MACHINE LEARNING ALGORITHMS FOR HATE SPEECH DETECTION

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE300 / INT300 / ICT300 - MINI PROJECT

Submitted by

KESHAV K

(Reg. No.: 126003131, B. Tech. Computer Science & Engineering)

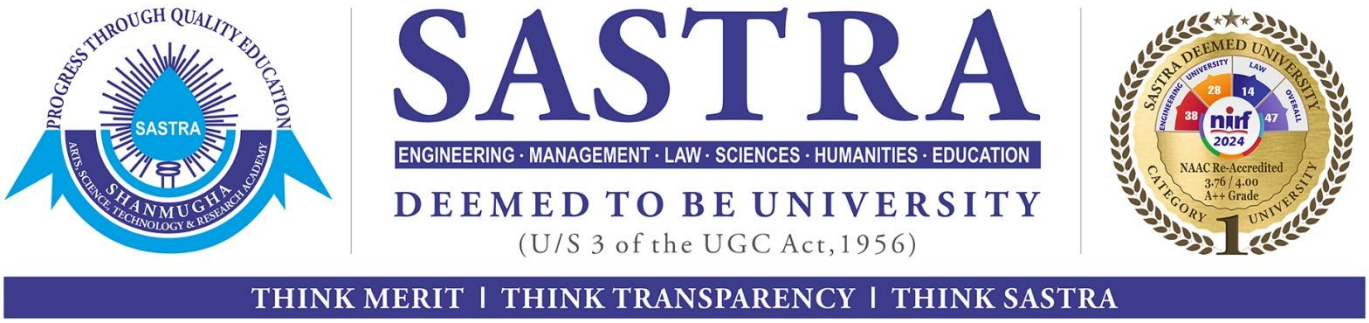
RAAHUL KUMAR N

(Reg. No.: 126003209, B. Tech. Computer Science & Engineering)

SHRUTI RAJESH

(Reg. No.: 126156144, B. Tech. in Computer Science & Engineering (Specialization in Artificial Intelligence and Data Science))

May 2025



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled " **Machine Learning Algorithms For Hate Speech Detection**" submitted as a requirement for the course, CSE300 / INT300 / ICT300: MINI PROJECT for B.Tech. is a Bonafide record of the work done by Mr. Keshav K (Reg. No.: 126003131, B. Tech. Computer Science & Engineering), Mr. Raahul Kumar N (Reg. No.: 126003209, B. Tech. Computer Science & Engineering), Ms. Shruti Rajesh (Reg. No.: 126156144, B. Tech. in Computer Science & Engineering (Specialization in Artificial Intelligence and Data Science)) during the academic year 2024-25, in the School of Computing, under my supervision.

Signature of Project Supervisor : N. Senthilselvan

Name with Affiliation : Dr. N. SENTHILSELVAN, Assistant Professor,CSE/SOC

Date:

Mini Project Viva voce held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS

We would like to thank our Honorable Chancellor Prof. R. Sethuraman for providing us with ample resources, opportunities, infrastructure and time for carrying out this project as a part of our course and curriculum.

We would like to thank our Honorable Vice-Chancellor Dr. S. Vaidhyasubramaniam and Dr. S. Swaminathan, Dean, Planning & Development, for the constant cheer, merry and any support at the time of need and necessity.

We extend our sincere thanks to Dr. R. Chandramouli, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to Dr. V. S. Shankar Sriram, Dean, School of Computing, Dr. R. Muthaiah, Associate Dean, Research, Dr. K.Ramkumar, Associate Dean, Academics, Dr. D. Manivannan, Associate Dean, Infrastructure, Dr. R. Alageswaran, Associate Dean, Students Welfare.

Our guide Dr. N. Senthil Selvan, B.Tech., M.E., Ph.D., School of Computing was the driving force and the push that we needed throughout this project. His valuable teachings in the field and invaluable opinions and suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing us an opportunity to showcase our skills through project.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

UN	United Nations
ESRC	Economic and Social Resource Council
CSV	Comma Separated Values
TF-IDF	Term Frequency-Inverse Document Frequency
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
XGBoost	Extreme Gradient Boosting
SVM	Support Vector Machine

ABSTRACT

Over the last two decades the proliferation of technology has enabled humans to connect with each other better through social media platforms and people exercise their freedom of speech to the fullest. With the increased platforms to share their opinions, hate speech has also flourished. The primary aim of this paper is to review Machine Learning Algorithms and Techniques for detecting such hate speech in social media. In this article, the proposed technique examines the baseline components for hate speech classification - exploration of the selected dataset, feature extraction, feature selection by reduction of dimensions, hate speech classifier selection and training, and model evaluation. In this literature, there is a plan on proposing different ensemble approaches using classical Machine Learning methods, and utilizing different performance metrics to evaluate the approaches such as Precision, F1 scores, Accuracy, and Recall parameters. The results from these will be assessed to have an estimation of what the best model among the approaches used will be, alongside their performance parameters to improve and stratify using a better algorithm. This study will have three major contributions. Firstly, it would equip the readers with all the critical steps involved in hate speech detection. Secondly, it would give a thorough analysis on all the models involved. Third, it would include any gaps in research and challenges that will be identified in the process. This study and implementation can immensely help learners, researchers, the public and professionals alike.

Keywords: Classification Algorithms, Feature Extraction, Machine Learning Algorithms, Text Classification, Cyber Hate, Ensemble Technique, Social Media Networks.

TABLE OF CONTENTS

CHAPTER 1

SUMMARY OF THE BASE PAPER

Introduction:

Hate speech on social media has become a growing concern, fostering online toxicity and misinformation. Usually propagated under the guise of anonymity, hate speech does detrimental damage to anyone involved explicitly in the exchange and implicitly coming across the interaction. Hate speech goes beyond just trying to ruin someone's temper. It is deeply impactful in relation to people's perception of a particular class, race, gender, religion, sexuality and/or any socially non-conforming minority groups. This gives rise to fascistic ideas among people and works towards going against the very idea of freedom of speech. It targets the powerless, and hence, it is very essential to remove any hate speech with the intent of protecting harmony and eradicating social imbalance. Misinformation and derogatory language are sometimes hidden behind the harmful implications and cannot be identified in plain sight. Hate speech ranges from explicit slurs and death threats to generalization and false presumptions of marginalized communities. This demands a model that can stratify online information and detect hate with high precision without infringing on the right to freedom of speech. Inaccuracies can lead to unnecessary censorship of content without malice.

Addressing hate speech often gets conflated with the idea of limiting free speech or to enforce censorship to uphold a bias or an agenda. However, addressing hate speech does not mean infringing on people's right to freedom of expression . This work only aims to target rampant aggression against the feeble and marginalized. In the case of Pravasi Bhalai Sangathan v Union of India (2014), which aimed to question whether the legal framework was sufficient to efficiently curb hate speeches made by political leaders and by those in higher social power, the Supreme Court provided a definition characterizing hate speech as an attempt to marginalize individuals based on their group membership. This involves an endeavour to delegitimize group members in the eyes of the majority, ultimately diminishing their social standing and acceptance within society [1]. Countries with very weak democracy are prone to the damages and the collateral charges that come with the cost of free speech. It is important that a diplomatic narrative in social media networks is maintained so that it is easier to mitigate baseless hatred and distinguish it from actual feedback that aims on addressing multiple issues.

Having the Internet accessible across every region in this world intensifies racism by a thousand-fold and this serves as one of the primary motivations to combat hate speeches by appropriate machine learning algorithms. Between 2020 and 2023, in partnership with the ESRC Human Rights, Big Data and Technology Project at the University of Essex organized four roundtable consultations to discuss the implementation of the United Nations Strategy and Plan of Action on Hate Speech in the online space. The roundtables engaged technology and social media companies on their role and responsibility to address online hate speech in line with international human rights norms and standards, and with the view to building partnerships for the implementation of the UN Strategy and Plan of Action on Hate Speech. In compliance with the UN and the ESRC Human Rights, this project aims to contribute to this cause and reduce the impact of hate speech which includes not only racism but homophobia, xenophobia, casteism, classism, colourism, ableism and a lot of other social stigmata that are ought to be removed effectively.

Competitions like the SemEval 2019 have conducted multiple workshops on Natural Language Processing (NLP) and worked on tasks that helped detect multilingual hate speech against immigrants and women using a Twitter (now known as X) dataset. [2] The task that GermEval 2021 conducted helped identify toxic, engaging and fact-claiming comments exclusively using a German dataset. [3] Events, competitions and workshops like the ones mentioned have helped NLP take a huge leap forward, particularly in hate speech and cybercrime detection, which ricochets advancements in machine learning (ML) and deep learning towards better classification of content in online blogs and social media. This project explores deep learning ensemble techniques to improve hate speech detection accuracy. Automated detection methods using machine learning and deep learning have proven effective in identifying harmful content. However, individual deep learning models often struggle with generalization due to data biases and class imbalances. By combining multiple models, ensemble learning leverages their strengths to enhance performance. This approach involves preprocessing text data, training various deep learning models, and integrating them using techniques like stacking and weighted averaging. The models are evaluated using standard metrics such as precision, recall, F1-score, and accuracy to ensure effectiveness.

