Bilingual Aggression Detection using Machine Learning

Final Year Project
Report

Submitted by
Riddhi Gawande (A026)
Shruti Jain (A032)
Garima Merani (A042)

Under The Guidance Of
Prof. Ruchi Sharma

In fulfillment for the award of the degree of

B.TECH.

INFORMATION TECHNOLOGY

At

SVKM'S NMIMS Deemed to be UNIVERSITY | MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Department of Information Technology
Mukesh Patel School of Technology Management & Engineering
NMIMS (Deemed –to-be University)
JVPD Scheme Bhaktivedanta Swami Marg,
Ville Parle (W), Mumbai-400 056.

April, 2023

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to everyone who has supported us throughout our final year project.

Firstly, we would like to thank my supervisor Prof. Ruchi Sharma, for providing us with invaluable guidance, support, and feedback throughout the entire project. Your expertise and encouragement have been invaluable to the successful completion of this project.

We are extremely thankful to Dr. Ketan Shah, HOD, Department of IT for giving us this opportunity to work on this project and enhance our skills in this domain. We would also like to extend our sincere thanks to all the faculty and staff members of Department of IT, MPSTME for their support in completing this project on time.

Our thanks and appreciations also go to our friends and all the people have willingly helped out with their abilities.

Table of Contents

List of Figures

List of Tables

Abbreviations	
Abbreviation	Description
LR	Logistic Regression
RF	Random Forest
SVM	Support Vector Machine
LSTM	Long Short Term Memory
GRU	Gated Recurrent Units
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
BERT	Bidirectional Encoder Representations from Transformers
GBDT	Gradient-Boosted Decision Trees
RoBERTa	Robustly Optimized BERT-Pretraining Approach
TFIDF	Term Frequency Inverse Document Frequency
MIMCT	Multi-Input Multi-Channel Transfer
TIF-DNN	Transformer based Interpreter and Feature extraction model on Deep Neural Network
DNN	Deep Neural Networks
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
XLNet	eXtreme Language understanding NETwork
DistilBERT	Distilled Bidirectional Encoder Representations from Transformers
AWD-LSTM	Average Stochastic Gradient Descent Weight-Dropped Long Short-Term Memory

Abstract

The identification of hate speech in online social communities has recently surfaced as a contentious issue in the field of computational linguistics study. When it comes to natural language processing, one of the most difficult challenges continues to be comprehending linguistic phenomena, particularly in low-resource languages. Writing in social media platforms, particularly in multilingual societies like India, is characterized by frequent instances of code-mixing. Traditional deep learning algorithms that have been trained on monolingual data will not perform well on code-mixed data, and training new models can be difficult since there are not enough resources. One of the most essential solutions to this problem is to convert data from multiple languages into a single language. Within the scope of this work, a Translation-Based Preprocessing Framework is provided for the identification of hate speech. Within the structure that we suggested, we implemented translation using the Googletrans library in conjunction with the Hinglish to English mapping dictionary.

1 OVERVIEW

1.1 Project Specification

Introduction

The aim of our project is to detect hate or aggression in texts on online platforms such as tweets on twitter. We found that there are a few gaps in the current hate speech detection algorithms that are used in Instagram and Twitter. We aim to bridge those gaps through our research and implementation. One gap, for example, is that offensive words that contain a starred letter do not get detected as hate words. One other issue is that aggressive sentences that are written in a combined language of English and Hindi (commonly known as Hinglish) are not detected by the algorithms as hateful sentences. Issues like these are what we will try to overcome by developing algorithms.

Importance

Due to the massive use of the internet and social media these days, people can express themselves using social media platforms such as Twitter and Facebook, which are both user-friendly and free. People of all ages use these sites to share every detail of their lives, filling them with information. These sites have very few constraints on their users' ability to express themselves freely, anyone can post negative and unrealistic comments in abusive language against anyone with the intent of tarnishing one's image and status in society. Also, such posts and comments can have a very negative impact on one's mental health as well. This hate information would be in code-mixed form, which makes it difficult to detect automatically, especially in a country like India with a large multilingual and bilingual populace. Hence we decided to research and use machine learning methods to detect hate speech in Hindi-English code-mixed social media text.

Objective and Scope

Creating a model to help detect hate or aggressive speeches is the main objective.
We aim to take forward the existing research to come up with a better model. This would help in tackling many worldly issues, such as cyberbullying, etc., and help users that face such issues.

Software Requirements

Jupyter Notebook or Google Colab

Deliverables

At the end of this project, along with obtaining a model which would successfully detect aggression in hindi-english code-mixed text a paper describing our research and findings will be written.

1.2 Literature Survey

A previous study [1] discovered that classification algorithms such as gradient boosting, XGBoost, and SVM with majority voting provided higher levels of accuracy. This method, however, did not work well with code-mixed data and can be improved. Another study [2] proposed an ensemble method for detecting hate speech in Hinglish tweets that combined text and graph-based features, resulting in higher F1-scores than the baseline mode because they used fastText embeddings. The authors of the paper [3] presented a method for translating Hinglish tweets into English which used a machine translation model. Then classified the tweets using their own English hate speech detection model (TIF-DNN) whose outcomes were better than the baseline classifiers. In a research paper [4], a novel method was proposed by the authors to identify cyber abuse in various languages, including Hinglish, by utilizing transformer-based language models and convolutional neural networks. The study demonstrated that their approach outperformed previous state-of-the-art models on multiple datasets. Another study [5] introduced a language modeling technique based on neural networks for code-switched data. The model was capable of generating code-switched sentences from monolingual corpora and demonstrated improved perplexity and accuracy compared to the baseline model.

In paper [6], the authors presented a new cyberbullying detection model for the Hinglish language that incorporated emoticons, sentiment, and emotions. Their proposed model obtained an impressive F1-score of 0.73, outperforming both unimodal and single task models. In [7], the authors compared the efficiency of various pre-trained word embeddings, such as FastText, GloVe, and Word2vec, in recognising hateful speech in code-mixed Indian languages. Using the aforementioned models, bagging was used to create an ensemble. In this case, XLNet outperformed all the other models. The paper [8] introduced a new model that employs character-level embeddings to identify hate speech in a code-mixed Hindi-English language. Using character-level embeddings, the authors demonstrated that their approach's performance improved, particularly for detecting hate speech in code-mixed text.

In [9], researchers conducted a comprehensive survey of natural language processing techniques used to detect hate speech. The research highlights the difficulties and complexities associated with identifying hate speech, such as the lack of annotated data and the nature of hate speech as it evolves. In addition, the research investigates several unique strategies for identifying hate speech, including lexicon-based, machine learning-based, and deep learning-based approaches. It was concluded there should be a benchmark corpus to understand the accuracy of different models. In [10], the authors compared various deep learning-based models which are used to identify hate speech in Afaan Oromo. They evaluated the performance of numerous models like CNNs, LSTMs and Transformers. The research demonstrated that the transformer-based model performed exceptionally well, achieving a remarkable F1 score of 0.92. The authors concluded that training an embedded representation with a model and incorporating augmented samples increase accuracy.

Long Short-Term Memory (LSTM) neural networks with attention mechanisms were proposed in [11] for social media text aggression detection. The attention mechanism helped the model focus on relevant words and phrases and perform better. The authors used datasets from Wikipedia comments and social media platforms and achieved 87.2% and 78.1% accuracy, respectively. The proposed model performed well on both datasets, but the Wikipedia dataset outperformed the social media dataset. The data's nature and the need for more social media training data explain this. MANDOLA is a big-data processing and visualization platform for monitoring and detecting online hate speech [12]. Machine learning algorithms and natural language processing identify hate speech in massive social media datasets. The authors successfully tested the platform using Twitter datasets. MANDOLA's real-time visualization lets users track online hate speech. The authors also noted that the platform needs more accuracy and efficiency. The platform was tested only on Twitter data, so its performance on other social media platforms is unknown.

In [13], the authors delve into the difficulties that come with detecting hate speech on social media. They emphasize the challenge of deciphering the meaning and purpose behind certain words, phrases, or symbols. Moreover, they stress the issue of determining what falls under the umbrella of hate speech since this can vary depending on legal, social, and cultural factors. Additionally labeled data is helpful, but not all labeled data is useful. CNN-LSTM, RoBERTa, FastText were the methods which were implemented. In [14], the authors conduct a multilingual evaluation for detecting online hate speech. They gather data from various languages, including Arabic, English, Hindi, and Indonesian, and evaluate several machine learning models for detecting hate speech. They conclude that utilizing multilingual models can enhance the performance of hate speech detection. In [15], the authors propose a deep learning-based model for identifying hate speech in tweets. They combine convolutional and recurrent neural networks and achieve remarkable accuracy in recognizing hate speech in tweets. In [16], the authors focus on identifying hate speech in Roman Urdu, a language commonly used in Pakistan. They implemented supervised learning techniques like NB, LR, RF, SVM, and CNN. LR is the most effective technique for differentiating between content that is neutral and content that is hostile, and count vectors are the most effective features.

