub(r'^a-zA-Z\s', '', text) # Remove special characters
    words = [lemmatizer.lemmatize(word) for word in text.split() if word not in stop_words]
    return ' '.join(words)
```

- **Key Engineering:**
 - ✓ **Lowercasing:** Reduces vocabulary complexity
 - ✓ **URL Removal:** Eliminates noisy web links
 - ✓ **Special Character Removal:** Focuses on linguistic content
 - ✓ **Lemmatization:** Reduces words to root forms ("running" → "run")
 - ✓ **Stopword Removal:** Drops uninformative words ("the", "and")

2. Sequence Encoding (Feature Creation)

```
# Tokenization and Padding
max_words = 20000 # Vocabulary size
max_len = 100 # Max sequence length
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(data['clean_text'])
sequences = tokenizer.texts_to_sequences(data['clean_text'])
x = pad_sequences(sequences, maxlen=max_len)
y = data['class']
```

- **Key Engineering:**
 - ✓ **Tokenization:** Maps words to integer IDs
 - ✓ **Padding:** Standardizes sequence length to max_len=100
 - ✓ **Out-of-Vocabulary Handling:** Uses <OOV> token for unknown words

3. Class Weighting (Feature Balancing)

```
# Compute Class Weights (Handles Imbalanced Data)
class_weights = compute_class_weight('balanced', classes=np.unique(y_train), y=y_train)
```

- **Key Engineering:**
 - ✓ Addresses imbalanced classes by weighting underrepresented classes more heavily during training

4.Embedded Features (LSTM Specific)

```
Embedding(input_dim=max_words, output_dim=128)
```

- **Key Engineering:**
 - ✓ Learns dense vector representations of words during training
 - ✓ Converts sparse tokenized sequences into rich 128-dimensional vectors

3.5. MODEL DESCRIPTION

Model Building

A suite of classification models was constructed, each employing distinct algorithms and methodologies. The models were implemented using Python with libraries such as scikit-learn and TensorFlow/Keras.

Traditional Machine Learning Models

- **Logistic Regression:** A linear model utilizing a logistic function to model the probability of multi-class membership. The implementation used a maximum iteration limit of 1000 for convergence.
- **Random Forest:** An ensemble learning technique composed of 100 decision trees. The model aggregates predictions from individual trees to improve predictive accuracy and robustness.
- **Decision Tree:** A hierarchical model employing a tree structure to partition the feature space and classify instances. The default parameters, including the Gini impurity criterion for splitting, were used.
- **Naive Bayes:** A probabilistic classifier grounded in Bayes' theorem, assuming independence between features. The Multinomial Naive Bayes variant was used, tailored for discrete feature counts such as word frequencies in text data.

Deep Learning Model: LSTM

A deep learning model based on Long Short-Term Memory (LSTM) networks was developed to capture sequential dependencies in the text data. The LSTM architecture comprised the following layers:

- **Embedding Layer:** Maps words to 128-dimensional vector representations, with a vocabulary size of 20,000.
- **Spatial Dropout Layer:** Applies dropout to entire feature maps, mitigating overfitting and improving generalization.
- **Bidirectional LSTM Layer:** A bidirectional LSTM layer with 128 units, employing dropout and recurrent dropout rates of 0.3 to capture sequential information in both forward and backward directions.
- **Dense Layers:** A fully connected layer with 64 units and ReLU activation, followed by a dropout layer with a rate of 0.3.
- **Output Layer:** A dense output layer with 3 units and softmax activation for multi-class classification, corresponding to the Hate Speech, Offensive, and Neutral categories.

The LSTM model was trained using the Adam optimizer and sparse categorical cross entropy loss function over 10 epochs with a batch size of 32. Class weights were computed to address class imbalance.

Model Selection

The selection of the optimal model was guided by a comparative assessment of their performance across various evaluation metrics. The aim was to balance accuracy, computational efficiency, and class-wise predictive power.