Key challenges in hate speech detection include covert aggression, sarcasm, implicit bias, and contextual ambiguity. Overt Aggression contains words that have lexical or syntactic structures that are universally recognized as derogatory and discriminatory. However, covert aggression and sarcastic hate speech do not contain any explicit words that can instantly be recognized. Statements of the latter kind must be contextualized to properly discover the intent. Hate speech is an ever-expanding dataset. New vocabulary and slang are constantly added by linguists and it is best to be on the lookout for perennially changing cultural conflicts to better understand empathy and social inclusivity. This project aims to refine existing methods by implementing a robust ensemble framework that improves classification reliability. The findings contribute to developing more accurate and efficient hate speech detection systems, making social media platforms safer and more inclusive.

Proposed methodology:

The machine was trained using a curated benchmark dataset [4] that consists of 726119 number of tuples in a CSV document. The document essentially contains a balanced combination of overtly and covertly aggressive comments and non-hateful comments. It involves binary labelling and is meant to be utilized for the purposes of training machine learning, deep learning and ensemble models. It reflects current social media trends and modern ways of writing hateful speech using specific slangs, emojis and emoticons. The dataset is neutralized and does not contain information regarding the users who posted the content. However, there were multiple trial datasets considered because machine learning is heavily dependent on the dataset as well and it’s extremely futile to use redundant data samples. The datasets proved to be overfit for the model and were discarded after several reviews and assessments. For example, dataset [5] consists of 998 sample comments alongside binary labelling. Upon testing and validating over multiple machine learning architectures, it was observed that there was no scope of improving the precision and even after augmenting the dataset with its corresponding synonyms and equivalent words, with a 50% probability of choosing between the original data and the synonyms intended to prevent overfitting, the precision was compromised with an obvious accuracy recession.

Hence, it is important to choose the right dataset for any model. Having a dataset large enough to be representative of all test cases is necessary, particularly talking about hate speech detection and elimination because this is an exceptionally broad text classification. Sampling of data over multiple independent reliable sources is pivotal to touch bases on all grounds of hate and bigotry. Hence dataset [4] proves to be characteristically perfect data sample for this cause.

The dataset consists of very unstructured text samples. Machine learning algorithms utilize mathematical modelling as a way to find patterns and mine knowledge from the same. Cleaning the data tuples to remove all stop words like “is”, “was” and the unessential tokens and vectorization of the data sample to create a vector space for the purposes of processing is the next step towards training the model. TF-IDF vectorizer is used for this reason. TF-IDF helps in converting a large collection of text documents into a TF-IDF matrix. This matrix numerically represents the importance of each term in each document. This serves to be very useful for the classifiers.

Out of the plethora of classifiers, there is an imperative need to choose the one which works well in collaboration with the dataset chosen. The machine learning models chosen along with the ones mentioned in the reference base paper (Refer Appendix) are as follows:

1. Stacked LSTM layers (LSTM x2): Stacked LSTM helps in providing a deeper context to text sequence inputs and contains multiple LSTM layers which provide an output sequence to the next layer. This helps in gathering accurate predictions about the data sample by cumulatively gathering the observations from multiple LSTM layers. Traditional RNNs struggle to maintain information in the long term due to the vanishing gradient problem. Stacking LSTMs help us alleviate this by using multiple gates to help retain sequences in the long term. Stacking also helps us in working with data in higher dimensions with improved feature extraction. This is a very lightweight classifier and performs the epochs in about thirty to forty minutes without any interference. The model has very good F1 score and gives decent results on testing. Accuracy: 87%

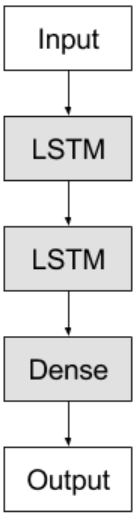


Figure 1.1 Stacked LSTM layers

2. Bidirectional GRU: It is also a sequence processing model much like LSTM but with additional memory and context. It consists of two GRUs – one for the sequence in forward and the other for the backward. It contains only the input and the forget gates. The window size is higher at any particular moment. By making it bidirectional, the context of a given word will now depend not only on the previous words in the data sample but also the successive ones. The motive behind this is to find more interesting patterns that correlate with the concept of hate speech. The results were promising with a very decent F1 score. The epochs were again performed in about thirty minutes similar to the previous model tested. Accuracy: 87%

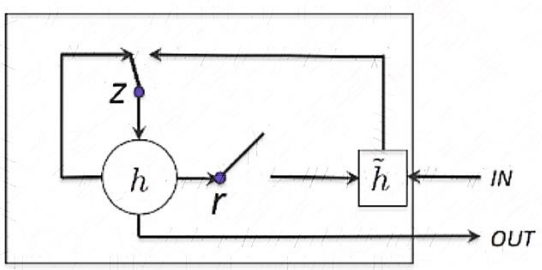


Figure 1.2 Gated Recurrent Unit

3. An ensemble of Random Forest, Gradient Boost and XGBoost classifiers: All the classifiers used in this ensemble utilize decision trees for predicting the label of the data sample. Random forest builds multiple parallel decision trees to train subsets of the dataset. Gradient Boost sequentially builds decision trees where the successor decision tree helps resolve the error made by its previous one. XGBoost is a highly optimized version of Gradient boosting and uses multiple implementation techniques like parallelization and regularization to improve its efficiency. It has reduced overfitting and usually produces a relatively high accuracy. In the ensemble, the individual performances of Gradient Boosting and Random Forest show multiple false positive cases. By using a voting classifier ensemble, the results are leveraged by giving each model a chance to vote for the test data set while evaluation. Accuracy: 83%

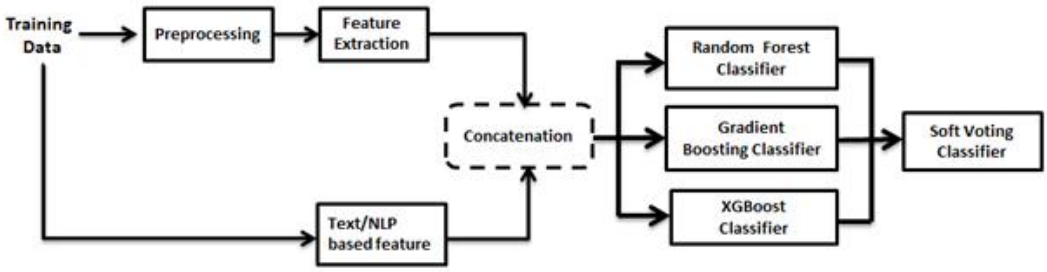


Figure 1.3 Ensemble of Random Forest, Gradient Boost and XGBoost Classifiers

CHAPTER 2

MERITS AND DEMERITS

Related Works:

In most works like [6] by Zeerak Waseem and Dirk Hovy, the main intention was to reduce the hate speech occurrence by trying to identify the gender and geographical information of the perpetuator in order to bring out better interesting patterns using *Logistic Regression*. Despite going beyond just looking for the similarities, semantics, and length along with 1-4 grams of the tuples in the dataset, the model yields very low F1 score and requires better coverage.

Other related works like [7] by Bouazizi M and Otsuki T aims to stratify tweets in the dataset by manually annotating them and scoring their sarcasm levels. It is important to understand that a lot of interpretations in regards to sarcasm is bound to be imperfect and cannot be taken as an absolute claim for the purposes of classification. For reasons such as conflict between sentiment and interpretation, the *Random Forest* model falls short.

In the research [8] by Davidson T and Warmsley D, the intention is broadly the same. However, the *Support Vector Machine* falls short because of the one-versus-rest framework where a separate classifier is trained for each class and the class label with the highest predicted probability across all classifiers is assigned to each tweet. It falsely classifies 40% of the hate speech in the dataset. The problem lies in the inability to distinguish between harmless tweets that contain offensive languages and the actual slurs used against the marginalized. In similar works like [9] by Bertie Vidgen and Taha Yasseri, the *SVM* is inefficient in classifying tweets based on the hate intensity of islamophobia. For instance, out of 100 tweets which are labelled as strong Islamophobic, 23 are actually weak. Similarly, out of 100 tweets which are labelled as weak Islamophobic, 22 are actually non- Islamophobic.

In papers like [10] by Pete Burnap and Matthew L Williams, it was inferred that there's a blend or intersectionality of bigotry and hate speeches in each tuple of a dataset. Hence multiple classifiers were utilized to categorize hate speeches that targeted different groups (disability, race and sexual orientation) and a multi-class classifier was also used in categorizing distinct types of hatred. But the model gives a very poor F1 score of 0.51 in detecting hate speeches targeting sexual orientation in particular. In [11] by Binny Mathew, et al., there is a heavy utility of neural networks in classifying hate speeches but the limitations still persist. Classification of misogynist posts have a low F1 score of 0.39.