In [17], the authors suggest a novel approach for detecting cyber hate speech using threats-based othering language embeddings. They implemented vector embedding and Paragraph2Vec algorithms were used for clustering. They use these embeddings to recognize hostile language and othering language on social media platforms. Their approach performs better than the existing models in identifying cyber hate speech. The authors evaluate the effectiveness of deep learning models for identifying hostile language in Hindi text in [18]. They test various models such as convolutional neural networks, recurrent neural networks, and transformer models. The IndicBERT performs marginally better than the mBERT. Multi-CNN-based models with any word embedding variant have a modest advantage over the other basic models in identifying hostility. The authors of [19] propose a real-time analysis method named Kafka for identifying abusive behavior on social media. They utilize this machine learning model to pinpoint specific phrases and patterns that suggest coercive behavior. Their approach can be utilized to promptly identify abusive behavior on social media.

In [20], the authors introduce two methods: data streaming and parallelization. The first makes it possible for the framework to update the classification model every time a new tweet is labeled and to find aggressive tweets in real time. This is important for scaling the proposed method in a distributed cluster to handle the growing number of tweets without changing how well it classifies them. The models achieved 82-93% accuracy, precision, recall. In [21], the authors present a novel model for identifying offensive tweets written in a code-switched language consisting of Hindi and English. The tweets are sent to a pre-processing pipeline in order to convert them into semantic feature vectors. The authors introduced a technique called Ternary Trans-CNN model for classifying Hinglish hate speech.

In [22], the authors build an annotated Hinglish Offensive Tweet dataset which would decide whether transfer learning is effective for classifying offensive tweets in hinglish. Then a novel MIMCT model is built which outperformed the baseline models on the dataset. They use a combination of feature selection and classification algorithms to accurately identify offensive tweets. Future improvements could include using feature selection methods to choose the most important features, extending MIMCT to other code-switched and codemixed languages.

In [23], the authors delve into the detection of hate speech in Hindi-English and highlight the significance of author profiling and debiasing in the detection process. They propose a model that, after data preprocessing, uses baseline classifiers in order to perform Transliteration-based and fine-tuning Preprocessing. A profanity vector is augmented in order to improve performance. In [24], the authors present a hate speech detection model for multilingual Twitter utilizing convolutional neural networks. They collect data from multiple languages, such as English, Spanish, Portuguese, and Arabic. In the Baseline Method, different word representations like TFIDF, Bag of Words, and Char n-grams with traditional classifiers are used. In DNNs, three types of neural networks i.e., CNN, LSTM, FastText are used. In DNNs + GBDTs, the neural networks used in the second experiment with the classifier GBDTs are combined. DNNs are used to extract features for GBDTs.They demonstrated that their approach can achieve high accuracy in identifying hate speech. In [25], the authors assess the progress of machine learning algorithms for detecting hate speech in social media. They discuss the obstacles associated with hate speech detection, such as bias and context ambiguity, and review the current state-of-the-art models for detecting hate speech.

While there is a vast amount of literature on the topic of hate speech detection, there is notably little on the topic of code-mixed datasets. Code-mixing refers to the practice of switching effortlessly between two or more languages while writing. Code-mixed text, such as that found in Hinglish tweets, makes it challenging to create reliable hate speech detection models. There is a significant gap in the existing research because there is currently no publicly available code-mixed dataset for detecting hate speech in Hinglish tweets.

2. ANALYSIS & DESIGN

2.1 Requirement Analysis

Developing a bilingual aggression detection model using machine learning requires additional considerations compared to a unilingual system. Here are some requirements for developing a bilingual aggression detection model using machine learning:

1. Data Collection: Collecting a large and diverse dataset of text data that includes examples of aggressive behavior in both languages is critical. The dataset should include examples of aggression in the languages to be detected, along with accurate translations for training the model.
2. Data Annotation: Annotating the dataset requires bilingual annotators who are fluent in the code-mixed languages. The annotators must identify the aggressive behaviors expressed in code-mixed languages and provide appropriate labels for the same.
3. Feature Extraction: Feature extraction is complex in code-mixed languages and requires careful analysis of language mixing patterns. Linguistic features such as tone, prosody, syntax, and language switching must be extracted to identify aggressive language in code-mixed languages.
4. Machine Learning Model Selection: The choice of machine learning model may differ depending on the complexity of the code-mixed languages. The model should be capable of processing multilingual inputs and handling the linguistic differences between the languages. The model should be capable of handling the mixing patterns of the languages and identifying the aggressive behavior expressed in the mixed language.
5. Training and Validation: The dataset should be split into training, validation, and test sets, with the model being trained on the training set and evaluated on the validation set. The performance of the model should be monitored separately for each code-mixed language, and the model should be trained to handle the mixing patterns of the languages and identify the aggressive behavior expressed in the mixed language.
6. Evaluation: The system should be evaluated for code-mixed language, and its performance should be measured using metrics such as accuracy, precision, recall, and F1 score depending on the data used.

In summary, developing a code-mixed language aggression detection system requires a diverse dataset, bilingual annotation, complex feature extraction, and a machine learning model that can handle the mixing patterns of the code-mixed languages.

2.2 Feasibility Study

A feasibility study of a code-mixed language aggression detection project using machine learning would involve analyzing various aspects of the project to determine if it is practical and achievable. Here are some key considerations for the feasibility study:

1. Data Availability: The availability of a suitable dataset is critical for the success of the project. A diverse and comprehensive dataset of code-mixed language aggression behavior is needed for training and validating the machine learning model. The availability of such a dataset should be analyzed to determine if it is feasible to collect or obtain one.

   The Hindi-English code mixed open-source dataset available online is quite small and hence, a dataset with scrapped tweets from twitter will have to be created for this project as well as labeled.

2. Resource Availability: The development of a code-mixed language aggression detection system using machine learning requires computing resources such as servers, storage, and computational power. The availability of such resources should be analyzed to determine if they are feasible to obtain or allocate.

   Google’s collab environment, or kaggle’s Notebooks will be used to create this project with the available GPU being sufficient for this project’s training.

3. Model Complexity: Developing a machine learning model capable of detecting aggression expressed in code-mixed languages is a complex task. The feasibility of developing a suitable model should be analyzed to determine if it is achievable within the project timeline and resource constraints.

   The project’s timeline and resources are sufficient enough to develop a decent model with decent results for this project.

4. Model Performance: The performance of the machine learning model should be analyzed to determine if it is feasible to achieve acceptable levels of accuracy, precision, recall, and F1 score for the detection of code-mixed language aggression.
5. Deployment: The feasibility of deploying the code-mixed language aggression detection system in a real-world scenario should be analyzed. The system should be able to handle inputs in code-mixed languages and integrate with existing applications or systems.

   The model created for this project is a part of the ongoing research being conducted for detecting hate speech and aggression in hindi-english code mixed language. The results received by us can be useful for further research and studies in this domain, and be useful in real world applications for forum based apps like twitter and facebook in improving their aggression detecting algorithms.

Considering all the above mentioned categories, we conclude that the project is feasible and can be conducted in the given timeline with the available resources.

2.3 Design Development

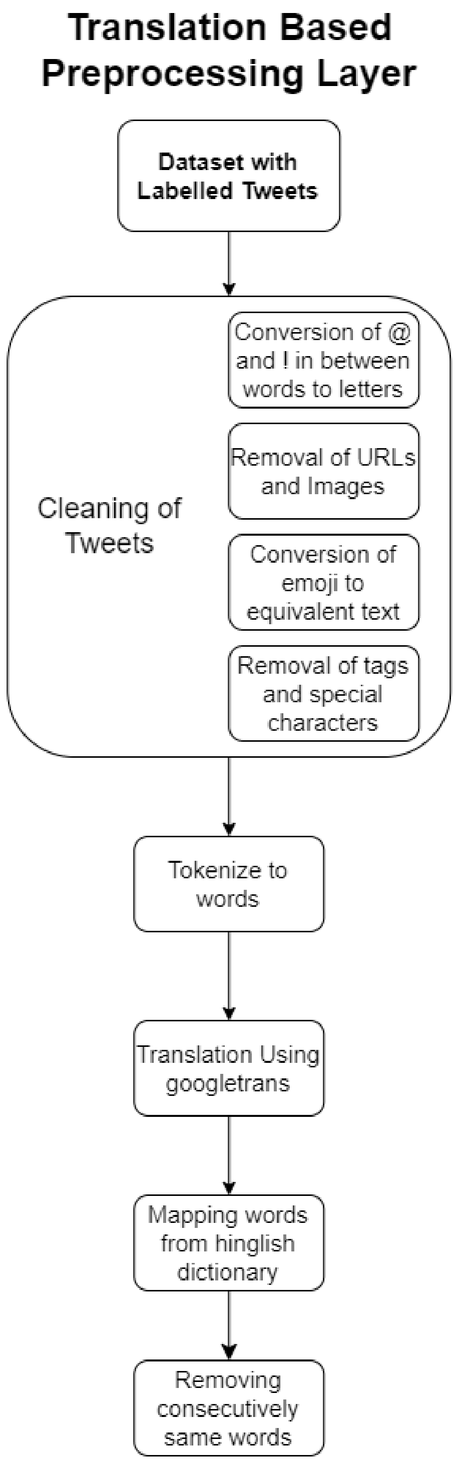

Translation Based Preprocessing Layer
Dataset with Labelled Tweets
Cleaning of Tweets
Conversion of @ and ! in between words to letters
Removal of URLs and Images
Conversion of emoji to equivalent text
Removal of tags and special characters
Tokenize to words
Translation Using googletrans
Mapping words from hinglish dictionary
Removing consecutively same words