Model Evaluation

Classification Report (Decision Tree):				
	precision	recall	f1-score	support
0	0.13	0.14	0.13	442
1	0.86	0.85	0.85	5729
2	0.47	0.47	0.47	1280
accuracy			0.74	7451
macro avg	0.49	0.49	0.49	7451
weighted avg	0.75	0.74	0.74	7451
Classification Report (Naive Bayes):				
	precision	recall	f1-score	support
0	0.07	0.25	0.11	442
1	0.78	0.62	0.69	5729
2	0.20	0.20	0.20	1280
accuracy			0.53	7451
macro avg	0.35	0.36	0.33	7451
weighted avg	0.64	0.53	0.57	7451

Classification Report (Logistic Regression):

	precision	recall	f1-score	support
0	0.50	0.00	0.00	442
1	0.77	1.00	0.87	5729
2	0.62	0.01	0.02	1280
accuracy			0.77	7451
macro avg	0.63	0.34	0.30	7451
weighted avg	0.73	0.77	0.67	7451

Classification Report (Random Forest):

	precision	recall	f1-score	support
0	0.39	0.03	0.06	442
1	0.81	0.97	0.89	5729
2	0.68	0.29	0.40	1280
accuracy			0.80	7451
macro avg	0.63	0.43	0.45	7451
weighted avg	0.76	0.80	0.75	7451

Model	Accuracy	Weighted F1-score	Key Observations
Logistic Regression	0.77	0.67	Excellent Class 1 (Offensive) recall, very poor Class 0 (Hate) and 2 detection
Random Forest	0.80	0.75	Strong Class 1 precision/recall, moderate Class 2, poor Class 0 recall
Decision Tree	0.74	0.74	Good Class 1, Weak Class 0 performance
Naive Bayes	0.53	0.57	Reasonable Class 1, Very poor Classes 0 and 2
LSTM	0.83	0.85	Strong overall performance, weak precision on Class 0 (Hate)

Comparative Analysis

- The LSTM model demonstrated superior performance, achieving the highest accuracy (83%) and weighted F1-score (85%). It effectively captured complex patterns within the text data, as evident from its performance across all classes.

- The Random Forest model provided competitive results with an accuracy of 80% and a weighted F1-score of 75%. Its performance was particularly strong for the Offensive class.
- The Logistic Regression model achieved an accuracy of 77% but exhibited limitations in distinguishing between Hate Speech and Neutral classes.
- The Decision Tree model demonstrated moderate performance but was prone to overfitting, leading to lower generalization ability.
- The Naive Bayes model exhibited the lowest performance, indicating its inadequacy for capturing the complexity of the text classification task.

3.6. LIBRARIES

```

1  import os
2  import pandas as pd
3  import numpy as np
4  import re
5  import pickle
6  import nltk
7  import tensorflow as tf
8  import matplotlib.pyplot as plt