In [12] by Anusha M D and H L Shashirekha, the task of classification is divided into two subtasks A and B. Subtask A deals with the binary classification of the given dataset into either hateful or non-hateful. Subtask B deals with intricacies within the hate posts classified in the former subtask. Subtask B deals with classifying the posts as Hate, Profanity or Offense. The extracted features from the dataset are used to train *Random Forest, Gradient Boosting and XGBoost Classifier* models to identify hate speech and offensive content. Statistics of the dataset used show that for English Subtask A, both training and development set are balanced but German and Hindi training and development set are imbalanced. Further, for English Subtask B training and development set are heavily imbalanced and German and Hindi training and development set are not

balanced. Since the classifier is heavily dependent on the dataset used, there is no claiming that one model is better than the rest which is the reason why an ensemble approach is usually preferred because weakness of one model can be compensated with another.

Merits and Demerits:

Table 2.1 Performance metric scores of each classifier

Model \ Weighted Average	F1-Score	Accuracy	Precision	Recall
lstm2x	0.86	0.86	0.86	0.86
biGRU	0.87	0.87	0.87	0.87
Random Forest	0.85	0.85	0.85	0.85
Gradient Boost Classifier	0.68	0.7	0.74	0.7
xg boost	0.79	0.79	0.79	0.79
Ensemble	0.83	0.84	0.84	0.84

From the table, it is easy to note that Bidirectional-GRU (biGRU) on average has a better score on all assessments and regards. biGRU has a larger window size relative to other classifiers. This allows the classifier to look for patterns in both directions of the text sequence without any hassle for memory. GRUs also have a simpler structure and don't require a lot of computational time. This makes biGRU the most optimal to work with. However, because biGRU has an increased window size, it takes on a lot. It requires an increased computational load and takes more time in comparison to unidirectional GRU or LSTM x2, especially for longer sequences. Interpretation from the predictions made by the biGRU can also be quite complex because it is difficult to pinpoint which sequence in a particular direction has contributed more to the prediction made. Lack of representation of any specific hate speech can cause the classifier to fall short because it is more prone to overfitting than one might think. It is important that the classifier is fed with a dataset that is as large as the one chosen.

Training deep LSTM models on large datasets is challenging and time consuming because it requires powerful hardware like GPUs and TPUs. On the other hand, it does help in capturing the required sequential patterns and interpretations to predict hate speeches accurately. Because of its narrower window of memory and the ability to not scan patterns on both the directions like the biGRU, it does not have as much accuracy as the previous classifier. But that doesn't take away from the fact that LSTMx2 is exceptional at NLP tasks and stratification, given the quality of the dataset chosen.

Random Forest is an extremely strong classifier and has multiple advantages over disadvantages. It is bound to give a high accuracy as all the distinct subsets of the data samples are trained parallelly and their predictions are aggregated to reduce the overall variation. It works best in an ensemble approach because on an off-chance that Random Forest falls short with its prediction, the other two models in collaboration can help compensate for the loss of accuracy. Random Forest used here has a decent score and is also better at being a standalone classifier outside of an ensemble. It is resilient to the noisy data and not one noisy datum has a hand in influencing the entire performance.

It is non-parametric and does not impose rigid limitations to the data samples. The disadvantage is that it is very taxing, time and memory wise, due to the number of trees in the forest. It takes a lot longer to predict the outcome with Random Forest than an average classifier.

Gradient Boosting and XGBoost, albeit excellent models, lack the ability to identify complex patterns in the sequence which poses a major threat to the accuracy of the model. In an ensemble of Random Forest, Gradient Boosting and XGBoost classifiers, the conflicts and the merits offset one another and yields us a decent precision and F1 score. The ensemble is also particularly heavy especially due to the Random Forest model. It is worth noting that the reliability of the ensemble is not on par with biGRU or LSTM x2.

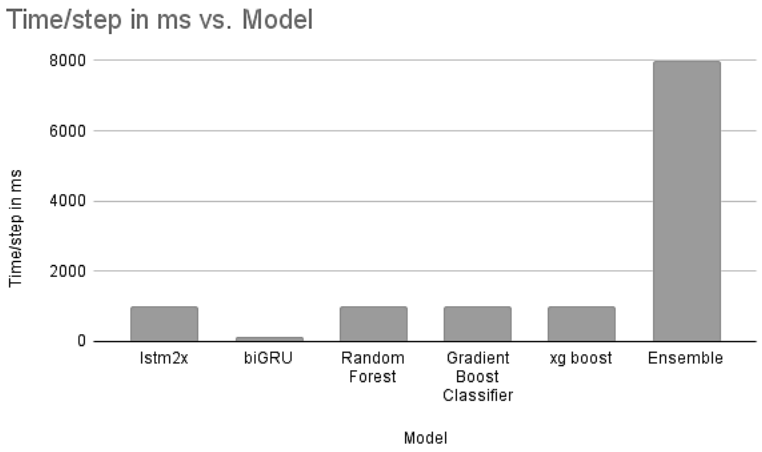


Figure 2.1 Time taken by each classifier (in ms)

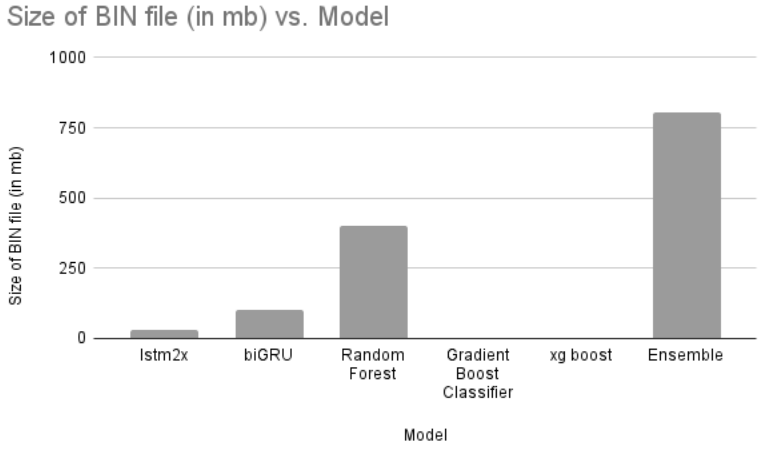


Figure 2.2 Size of binary file required for each model

CHAPTER 3

SOURCE CODE

LSTM x2 (Stacked LSTM):

```
#Import Necessary Libraries

!pip install nltk
!pip install scikit-learn

import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import  Input,Dense,Embedding, LSTM,Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from nltk.corpus import stopwords
import re
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords

# Load Dataset
file = "/kaggle/input/hate-speech-detection-curated-dataset/HateSpeechDatasetBalanced.csv"
df = pd.read_csv(file)
df.shape
>  (726119, 2)
df['Label'].unique()
>  array([1, 0])

# Preprocessing Text
nltk.download('stopwords')
stp = set(stopwords.words("english"))
# Function to preprocess text
def preprocess_text(text):
    text = re.sub("[^a-zA-Z]", " ", text)
    text = text.lower()
    text = text.split()
    words = [word for word in text if word not in stp]
    return " ".join(words)
df['ProcessedContent'] = df['Content'].apply(preprocess_text)
```

```
# Tokenization
from keras._tf_keras.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['ProcessedContent'])
sequences = tokenizer.texts_to_sequences(df['ProcessedContent'])
max_len=max([max(0,len(seq)) for seq in sequences])
max_len
> 294

# Padding Sequences
#Ensures all sequences are of same length
from keras._tf_keras.keras.preprocessing.sequence import pad_sequences
x = pad_sequences(sequences, maxlen=max_len)
y = df['Label'].values
print(x.shape)
print(y.shape)
>  (726119, 294)
> (726119,)

# Splitting into Train and Test Set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
> (580895, 294)
> (145224, 294)
> (580895,)
> (145224,)

# Model Creation
import random
random.seed(42)
np.random.seed(42)
tf.random.set_seed(42)
#Creating the model
i = Input(shape=(x_train.shape[1],))
x = Embedding(input_dim=len(tokenizer.word_index)+1,output_dim=20)(i)
x = LSTM(10,return_sequences=True)(x)
x = LSTM(10,return_sequences=True)(x)
x = Flatten()(x)
x = Dense(1,activation='sigmoid')(x)
model = Model(i,x)
```

```
# Using Adam optimizer
opt = tf.keras.optimizers.Adam(learning_rate=0.01)

# Compiling the model
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# Fitting the model
history = model.fit(x_train, y_train, epochs=7, validation_data=(x_test, y_test),batch_size=64)
```

```
> Epoch 1/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 185s 20ms/step - accuracy: 0.8106 - loss:
0.4025 - val_accuracy: 0.8608 - val_loss: 0.3145
> Epoch 2/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180s 20ms/step - accuracy: 0.8818 - loss:
0.2722 - val_accuracy: 0.8664 - val_loss: 0.3105
> Epoch 3/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180s 20ms/step - accuracy: 0.9017 - loss:
0.2319 - val_accuracy: 0.8653 - val_loss: 0.3211
> Epoch 4/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180s 20ms/step - accuracy: 0.9092 - loss:
0.2135 - val_accuracy: 0.8661 - val_loss: 0.3253
> Epoch 5/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180s 20ms/step - accuracy: 0.9143 - loss:
0.2015 - val_accuracy: 0.8646 - val_loss: 0.3415
> Epoch 6/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180s 20ms/step - accuracy: 0.9168 - loss:
0.1961 - val_accuracy: 0.8614 - val_loss: 0.3550
> Epoch 7/7
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179s 20ms/step - accuracy: 0.9168 - loss:
0.1968 - val_accuracy: 0.8627 - val_loss: 0.3491
```

```
# Analyzing Training History
plt.plot(history.history["accuracy"],label="training accuracy")
plt.plot(history.history["val_accuracy"],label="testing accuracy")
plt.title("model accuracy")
plt.xlabel("epoch")
plt.ylabel("accuracy")
plt.legend()
plt.show()
plt.plot(history.history["loss"],label="training loss")
plt.plot(history.history["val_loss"],label="testing loss")
plt.title("model loss")
```

```
plt.xlabel("epoch")
plt.ylabel("loss")
plt.ylim((0,0.6))
plt.xlim((0,7))
plt.legend()
plt.show()

## Model Evaluation
y_pred = model.predict(x_test)
y_pred.shape
> (145224, 1)
y_pred = y_pred.squeeze()
y_pred_binary = np.where(y_pred>0.5,1,0)
y_pred_binary
> array([1, 1, 0, ..., 1, 0, 1])

loss,accuracy = model.evaluate(x_test, y_test)
print("model loss",loss)
print("model accuracy",accuracy)
>  model loss 0.34910908341407776
> model accuracy 0.8626880049705505

from sklearn.metrics import accuracy_score,confusion_matrix
print(f"The accuracy score is:{accuracy_score(y_pred_binary,y_test)}")
> The accuracy score is:0.8626879854569492

# Confusion Matrix
import seaborn as sns
con= confusion_matrix(y_pred_binary,y_test)
sns.heatmap(con,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Classification Report
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_binary))

import pickle
#Save the iris classification model as a pickle file
model_pkl_file = "ltsm2x_model.pkl"
with open(model_pkl_file, 'wb') as file:
    pickle.dump(model, file)
with open('tokenizer.pkl', 'wb') as f:
```

```
pickle.dump(tokenizer, f)

from IPython.display import FileLink
FileLink(r'tokenizer.pkl')
```