Figure 1: Custom Framework

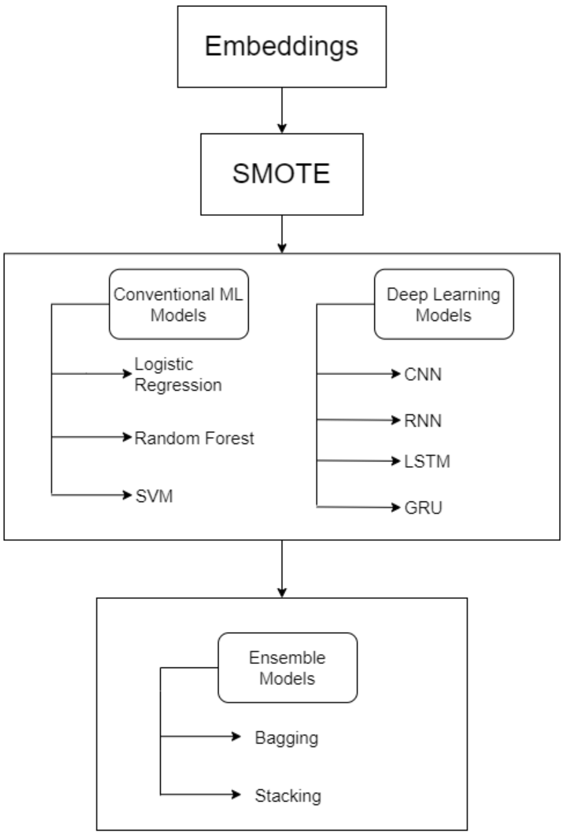


Figure 2: Model Building

1. **Data Collection**:

   Collect a dataset of Hindi-English code mixed text that contains aggressive or non-aggressive language. The dataset should be large enough to train a machine learning model.

2. **Labeling Data:**

   Labeling data into 2 classes: hate and non-hate

3. **Data preprocessing**

   I. Data Cleaning: Converting ! to i and @ to a if they are in between a word. Removal of # and other special characters as well as URLs. Converting all the emojis to their equivalent text meanings.

   II. Tokenization: This involves splitting the text at each whitespace character and separating punctuation from words.

   III. Translation using google trans library: Translating each token to its english meaning using the google trans library available in python.

IV. Mapping: Mapping each token to its english meaning using the hinglish to english mapping dictionary

V. Removing consecutive words: Removing duplicate words that occur consecutively in a sentence

4. **Model Building:**

   After splitting the data into training and testing sets, the training data is used to train various models. We will be using the traditional machine learning models such as logistic regression, random forest, SVM, for our task. Along with these traditional models, deep learning networks such as CNN, RNN, LSTM, GRU along with a few hybrid models would also be used for this NLP task. At last we will also be trying different ensemble methods to get the best outcome for the classification task.

5. **Embeddings:**

   All the above mentioned will be tried with various embeddings such as TF-IDF, glove.

6. **Tuning**:

   Top 3 models giving the best results will undergo hyperparameter tuning for gaining better results.

2.4 Technology and Software Details

To develop a Hindi-English code-mixed aggression detection model using machine learning, there are several technologies and software tools that can be used. Here are some of the details:

1. **Programming Language:**

   Python is a popular programming language for machine learning and natural language processing tasks and hence, this project will be made using python.

2. **Natural Language Processing Libraries**:

There are several NLP libraries available in Python that can be used for this project, such as NLTK, spaCy, and TextBlob. These libraries provide tools for tokenization, part-of-speech tagging, sentiment analysis, and other NLP tasks.

3. **Machine Learning Libraries**:

   Python also has several machine learning libraries, including scikit-learn and TensorFlow, that will be used to train and test machine learning models for aggression detection.

4. **Platform**:

   Google Colab, also known as Google Colaboratory, is a cloud-based platform provided by Google for machine learning and data analysis tasks. It allows users to write, run, and share Python code in a browser-based environment that provides access to powerful computing resources, such as GPUs and TPUs. This project will be coded on this platform to make it easily accessible for all group members to edit, make changes and collaborate in an easy way.