9  from nltk.corpus import stopwords
10 from nltk.stem import WordNetLemmatizer
11 from tensorflow.keras.models import Sequential, load_model
12 from tensorflow.keras.layers import Embedding, LSTM, Bidirectional, Dense, Dropout, SpatialDropout1D
13 from tensorflow.keras.preprocessing.text import Tokenizer
14 from tensorflow.keras.preprocessing.sequence import pad_sequences
15 from sklearn.model_selection import train_test_split
16 from sklearn.metrics import classification_report, accuracy_score
17 from sklearn.utils.class_weight import compute_class_weight
18 from sklearn.linear_model import LogisticRegression
19 from sklearn.ensemble import RandomForestClassifier
20 from sklearn.tree import DecisionTreeClassifier
21 from sklearn.naive_bayes import MultinomialNB
22 import seaborn as sns
23 from wordcloud import WordCloud

```

```

from flask import Flask, request, render_template
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.initializers import Orthogonal
from tensorflow.keras.layers import LSTM, LSTMCell
import speech_recognition as sr
from tensorflow.keras.preprocessing.sequence import pad_sequences
import pickle
import os
import librosa
from pydub import AudioSegment # For converting MP3 to WAV

```


IV. RESULT AND DISCUSSION

4.1 About dataset

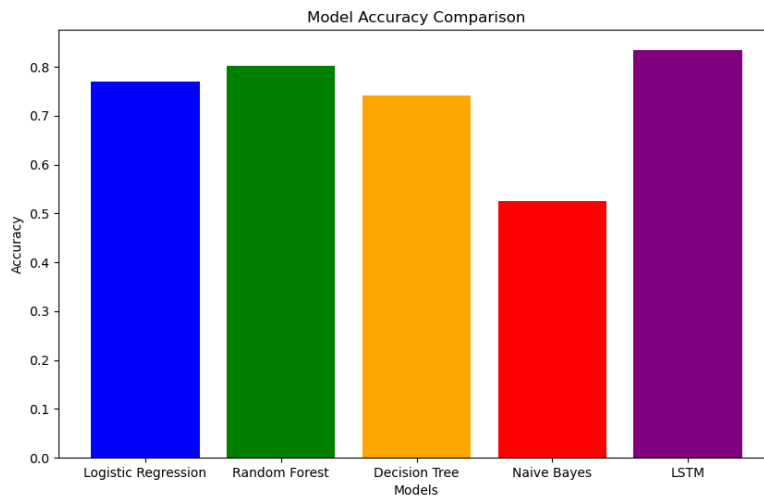
The dataset used in this project consists of both real-world labelled data and synthetically generated data to enhance model performance. The labelled dataset contains text samples categorized into three classes: **Hate Speech, Offensive Speech, and Neutral Speech**. The synthetic dataset was generated to address class imbalances and improve the model's ability to generalize across different types of hate speech.

Each text entry in the dataset includes:

- **Raw text content** (tweets or social media posts)
- **Predefined labels** (Hate, Offensive, Neutral)

4.2 Performance of the model and Comparison

The performance of different machine learning models for hate speech detection was evaluated based on their accuracy. The bar chart above compares the accuracy of five models: Logistic Regression, Random Forest, Decision Tree, Naïve Bayes, and LSTM.



Key Observations:

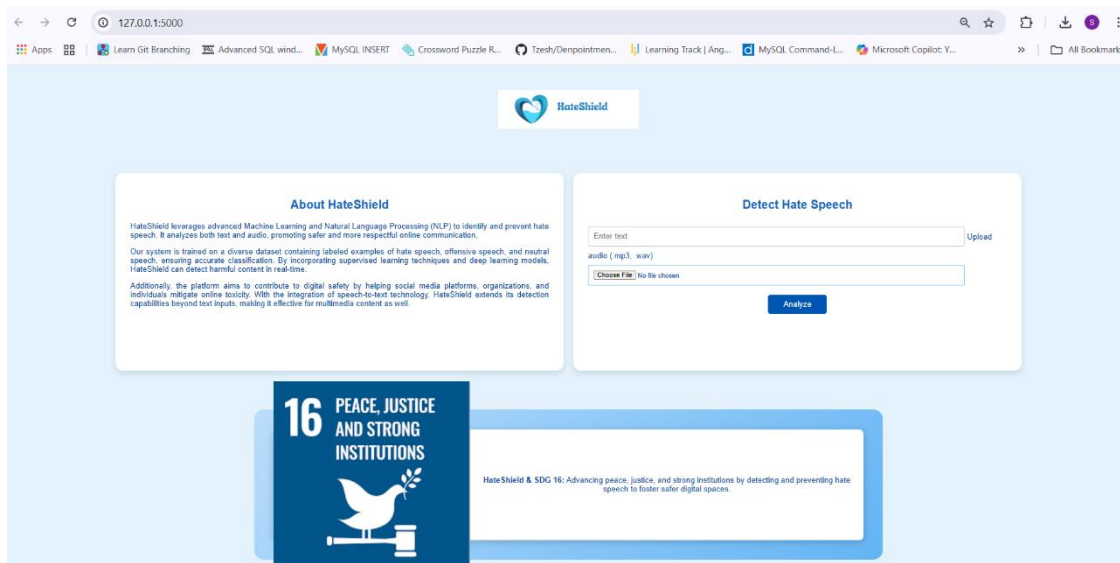
- LSTM (Long Short-Term Memory) achieved the highest accuracy, making it the most effective model for this task. This result highlights the power of deep learning in capturing sequential patterns in text.