Bidirectional-GRU:

```
# Hate Speech Detection using Bidirectional GRU on a balanced dataset
# Import Necessary Libraries
!pip install nltk
!pip install scikit-learn
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Embedding, Bidirectional, LSTM, GRU, GlobalMaxPooling1D,
Dropout, Dense
from tensorflow.keras.models import Model
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from nltk.corpus import stopwords
import re
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords

# Load Dataset
file = "/kaggle/input/hate-speech-detection-curated-dataset/HateSpeechDatasetBalanced.csv"
df = pd.read_csv(file)
df.shape
> (726119, 2)
df['Label'].unique()
> array([1, 0])

## Preprocessing Text
nltk.download('stopwords')
stp = set(stopwords.words("english"))

# Function to preprocess text
def preprocess_text(text):
    text = re.sub("[^a-zA-Z]", " ", text)
    text = text.lower()
```

```
    text = text.split()
    words = [word for word in text if word not in stp]
    return " ".join(words)
df['ProcessedContent'] = df['Content'].apply(preprocess_text)

# Tokenization
from keras._tf_keras.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['ProcessedContent'])
sequences = tokenizer.texts_to_sequences(df['ProcessedContent'])
max_len=max([max(0,len(seq)) for seq in sequences])
max_len
> 294

# Padding Sequences
# Ensures all sequences are of same length
from keras._tf_keras.keras.preprocessing.sequence import pad_sequences
x = pad_sequences(sequences, maxlen=max_len)
y = df['Label'].values
print(x.shape)
print(y.shape)
> (726119, 294)
> (726119,)


# Splitting into Train and Test Set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
> (580895, 294)
> (145224, 294)
> (580895,)
> (145224,)

# Model Creation
import random
random.seed(42)
np.random.seed(42)
tf.random.set_seed(42)
```

```
from tensorflow.keras.layers import Input, Embedding, Bidirectional, GRU, GlobalMaxPooling1D, Dropout, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import f1_score
import tensorflow as tf

# Model
i = Input(shape=(x_train.shape[1],))
x = Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=64)(i)
x = Bidirectional(GRU(64, return_sequences=True))(x)
x = GlobalMaxPooling1D()(x)
x = Dropout(0.5)(x)
x = Dense(1, activation='sigmoid')(x)
model = Model(i, x)

# Compile
model.compile(loss="binary_crossentropy", optimizer=Adam(0.001), metrics=["accuracy"])

# EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)

# F1-score callback
class F1Callback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        val_pred = (self.model.predict(x_test) > 0.5).astype(int)
        f1 = f1_score(y_test, val_pred)
        print(f" — val_f1: {f1:.4f}")


# Fit
history = model.fit(
    x_train, y_train,
    epochs=10,
    batch_size=64,
    validation_data=(x_test, y_test),
    callbacks=[early_stop, F1Callback()]
)
```

> Epoch 1/10
4539/4539 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35s 8ms/step
 — val_f1: 0.8609
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 279s 30ms/step - accuracy: 0.8038 - loss: 0.4159 - val_accuracy: 0.8592 - val_loss: 0.3168
> Epoch 2/10
4539/4539 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35s 8ms/step
 — val_f1: 0.8726
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275s 30ms/step - accuracy: 0.8762 - loss: 0.2854 - val_accuracy: 0.8710 - val_loss: 0.3004
> Epoch 3/10
4539/4539 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35s 8ms/step
 — val_f1: 0.8737
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275s 30ms/step - accuracy: 0.9046 - loss: 0.2268 - val_accuracy: 0.8726 - val_loss: 0.3110
> Epoch 4/10
4539/4539 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35s 8ms/step
 — val_f1: 0.8768
9077/9077 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 277s 31ms/step - accuracy: 0.9242 - loss: 0.1839 - val_accuracy: 0.8739 - val_loss: 0.3322

Analyzing Training History

```
plt.plot(history.history["accuracy"],label="training accuracy")
plt.plot(history.history["val_accuracy"],label="testing accuracy")
plt.title("model accuracy")
plt.xlabel("epoch")
plt.ylabel("accuracy")
plt.legend()
plt.show()
plt.plot(history.history["loss"],label="training loss")
plt.plot(history.history["val_loss"],label="testing loss")
plt.title("model loss")
plt.xlabel("epoch")
plt.ylabel("loss")
plt.ylim((0,1))
plt.xlim((0,5))
plt.legend()
plt.show()
```

```
## Model Evaluation
y_pred = model.predict(x_test)
y_pred.shape
> (145224, 1)
y_pred = y_pred.squeeze()
y_pred_binary = np.where(y_pred>0.5,1,0)
y_pred_binary
> array([0, 1, 1, ..., 1, 0, 1])

loss,accuracy = model.evaluate(x_test, y_test)
print("model loss",loss)
print("model accuracy",accuracy)
>  model loss 0.3003596067428589
> model accuracy 0.8709786534309387

from sklearn.metrics import accuracy_score,confusion_matrix
print(f"The accuracy score is:{accuracy_score(y_pred_binary,y_test)}")
> The accuracy score is:0.8709786261224041

# Confusion Matrix
import seaborn as sns
con= confusion_matrix(y_pred_binary,y_test)
sns.heatmap(con,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_binary))

import pickle
# Save the iris classification model as a pickle file
model_pkl_file = "bi_gru_model.pkl"
with open(model_pkl_file, 'wb') as file:
    pickle.dump(model, file)
with open('tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)

from IPython.display import FileLink
FileLink(r'tokenizer.pkl')
FileLink(r'biGRU_model.pkl')
```