2.5 Project Planning

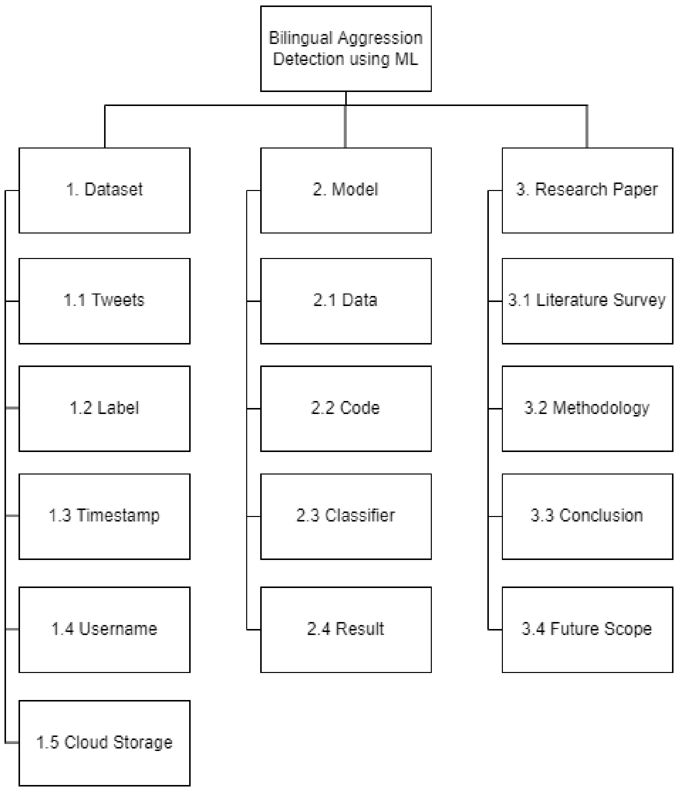


Figure 3: Product Breakdown Structure

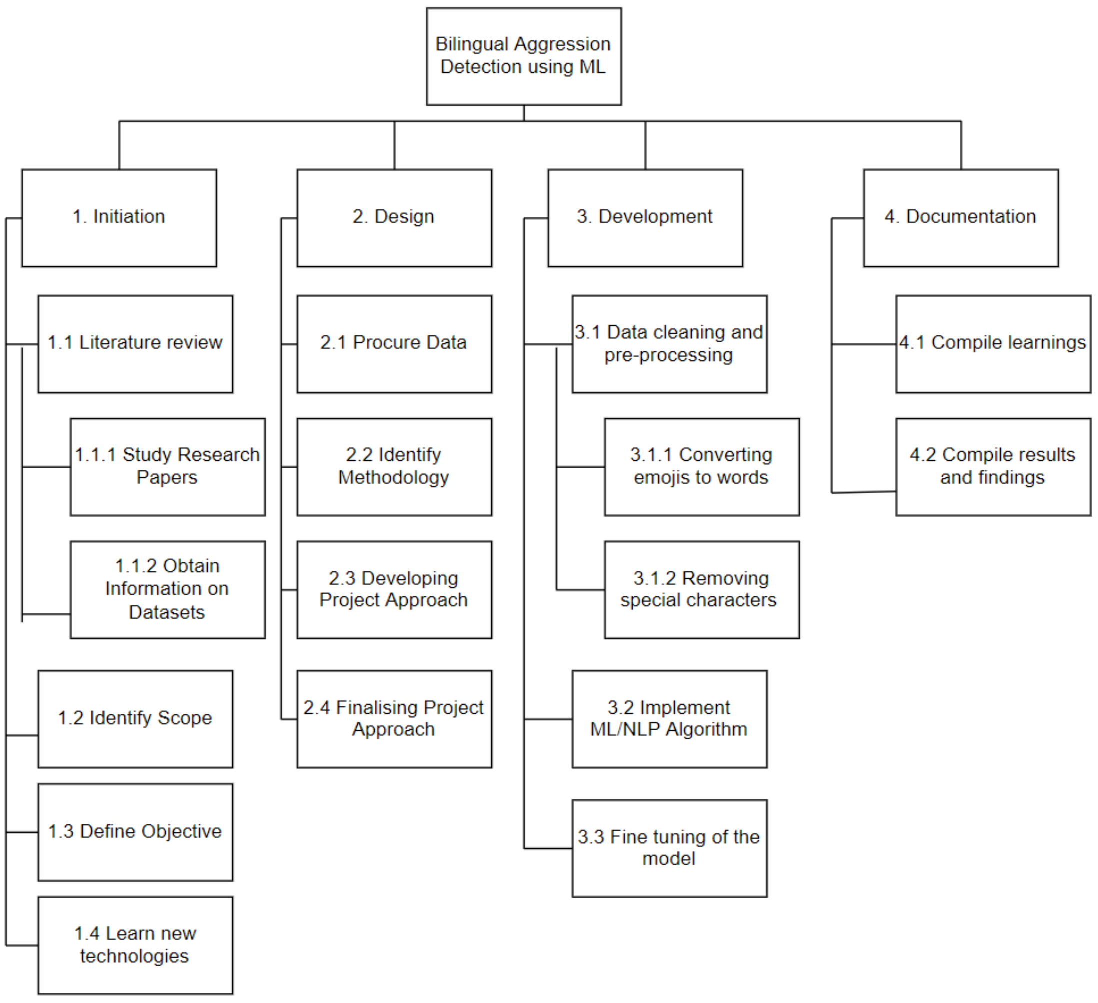

Bilingual Aggression Detection using ML
1. Initiation
1.1 Literature review
1.1.1 Study Research Papers
1.1.2 Obtain Information on Datasets
1.2 Identify Scope
1.3 Define Objective
1.4 Learn new technologies
2. Design
2.1 Procure Data
2.2 Identify Methodology
2.3 Developing Project Approach
2.4 Finalising Project Approach
3. Development
3.1 Data cleaning and pre-processing
3.1.1 Converting emojis to words
3.1.2 Removing special characters
3.2 Implement ML/NLP Algorithm
3.3 Fine tuning of the model
4. Documentation
4.1 Compile learnings
4.2 Compile results and findings

Figure 4: Work Breakdown Structure

SN	Task	Duration in hrs – per week	Start Date	End Date	Predecessors
1	Initial Planning	4	1/7/2022	14/7/2022	-
2	Identifying Scope and Deliverables	4	14/7/2022	31/7/2022	1
3	Literature Survey	8	21/7/2022	21/8/2022	1
4	Scraping Data	4	21/8/2022	31/8/2022	2, 3
5	Data Labeling	12	1/9/2022	14/10/2022	4
6	Data Pre Processing	6	14/10/2022	30/11/2022	5
7	Basic Model Building	6	1/12/2022	7/1/2023	6
8	Model Tuning	8	7/1/2022	14/2/2023	7
9	Approval and Final Eval	6	14/2/2023	7/3/2023	8
10	Technical Paper Writing	12	14/2/2023	14/3/2023	3
11	Final Editing and Submission	4	7/3/2023	20/3/2023	10

Table I. project timeline

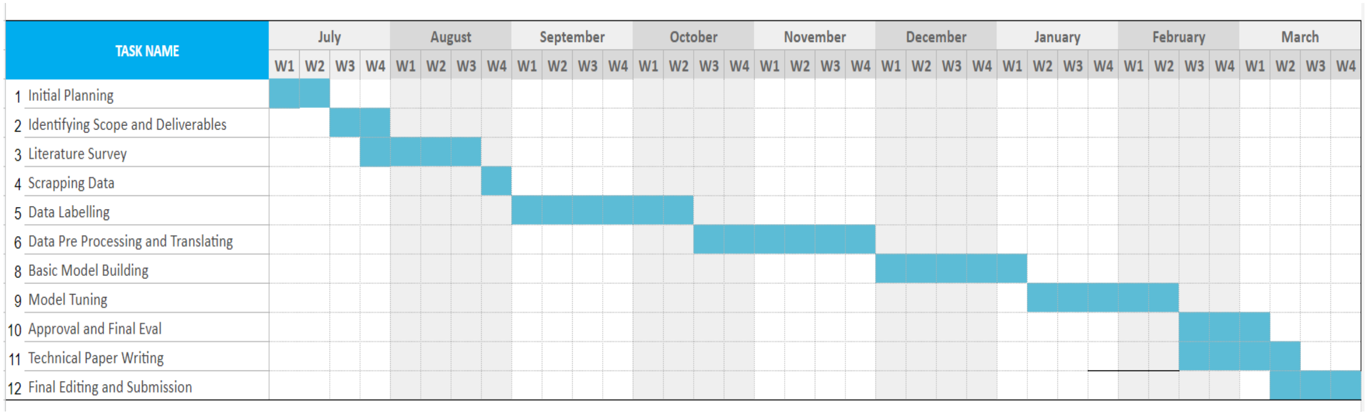


Figure 5: Gantt Chart

3 Project Description

3.1 Problem Statement

Hate speech and aggression in online texts, such as social media posts and comments, pose a serious threat to social harmony and can have harmful effects on individuals and communities. With the rise of social media users, a rapid increase in cyberbullying cases has also increased. Since, current algorithms used by social media platforms to detect hate speech have limitations, such as the inability to detect offensive words with starred letters, or special characters, or to identify Hinglish sentences as hateful. Twitter faces the problem of that as even when users report certain tweets for hate speech, often those users don't face any repercussions as twitter is unable to understand hinglish hence these tweets are not flagged as inappropriate even though they contain racial and/or offensive slurs. Hence social media platforms are limited in their ability to accurately detect and mitigate hate speech and slurs in code-mixed languages like Hinglish

3.2 Objective and Goals

Our project aims to improve the accuracy of hate speech detection on Twitter. We're focusing on Hinglish tweets, where current algorithms struggle with offensive words containing special characters like '@' and '!'. Additionally, aggressive sentences written in Hinglish aren't recognized as hateful. Our goal is to develop a framework that can accurately detect hate speech and aggression in online texts. By doing so, we hope to contribute to effectively moderating and mitigating this issue.

3.3 Scope

Our project's main objective is to develop a reliable model for detecting hate speech and aggression in online texts, specifically Hinglish tweets on Twitter. The focus of this project is on the gap which algorithms of social media platforms like Twitter face when it comes to detecting hate speech.

In order to achieve this, we will use a large dataset which will contain Hinglish tweets and conduct analysis accordingly. We'll preprocess, clean, and label the data using natural language processing techniques. Our primary focus is to address the gaps in the current frameworks, such as the inability to detect offensive words containing special characters and failure to recognize aggressive sentences written in Hinglish as hateful. We'll use machine learning and deep learning algorithms to develop a detection model that can accurately detect instances of hate speech and aggression in Hinglish tweets.

Our project aims to contribute to mitigating the negative impact of hate speech and aggression on social media platforms by developing an effective hate speech detection framework.

4 PROJECT IMPLEMENTATION

4.1 Dataset Preparation

We made a custom dataset of Hinglish tweets found on Twitter. For this, we used Python Libraries “snscrape” and “twint”. We used hinglish keywords (offensive as well as non-offensive) to extract these tweets. If the tweet was offensive or hate inducing, it was labeled 1, whereas if the tweet was neutral, it was labeled 0.

After extraction and adding of tweets, the size of the total dataset was 10.5k consisting of 6405 tweets of class 1 and 4097 of class 0. This dataset is the one on which we would implement the project.

4.2 Preprocessing of Data

To clean and preprocess our data, we created our own framework which is a Translation Based Preprocessing Layer.

As mentioned above in the architecture, the first step was to clean the data. Cleaning the data included the following:

- Conversion of ‘@’ to ‘a’ and ‘!’ to ‘i’: It was observed that many people used special characters such as @ and ! to mask abusive words. This would lead to ignorance of the word and more usage of such words. To overcome this issue, we replaced these special characters with its respective letter if it was present in between a word. A custom function was created for this purpose.
- Removal of URLs and Images: All links were removed from tweets using the Python Library “re”.
- Conversion of emojis to equivalent text: Since emojis have sentimental value and meaning, removing them entirely would not have been beneficial. For this reason, we used the Python Library “emoji” to convert the emojis to their respective text.
- Removal of tags and special characters: Usernames and hashtags were removed in this step. Other special characters and punctuations were removed as well. This was done using the Python Library “re”.

Once the tweets were cleaned, we tokenized them. This was done using the split function. This resulted in a list of words for each tweet.

Further, we converted the tweets from hinglish to english. This was done using 2 methods which were done one after the other to get a more accurate translation.