- Random Forest performed better than Logistic Regression and Decision Tree, indicating that ensemble methods are beneficial for classification tasks.
- Decision Tree had slightly lower accuracy than Logistic Regression, showing that a single tree-based model may not generalize well compared to ensemble techniques.
- Naïve Bayes performed the worst, likely due to its strong independence assumptions, which may not hold well for complex hate speech detection tasks.

From this comparison, it is evident that deep learning models (LSTM) outperform traditional machine learning models for hate speech detection. However, Random Forest and Logistic Regression also provide competitive performance, making them viable options for less computationally expensive implementations.

4.3 Screenshots of output

```
C:\Users\sujill\OneDrive\Documents\Hate_speech>python app.py
2025-03-29 22:18:04.916699: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-03-29 22:18:06.615938: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-03-29 22:18:19.597366: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
[+] Model loaded successfully!
[+] Model loaded successfully!
[+] Tokenizer loaded successfully!
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
2025-03-29 22:18:21.869846: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-03-29 22:18:22.555814: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-03-29 22:18:36.662179: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
[+] Model loaded successfully!
[+] Model loaded successfully!
[+] Tokenizer loaded successfully!
* Debugger is active!
* Debugger PIN: 381-835-138
```



Text output

Detect Hate Speech

Upload audio (.mp3, .wav)

No file chosen

Prediction: Neutral

Detect Hate Speech

Upload audio (.mp3, .wav)

No file chosen

Prediction: Hate Speech

Detect Hate Speech

these bitches even worst they'll send them guys for you

Upload audio (.mp3, .wav)

Choose File

No file chosen

Analyze

Prediction: Offensive Language

Audio output:

Detect Hate Speech

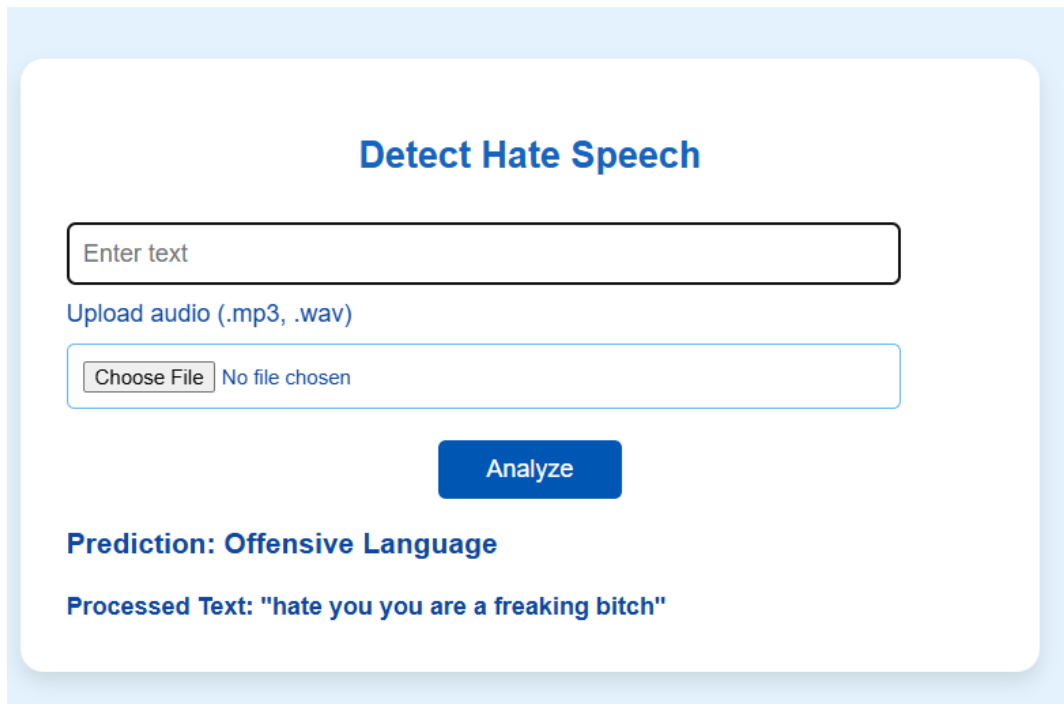
Enter text

Upload audio (.mp3, .wav)

Choose File

hate1.mp3

Analyze



Detect Hate Speech

Enter text

Upload audio (.mp3, .wav)

Choose File No file chosen

Analyze

Prediction: Offensive Language

Processed Text: "hate you you are a freaking bitch"

4.4 Challenges and limitations

- **Data Imbalance** – Hate speech examples are often fewer than neutral or non-offensive content, making it harder for models to learn effectively.
- **Context Understanding** – Sarcasm, slang, and indirect hate speech can be difficult for models to interpret correctly.
- **Bias in Training Data** – If the dataset contains biases, the model may unfairly target certain groups or fail to detect hate speech accurately.