Ensemble of Random Forest, Gradient Boost and XGBoost Classifiers:

```
#Ensemble of Random Forest, GBC and XGB

import pandas as pd
import numpy as np
import re
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from scipy.sparse import hstack
nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text.lower())
    tokens = text.split()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
    return ' '.join(tokens)

df = pd.read_csv('/kaggle/input/hate-speech-detection-curated-dataset/HateSpeechDatasetBalanced.csv')
df['clean_text'] = df['Content'].apply(clean_text)
tfidf = TfidfVectorizer(max_features=3000)
tfidf_features = tfidf.fit_transform(df['clean_text'])
df['text_len'] = df['clean_text'].apply(lambda x: len(x.split()))
nlp_features = df[['text_len']].values

from scipy.sparse import csr_matrix
combined_features = hstack([tfidf_features, csr_matrix(nlp_features)])
X_train, X_test, y_train, y_test = train_test_split(combined_features, df['Label'], test_size=0.2,
random_state=42)

clf1 = RandomForestClassifier(n_estimators=50, n_jobs=-1)
print("In RF")
clf1.fit(X_train, y_train)
clf2 = GradientBoostingClassifier(n_estimators=50, random_state=1, verbose=1)
print("In GBC")
clf2.fit(X_train, y_train)
```

```
clf3 = XGBClassifier(use_label_encoder=False, eval_metric='logloss', verbosity=1)
print("In XGB")
clf3.fit(X_train, y_train)


ensemble = VotingClassifier(estimators=[
   ('rf', clf1),
   ('gb', clf2),
   ('xgb', clf3)
], voting='soft')

ensemble.fit(X_train, y_train)
y_pred = ensemble.predict(X_test)
print(classification_report(y_test, y_pred))

# Confusion Matrix for the ensemble
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=ensemble.classes_)
disp = ConfusionMatrixDisplay.from_predictions(y_test,y_pred,
                        display_labels=ensemble.classes_,
                       cmap=plt.cm.Blues)
disp.plot()

plt.title("Ensemble Confusion Plot")
plt.show()

# Confusion matrix for RF
y_pred = clf1.predict(X_test)
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred, labels=clf1.classes_)
disp = ConfusionMatrixDisplay.from_predictions(y_test,y_pred,
                                    display_labels=clf1.classes_,
                                    cmap=plt.cm.Blues
                                   )
disp.plot()

plt.title("Random Forest Classifier")
plt.show()

y_pred = clf2.predict(X_test)
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred, labels=clf2.classes_)
```

```
disp = ConfusionMatrixDisplay.from_predictions(y_test,y_pred,
                          display_labels=clf2.classes_,
                         cmap=plt.cm.Blues
                         )
disp.plot()

plt.title("Gradient Boosting Classifier")
plt.show()

y_pred = clf3.predict(X_test)
print(classification_report(y_test, y_pred))



cm = confusion_matrix(y_test, y_pred, labels=clf3.classes_)
disp = ConfusionMatrixDisplay.from_predictions(y_test, y_pred,
                          display_labels=clf3.classes_,
                         cmap=plt.cm.Blues    )
disp.plot()

plt.title("XGBoost")
plt.show()

import pickle
with open('rf.pkl',"wb") as file:
    pickle.dump(clf1,file)

with open('gbc.pkl','wb') as file:
    pickle.dump(clf2,file)

with open('xgb.pkl','wb') as file:
    pickle.dump(clf3,file)

'''
clf1 = pickle.load(open('rf.pkl','rb'))
clf2 = pickle.load(open('gbc.pkl','rb'))
clf3 = pickle.load(open('xgb.pkl','rb'))
'''
with open('ensemble_rgx.pkl','wb') as file:
    pickle.dump(ensemble,file)
```

(To refer to the implementation of an interface to test the demonstration:
https://github.com/KKeshav1101/mini_project/blob/master/HateSpeechDetector/views.py**)**

CHAPTER 4

SNAPSHOTS

LSTM x2 (Stacked LSTM):

	precision	recall	f1-score	support
0	0.88	0.84	0.86	72321
1	0.85	0.89	0.87	72903
accuracy			0.86	145224
macro avg	0.86	0.86	0.86	145224
weighted avg	0.86	0.86	0.86	145224

Figure 4.1 Classification report of LSTM x2

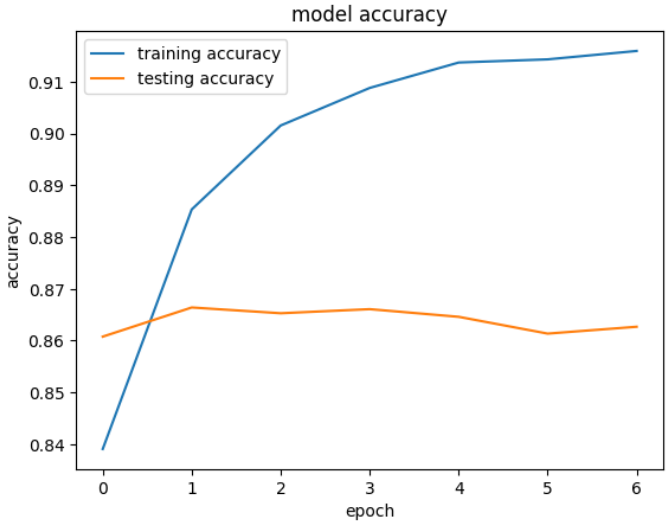


Figure 4.2 Model Accuracy of LSTM x2

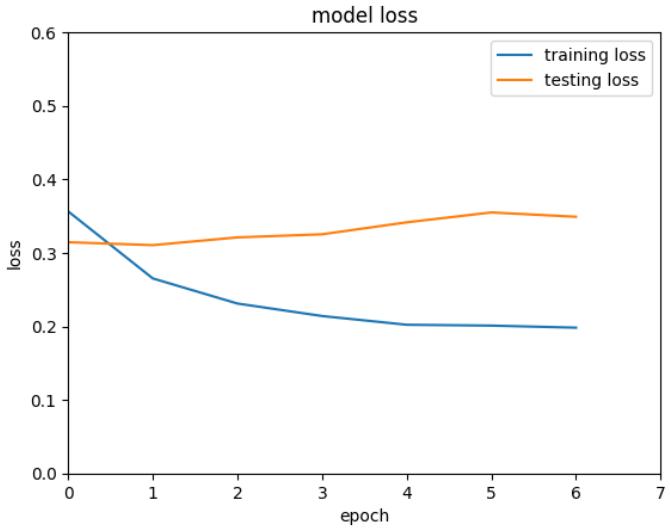


Figure 4.3 Model Loss of LSTM x2

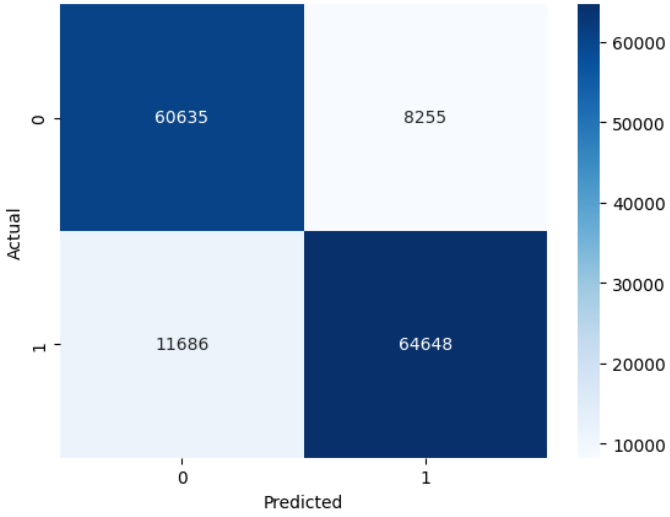


Figure 4.4 Confusion Matrix of LSTM x2

Bidirectional-GRU:

	precision	recall	f1-score	support
0	0.88	0.86	0.87	72321
1	0.87	0.88	0.87	72903
accuracy			0.87	145224
macro avg	0.87	0.87	0.87	145224
weighted avg	0.87	0.87	0.87	145224