- Google Translate: The Python Library “googletrans” was used. Each word in the tokenized list was checked and converted to english.
- Mapping: We found a dictionary on GitHub containing the mapping of Hinglish words to their respective English words. We used this dictionary to translate words that could not be translated by “googletrans”. The dictionary contained abusive as well as neutral words.

Once the translated list was ready, we removed words that were consecutively repeated, keeping the frequency once. For example, “good good good” got replaced with “good”.

4.3 Balancing

Since our dataset was not balanced, we implemented a data balancing method before implementing the models. Method used for this was SMOTE .

SMOTE: Synthetic Minority Over-sampling Technique is a type of data augmentation technique which balances the class distribution, if any, by generating synthetic samples of the class which is considered as minority, in order to prevent bias towards the majority class. This helps in the performance of the models as it provides better training data in terms of diversity.

4.4 Model Implementations

Once the dataset was balanced, we implemented multiple models on the same.

Machine Learning Models:

- **Logistic Regression**:

  Based on a set of input features, a logistic regression model predicts the likelihood of a binary outcome (in our case, aggressive or non-aggressive). It operates by fitting a logistic function to a linear combination of input features. To compare the performance of more complex models, logistic regression was utilized as a baseline model.

- **Random Forest**:

  Random forest is an ensemble learning method for improving classification accuracy by combining multiple decision trees. Each tree is trained using a subset of the training data and features chosen at random. The majority vote of the individual tree predictions is used to make the final prediction.

- **SVM:**

  SVM is an effective classification method for determining the best boundary between two classes of data points. It works by mapping the input features to a higher-dimensional space and locating the hyperplane that divides the two classes the most. SVM can be used with various kernel types to handle non-linearly separable data.

Deep Learning Models:

- **CNN**:

  These neural networks excel at image classification but can also be used for text classification. In the case of hate speech detection, a CNN could examine the structure and meaning of words and phrases in tweets to determine whether or not they are aggressive.

- **RNN:**

  Recurrent neural networks (RNNs) are specialized networks capable of processing sequential data, including text. By analyzing each word in a sequence and taking into account the preceding words, RNNs develop a contextual comprehension of the text. This property makes RNNs particularly useful for a task like hate speech detection, where the meaning of a message is heavily influenced by the words that come before and after it as context is capable of changing the entire meaning of the message which was meant to be conveyed.

- **LSTM:**

  LSTMs are a type of RNN that model long-term dependencies in sequential data particularly well. This makes them well-suited for tasks such as detecting hate speech on Twitter, where the context can span multiple words and phrases.

- **GRU:**

  Gated Recurrent Units (GRUs) are a type of neural network that processes sequential data similarly to Long Short-Term Memory (LSTM) networks.GRU would be useful for hate speech classification as it can effectively capture long-term dependencies in the input sequence and selectively retain or discard information. Its gating mechanisms enable it to recognize patterns in the input sequence that are indicative of hate speech and use those patterns to make classification decisions.

Based on these models, Ensemble models were created to improve accuracy. For the models, ".joblib" files were downloaded, using which ensemble models were created. The following were created using the pretrained models:

- **Bagging**:

  The implementation of the bagging ensemble technique helped enhance the hate speech detection models by reducing their variance. The technique functioned by producing multiple subsets of the data and training distinct models on each subset. By aggregating the forecasts of each model, this technique achieved superior outcomes in terms of precision and consistency. The base estimator used by us is SVM.

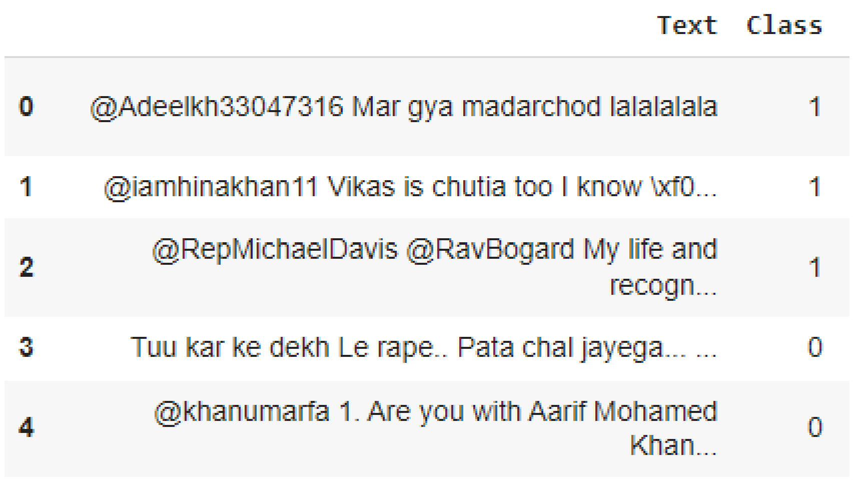
- **Stacking**:

  A stacking ensemble technique was used to enhance the precision of the hate speech detection models. This technique involved amalgamating multiple machine learning models with a meta-classifier, which helped to reduce the misclassification rate drastically. Stacking also helped in combating the problem of overfitting and improving the resilience of the models. The meta estimator used is SVM, while RF and LR are used as other estimators.

5. SCREENSHOTS

5.1 Dataset

The screenshot shows 5 rows of the dataset used. It is a self scraped and labeled dataset, consisting of a total of 10.5k rows. The class of each tweet is seen; 0 symbolizing non-hate and 1 symbolizing hate speech.

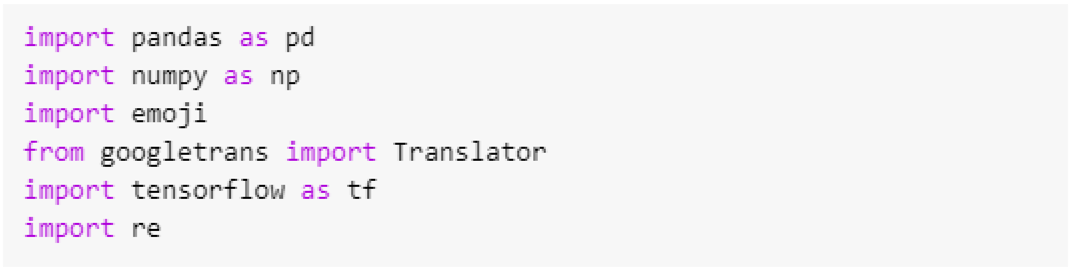


	Text	Class
0	@Adeelkh33047316 Mar gya madarchod lalalalala	1
1	@iamhinakhan11 Vikas is chutia too I know \xf0...	1
2	@RepMichaelDavis @RavBogard My life and recogn...	1
3	Tuu kar ke dekh Le rape.. Pata chal jayega... ...	0
4	@khanumarfa 1. Are you with Aarif Mohamed Khan...	0

Figure 6: Dataset

5.2 Libraries Used

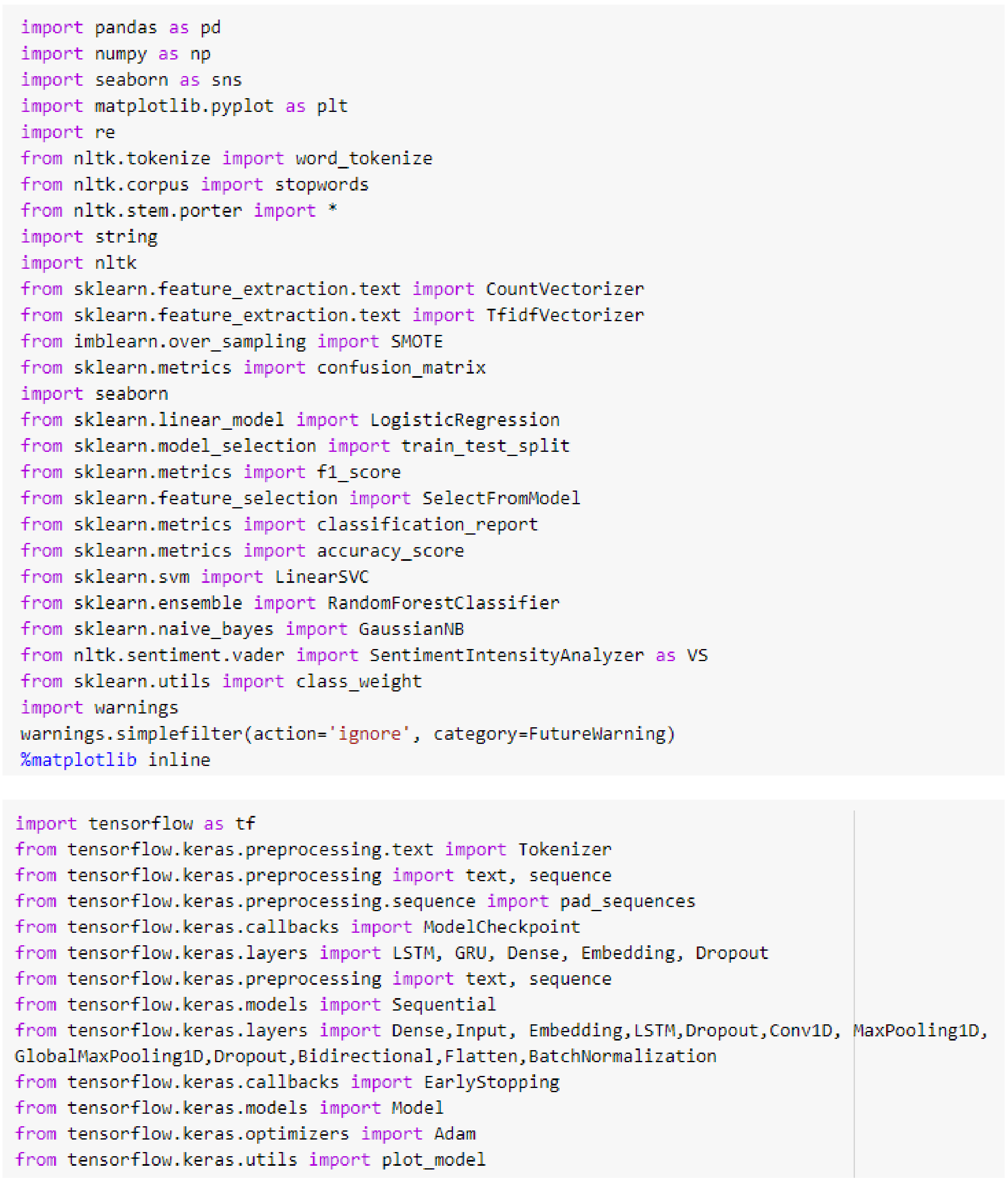
Multiple libraries were used in the notebooks, as seen below.



```
import pandas as pd
import numpy as np
import emoji
from googletrans import Translator
import tensorflow as tf
import re
```