- **Real-time Processing** – Advanced models like LSTM require more computational power, which can slow down real-time detection.
- **False Positives and Negatives** – Some neutral content may be wrongly classified as hate speech, while actual hate speech may be missed.

Drive link of the dataset and code uploaded:

<https://drive.google.com/drive/folders/1u1itVvrVCsE6zU47cqLS76vAu9ppfZ-?usp=sharing>

V. CONCLUSION

Hate speech detection using machine learning algorithms such as Naive Bayes is a promising approach for identifying and classifying offensive language online. Through the analysis of various features and the training of the model on a large dataset of labelled data, Naive Bayes can accurately classify text into hate speech , offensive or Neutral categories. However, it is important to note that the effectiveness of hate speech detection using Naive Bayes, or any other machine learning algorithm heavily relies on the quality and diversity of the dataset used for training. Therefore, it is crucial to carefully curate the training dataset and ensure that it accurately represents the diverse types of hate speech that can be encountered in different contexts and cultures. Additionally, it is important to consider the ethical implications of using machine learning for hate speech detection, such as the potential for algorithmic biases and the impact on free speech. Therefore, it is crucial to develop and apply these tools in a responsible and ethical manner, taking into account the broader social, cultural, and political context.

REFERENCES

1. Fortuna, P., Nunes, S., & Rodrigues, P. (2018). A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4), 1-30.
2. Bhatia, P., Jain, R., & Kar, S. (2020). Automatic detection of hate speech: A survey. *Journal of Ambient Intelligence and Humanized Computing*, 11(9), 3837-3855.
3. Thakur, V., & Jain, A. (2020). A review on hate speech detection using machine learning techniques. *Journal of Ambient Intelligence and Humanized Computing*, 11(11), 5021-5034.
4. Schmidt, A., Wiegand, M., & Fox, C. (2017). A survey on hate speech detection using natural language processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media* (pp. 1-10).
5. Kwok, I., & Wang, Y. (2013). Locate the hate: Detecting tweets against blacks. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1781-1791).
6. Davidson, T., Warmusley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Proceedings of the 11th International AAAI Conference on Web and Social Media* (pp. 512-515).
7. Ko, T., Peddinti, V., Povey, D., & Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 5220-5224.
8. Macavaney, S., Dori-Hacohen, S., Yao, H. R., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1097-1100.