Figure 4.5 Classification Report of Bidirectional-GRU

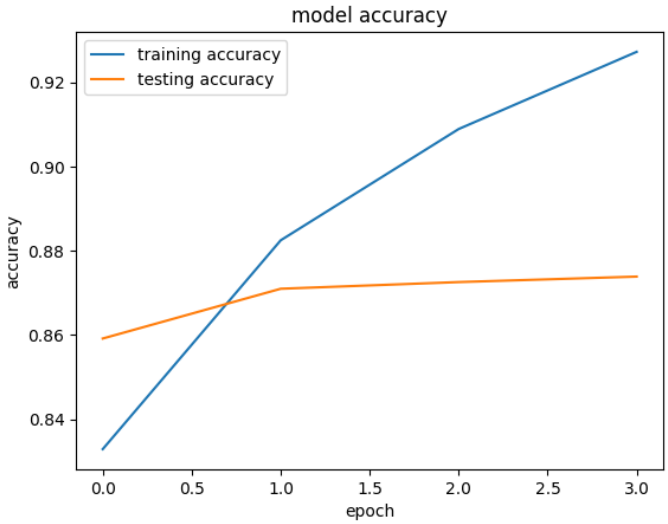


Figure 4.6 Model Accuracy of biGRU

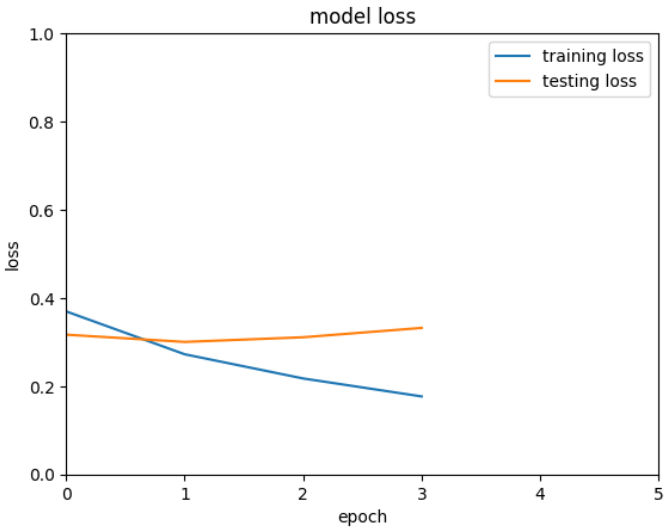


Figure 4.7 Model Loss of biGRU

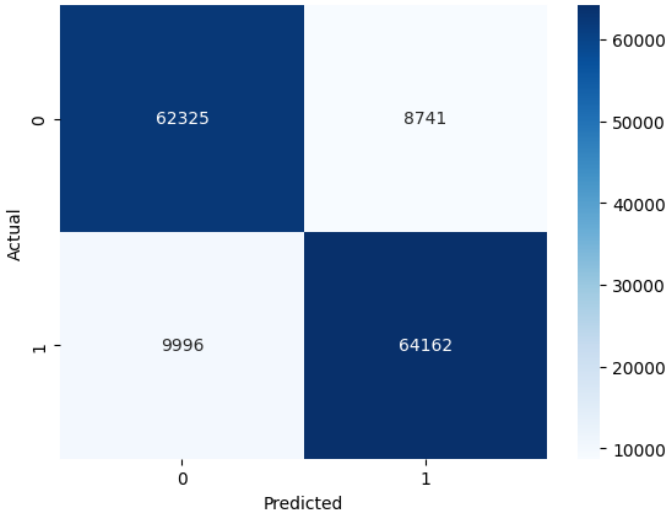


Figure 4.8 Confusion Matrix of biGRU

Ensemble of Random Forest, Gradient Boost and XGBoost Classifiers:

	precision	recall	f1-score	support
0	0.86	0.80	0.83	72043
1	0.81	0.87	0.84	73181
accuracy			0.84	145224
macro avg	0.84	0.83	0.83	145224
weighted avg	0.84	0.84	0.83	145224

Figure 4.9 Classification Report of the Ensemble

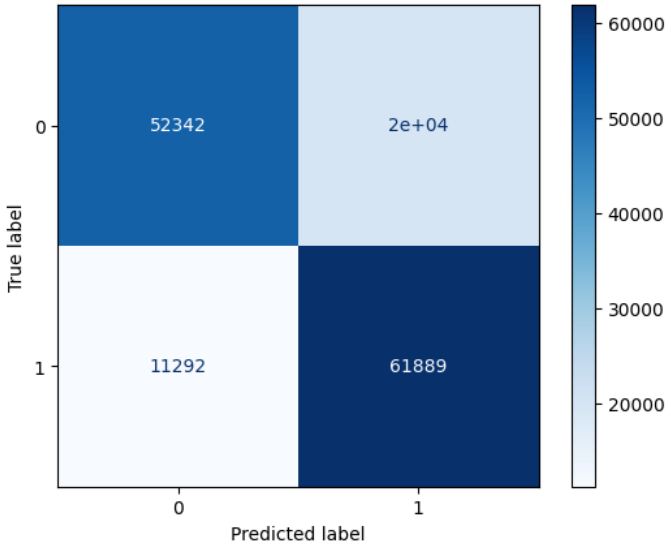


Figure 4.10 Confusion Matrix of Ensemble

	precision	recall	f1-score	support
0	0.87	0.82	0.84	72043
1	0.83	0.88	0.85	73181
accuracy			0.85	145224
macro avg	0.85	0.85	0.85	145224
weighted avg	0.85	0.85	0.85	145224

Figure 4.11 Classification Report of Random Forest Classifier

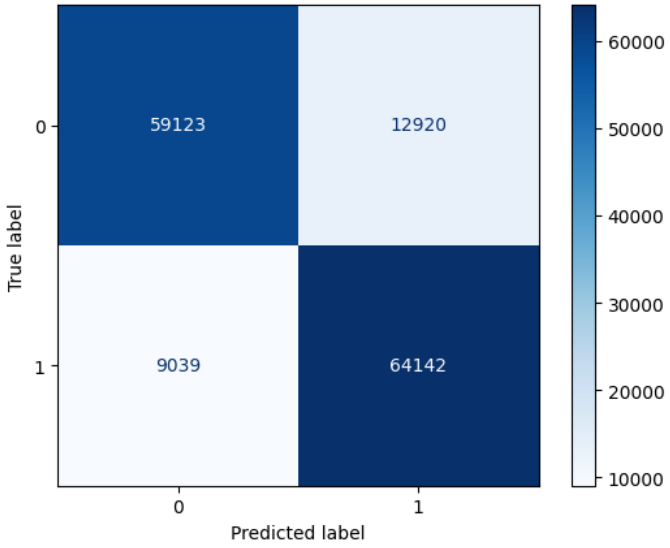


Figure 4.12 Confusion Matrix of Random Forest Classifier

	precision	recall	f1-score	support
0	0.83	0.49	0.62	72043
1	0.64	0.90	0.75	73181
accuracy			0.70	145224
macro avg	0.74	0.70	0.68	145224
weighted avg	0.73	0.70	0.68	145224

Figure 4.13 Classification Report of Gradient Boosting Classifier

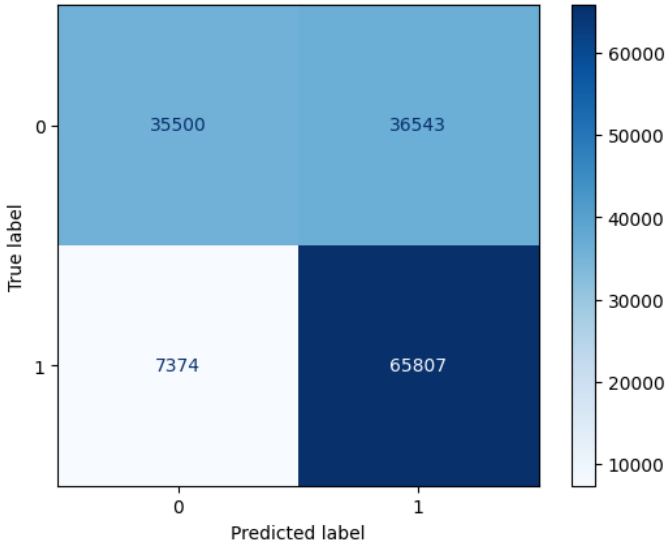


Figure 4.14 Confusion Matrix of Gradient Boosting Classifier

	precision	recall	f1-score	support
0	0.82	0.73	0.77	72043
1	0.76	0.85	0.80	73181
accuracy			0.79	145224
macro avg	0.79	0.79	0.79	145224
weighted avg	0.79	0.79	0.79	145224

Figure 4.15 Classification Report of XGBoost Classifier

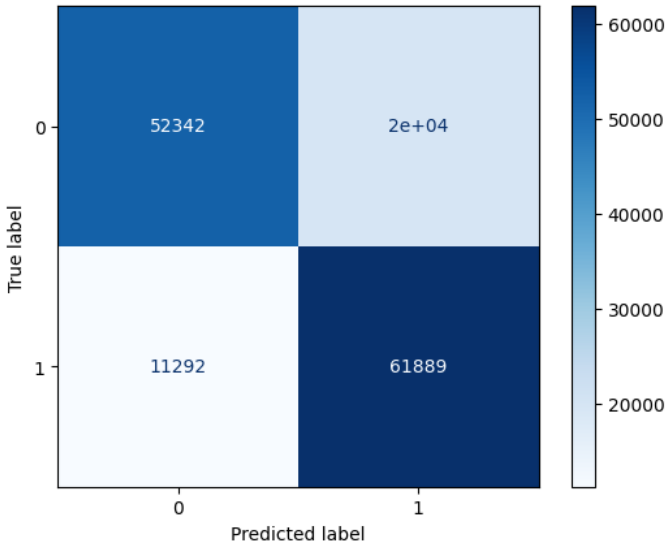


Figure 4.16 Confusion Matrix of XGBoost Classifier

Interface for testing the Classifiers:

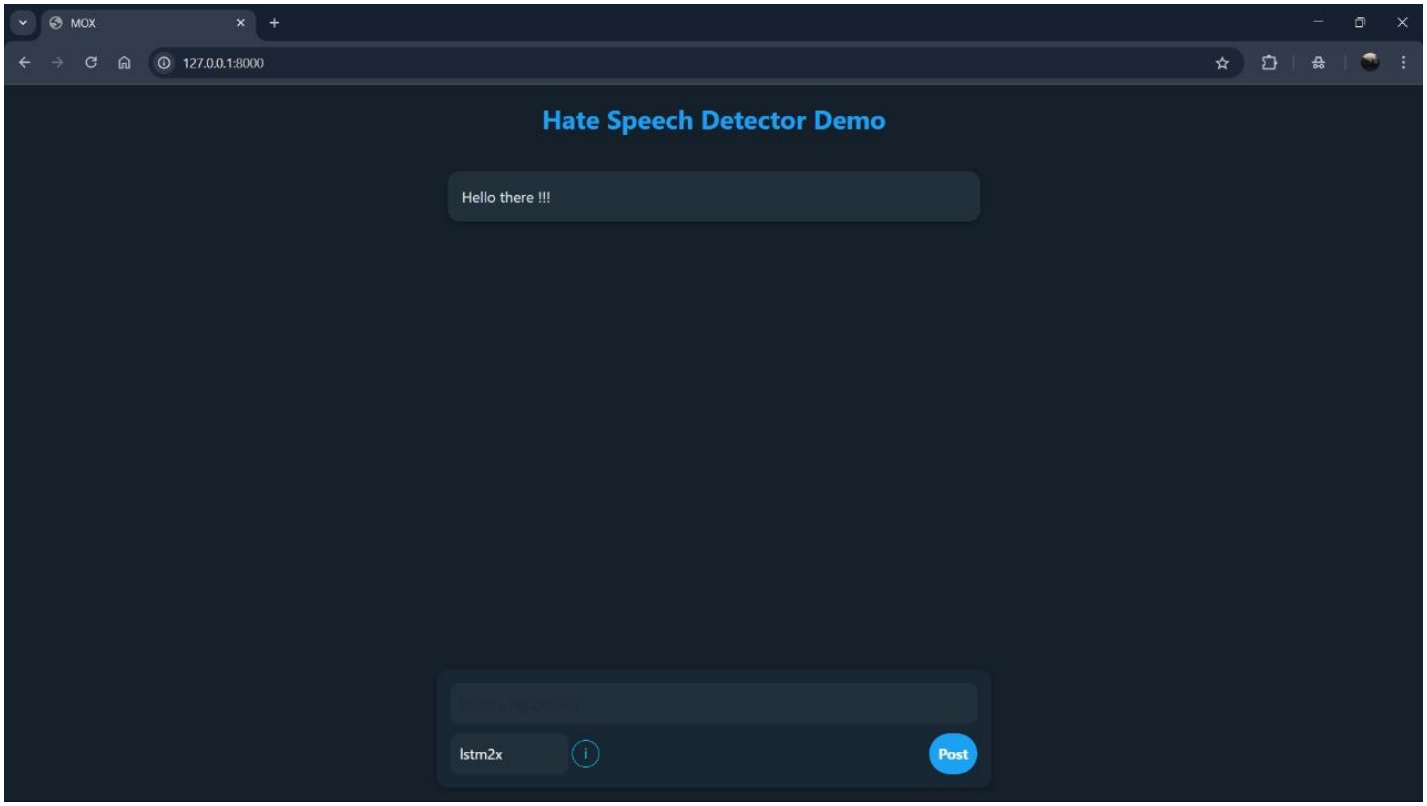


Figure 4.17 Interface Home Screen

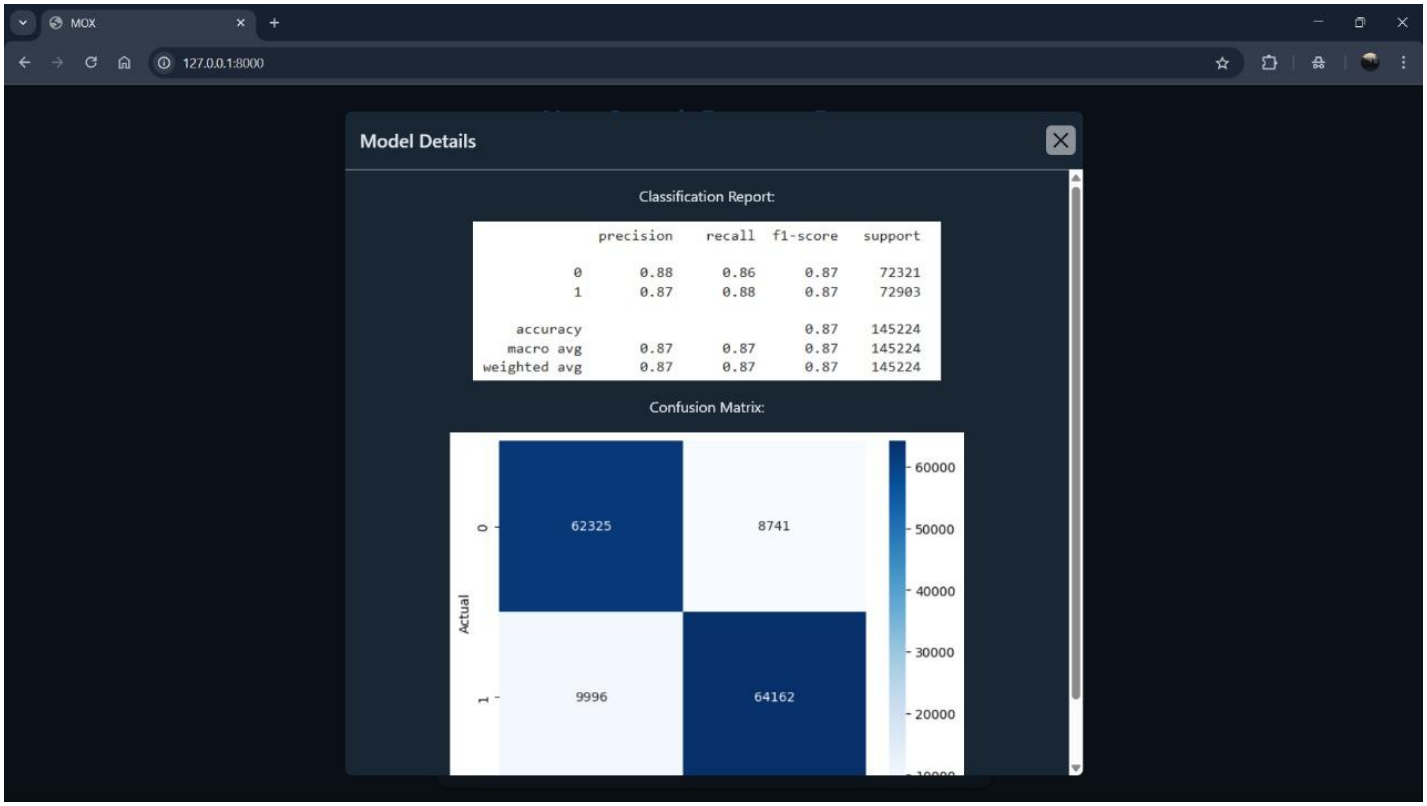


Figure 4.18 Model details in the interface

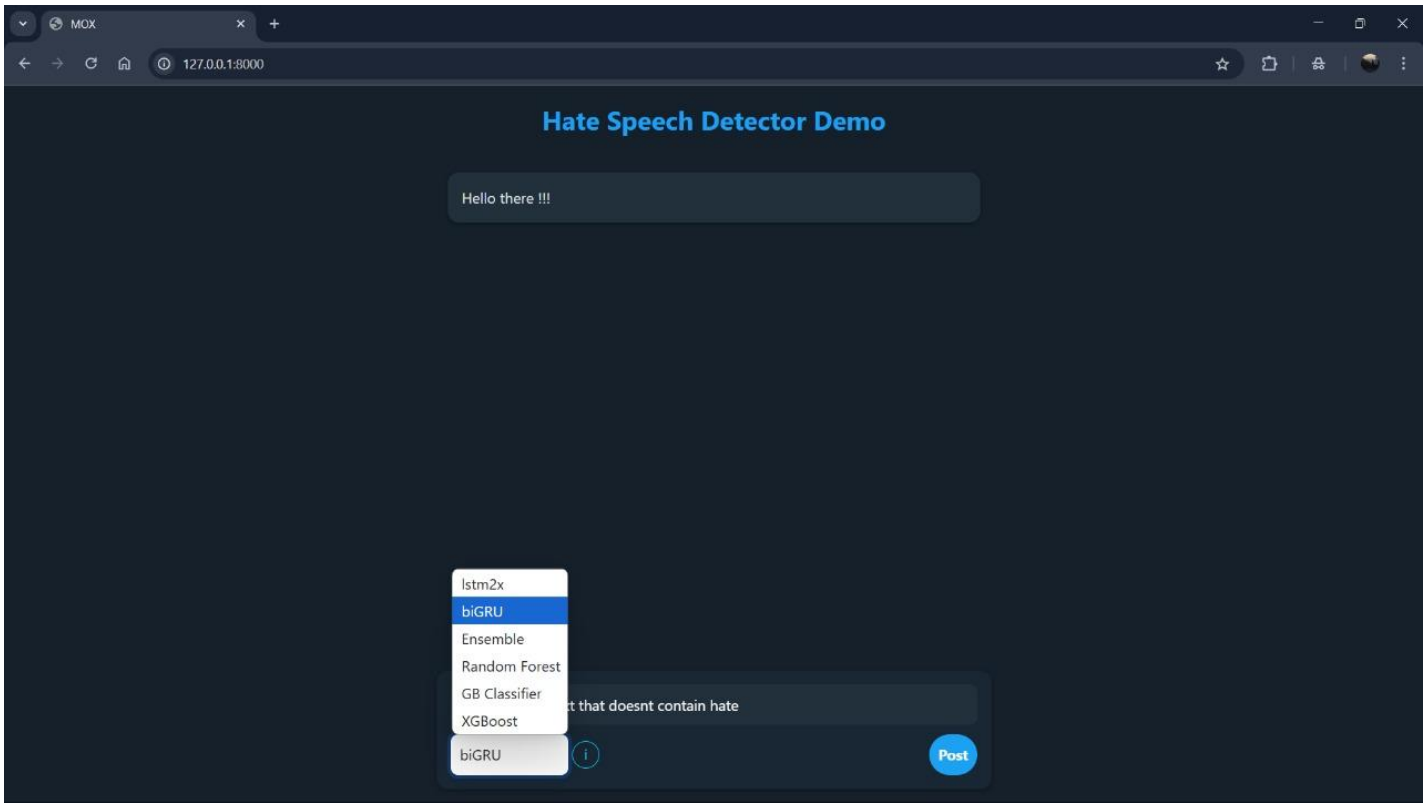


Figure 4.19 Drop down menu for classifier selection

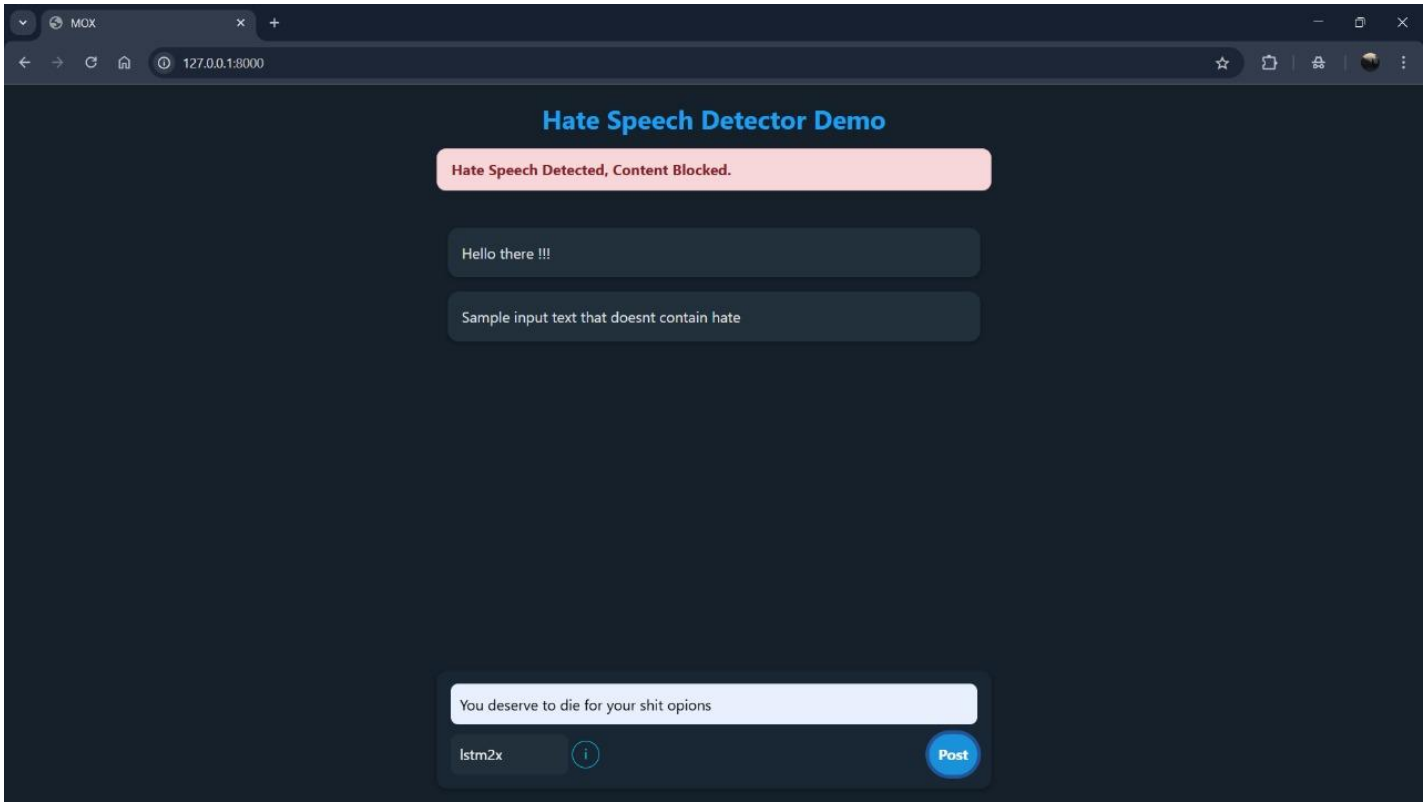


Figure 4.20 Hate Speech Detection on the interface

CHAPTER 5

CONCLUSION AND FUTURE PLANS

In an effort to reduce and bar hate speech on social media, Machine Learning algorithms have been developed with fairly decent accuracy and reliability. Although this doesn't encompass the ever-changing data sample space of hate speeches in the contemporary world, it definitely helps in reducing it to a large extent. These machine learning algorithms offer a leeway for a better understanding of social inclusivity and strengthening of the voices of the marginalized. However, there is a need to update researchers on new linguistics and vocabulary as a way to label the new influx of data samples. This just goes to show that one of the main limitations in this work is the unavailability of a dataset that is representative of all cultural and continental hate speeches. The dataset chosen barely scratches the surface of hate speeches, particularly the niche ones. Rampant spread of hate speeches cannot be contained but can be controlled, as most hate speeches also happen outside the influences of social media. With the little information there is about the most prevalent forms of bigotry, it is hardly true that these samples are representative of the linguistics used in a local household or even in a global sense. Hence, there is a necessity to survey, understand and interrogate people on the basis of humanities to really delve into this topic of interest.

For example, in India, the number '9' is used to insult gays, queers and transsexuals in a derogatory manner. This cannot be easily identified by the classifier as it requires a lot of context and history. Dissecting every word and labelling hate speeches have become one of the most open challenges in hate speech classification. It also doesn't help that sarcasm and comedy added in the mixture of data samples makes the stratification process tedious. Covert aggression and other devious sentiments behind humour takes time to uncover and might need manual labelling, which is again subjective and cannot be considered absolute. All future plans pertaining to hate speech text classification involves consideration of hate speech variables from multiple countries, a deep study of history