Figure 7: Libraries in Preprocessing Notebook

Libraries used in the model building notebooks are also mentioned below.



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import *
import string
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix
import seaborn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from nltk.sentiment.vader import SentimentIntensityAnalyzer as VS
from sklearn.utils import class_weight
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import text, sequence
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.layers import LSTM, GRU, Dense, Embedding, Dropout
from tensorflow.keras.preprocessing import text, sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Input, Embedding,LSTM,Dropout,Conv1D, MaxPooling1D,
GlobalMaxPooling1D,Dropout,Bidirectional,Flatten,BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
```

Figure 8: Libraries in Models Notebook

Pandas:

Pandas is a popular open-source Python library for data manipulation and analysis. It provides data structures and functions for handling and analyzing structured data. It is commonly used for tasks such as data cleaning, merging, filtering, and visualization.

Numpy:

NumPy is a Python library used for scientific computing and numerical analysis. It provides data structures for multi-dimensional arrays, as well as mathematical functions to operate on these arrays. NumPy is widely used for tasks such as linear algebra, statistical analysis, and image processing.

Emoji:

The Emoji library is a Python package for working with emojis. It provides a range of functions for manipulating and working with emojis, including converting text to emojis, extracting emojis from text, and replacing emojis with text. It can be useful for tasks such as sentiment analysis or text classification where emojis may carry additional meaning.

Re:

The re (regular expression) library in Python provides a set of tools for working with regular expressions. Regular expressions are patterns used to match and manipulate text. The library can be used to search for specific patterns, replace patterns with new text, and split text based on patterns.

Googletrans:

Googletrans is a Python library that provides easy access to Google Translate API. It allows you to translate text from one language to another, detect the language of the source text, and obtain the pronunciation of the translated text. The library supports multiple languages and is free to use.

Tensorflow:

TensorFlow is an open-source machine learning library developed by Google. It provides a platform for building and training neural networks to solve various machine learning problems, such as image recognition, natural language processing, and speech recognition. TensorFlow offers a flexible and scalable architecture that allows developers to build and deploy machine learning models across different devices and platforms.

NLTK:

NLTK (Natural Language Toolkit) is a Python library used for working with human language data. It provides easy-to-use interfaces for tasks such as tokenization, stemming, and tagging parts of speech. NLTK also includes corpora and resources for training and testing natural language processing models.

Seaborn:

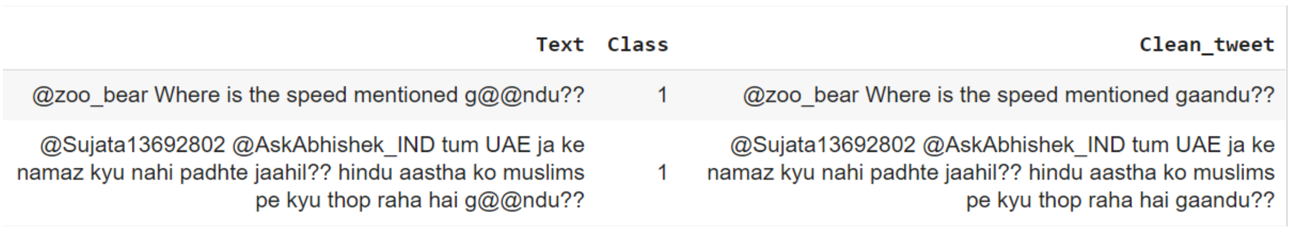
Seaborn is a data visualization library in Python that is built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. With Seaborn, you can easily create plots such as histograms, bar plots, heatmaps, and scatterplots with minimal code.

Sklearn:

Scikit-learn, also known as sklearn, is a popular machine learning library in Python. It provides tools for data preprocessing, feature selection, model selection, and evaluation. The library includes a wide range of algorithms for classification, regression, clustering, and dimensionality reduction.

5.3 Preprocessing Framework

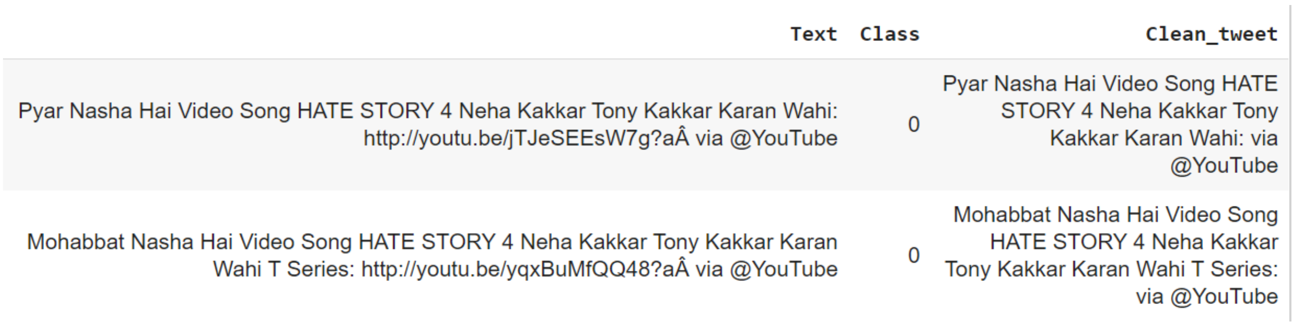
The framework used in our preprocessing model is a customized framework. The first step of the same, was to replace special characters in the middle of words with their respective alphabet. For example: “f@m!ly” gets converted to “family”.



Text	Class	Clean_tweet
@zoo_bear Where is the speed mentioned g@@ndu??	1	@zoo_bear Where is the speed mentioned gaandu??
@Sujata13692802 @AskAbhishek_IND tum UAE ja ke namaz kyu nahi padhte jaahil?? hindu aastha ko muslims pe kyu thop raha hai g@@ndu??	1	@Sujata13692802 @AskAbhishek_IND tum UAE ja ke namaz kyu nahi padhte jaahil?? hindu aastha ko muslims pe kyu thop raha hai gaandu??

Figure 9: Output of function to replace special characters with respective letter

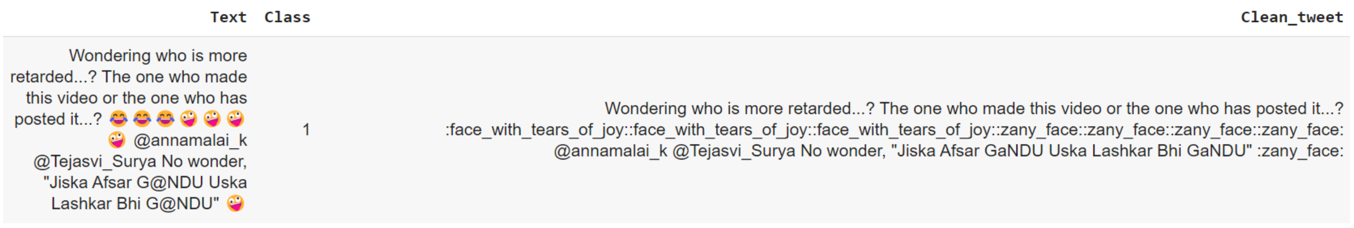
The next step was to remove URLs, images and tags. The output achieved for the same is visible in the screenshot below.



Text	Class	Clean_tweet
Pyar Nasha Hai Video Song HATE STORY 4 Neha Kakkar Tony Kakkar Karan Wahi: http://youtu.be/jTJeSEEsW7g?aÂ via @YouTube	0	Pyar Nasha Hai Video Song HATE STORY 4 Neha Kakkar Tony Kakkar Karan Wahi: via @YouTube
Mohabbat Nasha Hai Video Song HATE STORY 4 Neha Kakkar Tony Kakkar Karan Wahi T Series: http://youtu.be/yqxBuMfQQ48?aÂ via @YouTube	0	Mohabbat Nasha Hai Video Song HATE STORY 4 Neha Kakkar Tony Kakkar Karan Wahi T Series: via @YouTube

Figure 10: Output of removing URLs and Images

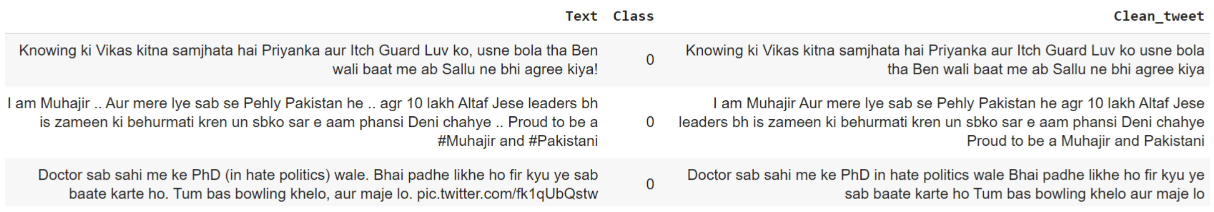
We realized that emojis hold an importance in the sentiment of the tweet, which is why we decided to use the library “emoji” and replace all emojis with its respective word. The output can be visible below.



Text	Class	Clean_tweet
Wondering who is more retarded...? The one who made this video or the one who has posted it...? 😂😂😂🤪🤪🤪🤪 @annamalai_k @Tejasvi_Surya No wonder, "Jiska Afsar G@NDU Uska Lashkar Bhi G@NDU" 🤪	1	Wondering who is more retarded...? The one who made this video or the one who has posted it...? :face_with_tears_of_joy::face_with_tears_of_joy::face_with_tears_of_joy::zany_face::zany_face::zany_face::zany_face: @annamalai_k @Tejasvi_Surya No wonder, "Jiska Afsar GaNDU Uska Lashkar Bhi GaNDU" :zany_face:

Figure 11: Output after converting emojis to equivalent text

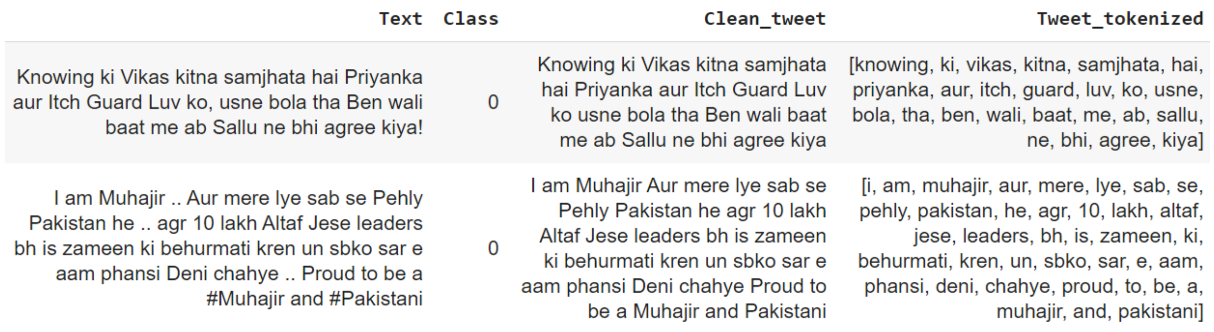
Next was to remove other special characters that were not replaced earlier.



Text	Class	Clean_tweet
Knowing ki Vikas kitna samjhata hai Priyanka aur Itch Guard Luv ko, usne bola tha Ben wali baat me ab Sallu ne bhi agree kiya!	0	Knowing ki Vikas kitna samjhata hai Priyanka aur Itch Guard Luv ko usne bola tha Ben wali baat me ab Sallu ne bhi agree kiya
I am Muhajir .. Aur mere lye sab se Pehly Pakistan he .. agr 10 lakh Altaf Jese leaders bh is zameen ki behurmati kren un sbko sar e aam phansi Deni chahye .. Proud to be a #Muhajir and #Pakistani	0	I am Muhajir Aur mere lye sab se Pehly Pakistan he agr 10 lakh Altaf Jese leaders bh is zameen ki behurmati kren un sbko sar e aam phansi Deni chahye Proud to be a Muhajir and Pakistani
Doctor sab sahi me ke PhD (in hate politics) wale. Bhai padhe likhe ho fir kyu ye sab baate karte ho. Tum bas bowling khelo, aur maje lo. pic.twitter.com/fk1qUbQstw	0	Doctor sab sahi me ke PhD in hate politics wale Bhai padhe likhe ho fir kyu ye sab baate karte ho Tum bas bowling khelo aur maje lo

Figure 12: Output after removing other special characters

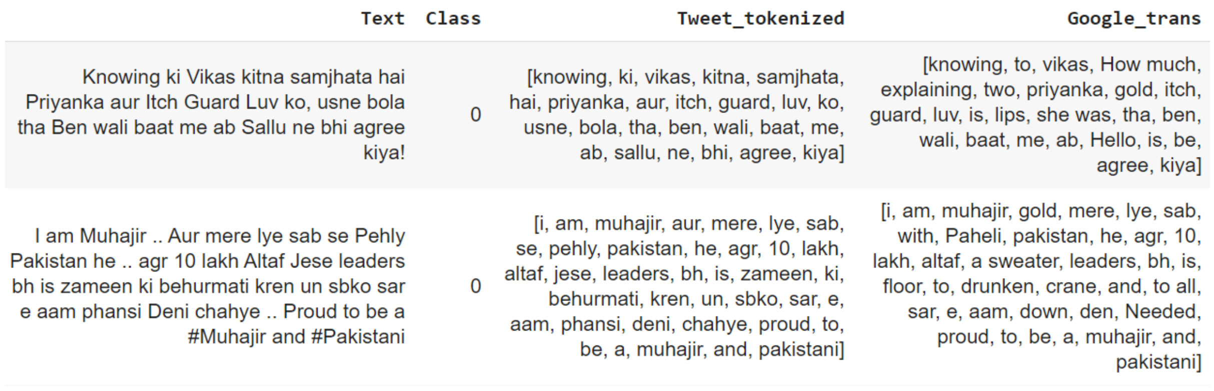
After the previous steps, the tweets were cleaned. We then converted the string to lowercase and tokenized them, by splitting each tweet into words.



Text	Class	Clean_tweet	Tweet_tokenized
Knowing ki Vikas kitna samjhata hai Priyanka aur Itch Guard Luv ko, usne bola tha Ben wali baat me ab Sallu ne bhi agree kiya!	0	Knowing ki Vikas kitna samjhata hai Priyanka aur Itch Guard Luv ko usne bola tha Ben wali baat me ab Sallu ne bhi agree kiya	[knowing, ki, vikas, kitna, samjhata, hai, priyanka, aur, itch, guard, luv, ko, usne, bola, tha, ben, wali, baat, me, ab, sallu, ne, bhi, agree, kiya]
I am Muhajir .. Aur mere lye sab se Pehly Pakistan he .. agr 10 lakh Altaf Jese leaders bh is zameen ki behurmati kren un sbko sar e aam phansi Deni chahye .. Proud to be a #Muhajir and #Pakistani	0	I am Muhajir Aur mere lye sab se Pehly Pakistan he agr 10 lakh Altaf Jese leaders bh is zameen ki behurmati kren un sbko sar e aam phansi Deni chahye Proud to be a Muhajir and Pakistani	[i, am, muhajir, aur, mere, lye, sab, se, pehly, pakistan, he, agr, 10, lakh, altaf, jese, leaders, bh, is, zameen, ki, behurmati, kren, un, sbko, sar, e, aam, phansi, deni, chahye, proud, to, be, a, muhajir, and, pakistani]

Figure 13: Tokenization

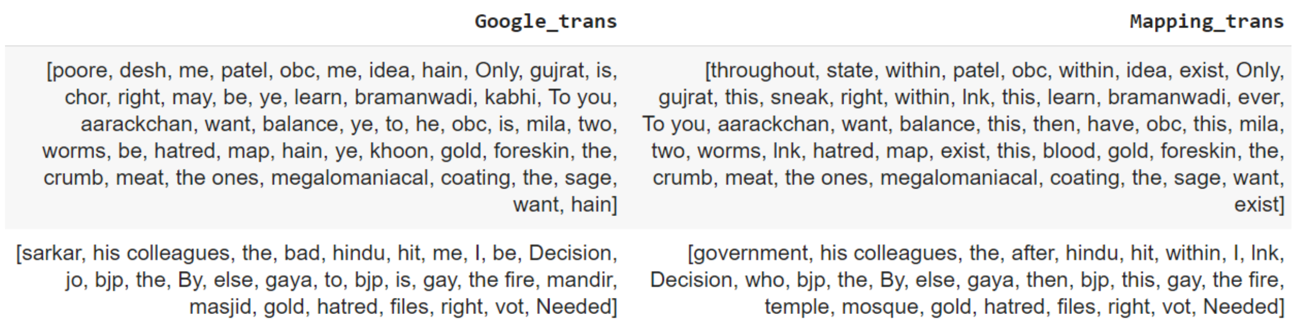
Each word in the tokenized list of each tweet was then converted using the Google Translate library.