of the marginalized and their lives, and popular anti-sentimental narratives relating to the oppressed to identify the fabrication of lies to silence the voices of the powerless, countless surveys to evaluate and create data samples to customize a balanced dataset. This report is particularly useful for any beginner entering into the domain of hate speech and machine learning classification. The source code and the interface lie in alignment to the objective of the base reference paper.

REFERENCES

[1] Hate speech vis-a-vis Freedom of speech in Indian democracy - IJFMR https://www.ijfmr.com/research-paper.php?id=14942

[2] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[3] Julian Risch, Anke Stoll, Lena Wilms, and Michael Wiegand. 2021. Overview of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments*, pages 1–12, Duesseldorf, Germany. Association for Computational Linguistics.

[4] Dataset Chosen for this research https://www.kaggle.com/datasets/waalbannyantudre/hate-speech-detection-curated-dataset?resource=download&select=HateSpeechDatasetBalanced.csv

[5] Ethos Binary Dataset https://www.kaggle.com/datasets/konradb/ethos-hate-speech-dataset

[6] Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

[7] M. Bouazizi and T. Otsuki Ohtsuki, "A Pattern-Based Approach for Sarcasm Detection on Twitter," in IEEE Access, vol. 4, pp. 5477-5488, 2016, doi: 10.1109/ACCESS.2016.2594194.

[8] Davidson, T., Warmsley, D., Macy, M. and Weber, I., 2017, May. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media* (Vol. 11, No. 1, pp. 512-515).

[9] Vidgen, Bertie, and Taha Yasseri. “Detecting Weak and Strong Islamophobic Hate Speech on Social Media.” Taylor & Francis, 2020. https://doi.org/10.1080/19331681.2019.1702607. [12] Burnap, P., Williams, M.L. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Sci.* **5**, 11 (2016).

[10] Burnap, P., Williams, M.L. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Sci.* **5**, 11 (2016).

[11] Mathew, B., Saha, P., Yimam, S. M., Biemann, C., Goyal, P., & Mukherjee, A. (2021). HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(17), 14867-14875.

[12] Anusha, Mudoor Devadas and Hosahalli Lakshmaiah Shashirekha. “An Ensemble Model for Hate Speech and Offensive Content Identification in Indo-European Languages.” *Fire* (2020).

APPENDIX

Base Reference Paper:

N. S. Mullah and W. M. N. W. Zainon, "Advances in Machine Learning Algorithms for Hate Speech Detection in Social Media: A Review," in IEEE Access, vol. 9, pp. 88364-88376, 2021, doi: 10.1109/ACCESS.2021.3089515.