Text	Class	Tweet_tokenized	Google_trans
Knowing ki Vikas kitna samjhata hai Priyanka aur Itch Guard Luv ko, usne bola tha Ben wali baat me ab Sallu ne bhi agree kiya!	0	[knowing, ki, vikas, kitna, samjhata, hai, priyanka, aur, itch, guard, luv, ko, usne, bola, tha, ben, wali, baat, me, ab, sallu, ne, bhi, agree, kiya]	[knowing, to, vikas, How much, explaining, two, priyanka, gold, itch, guard, luv, is, lips, she was, tha, ben, wali, baat, me, ab, Hello, is, be, agree, kiya]
I am Muhajir .. Aur mere lye sab se Pehly Pakistan he .. agr 10 lakh Altaf Jese leaders bh is zameen ki behurmati kren un sbko sar e aam phansi Deni chahye .. Proud to be a #Muhajir and #Pakistani	0	[i, am, muhajir, aur, mere, lye, sab, se, pehly, pakistan, he, agr, 10, lakh, altaf, jese, leaders, bh, is, zameen, ki, behurmati, kren, un, sbko, sar, e, aam, phansi, deni, chahye, proud, to, be, a, muhajir, and, pakistani]	[i, am, muhajir, gold, mere, lye, sab, with, Paheli, pakistan, he, agr, 10, lakh, altaf, a sweater, leaders, bh, is, floor, to, drunken, crane, and, to all, sar, e, aam, down, den, Needed, proud, to, be, a, muhajir, and, pakistani]

Figure 14: Google Translation Output

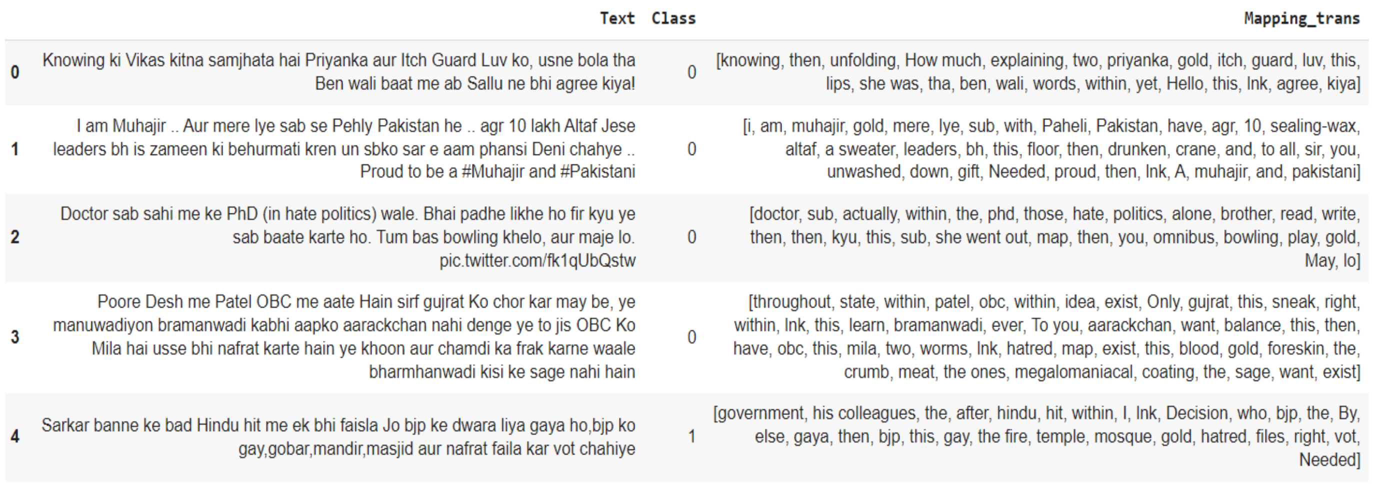
A Hinglish Dictionary was also used translation. The Google Translated tokenized tweets were again translated using the Hinglish Dictionary to yield more accurate translation results. The output of mapping translation done after google translation is visible below.



Google_trans	Mapping_trans
[poore, desh, me, patel, obc, me, idea, hain, Only, gujrat, is, chor, right, may, be, ye, learn, bramanwadi, kabhi, To you, aarackchan, want, balance, ye, to, he, obc, is, mila, two, worms, be, hatred, map, hain, ye, khoon, gold, foreskin, the, crumb, meat, the ones, megalomaniacal, coating, the, sage, want, hain]	[throughout, state, within, patel, obc, within, idea, exist, Only, gujrat, this, sneak, right, within, Ink, this, learn, bramanwadi, ever, To you, aarackchan, want, balance, this, then, have, obc, this, mila, two, worms, Ink, hatred, map, exist, this, blood, gold, foreskin, the, crumb, meat, the ones, megalomaniacal, coating, the, sage, want, exist]
[sarkar, his colleagues, the, bad, hindu, hit, me, I, be, Decision, jo, bjp, the, By, else, gaya, to, bjp, is, gay, the fire, mandir, masjid, gold, hatred, files, right, vot, Needed]	[government, his colleagues, the, after, hindu, hit, within, I, Ink, Decision, who, bjp, the, By, else, gaya, then, bjp, this, gay, the fire, temple, mosque, gold, hatred, files, right, vot, Needed]

Figure 15: Mapping Translation Output

After translation, we wrote and implemented a function to remove consecutively the same words from each list. The final column used in the model implementation is named "Mapping_trans" and the output is shown below. The changes done after preprocessing are clearly visible in the screenshot below.

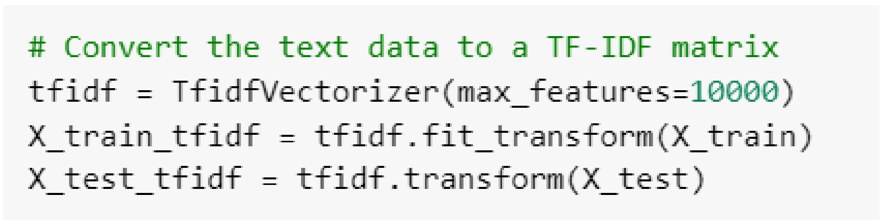


	Text	Class	Mapping_trans
0	Knowing ki Vikas kitna samjhata hai Priyanka aur Itch Guard Luv ko, usne bola tha Ben wali baat me ab Sallu ne bhi agree kiya!	0	[knowing, then, unfolding, How much, explaining, two, priyanka, gold, itch, guard, luv, this, lips, she was, tha, ben, wali, words, within, yet, Hello, this, Ink, agree, kiya]
1	I am Muhajir .. Aur mere lye sab se Pehly Pakistan he .. agr 10 lakh Altaf Jese leaders bh is zameen ki behurmati kren un sbko sar e aam phansi Deni chahye .. Proud to be a #Muhajir and #Pakistani	0	[i, am, muhajir, gold, mere, lye, sub, with, Paheli, Pakistan, have, agr, 10, sealing-wax, altaf, a sweater, leaders, bh, this, floor, then, drunken, crane, and, to all, sir, you, unwashed, down, gift, Needed, proud, then, Ink, A, muhajir, and, pakistani]
2	Doctor sab sahi me ke PhD (in hate politics) wale. Bhai padhe likhe ho fir kyu ye sab baate karte ho. Tum bas bowling khelo, aur maje lo. pic.twitter.com/fk1qUbQstw	0	[doctor, sub, actually, within, the, phd, those, hate, politics, alone, brother, read, write, then, then, kyu, this, sub, she went out, map, then, you, omnibus, bowling, play, gold, May, lo]
3	Poore Desh me Patel OBC me aate Hain sirf gujrat Ko chor kar may be, ye manuwadiyon bramanwadi kabhi aapko aarackchan nahi denge ye to jis OBC Ko Mila hai usse bhi nafrat karte hain ye khoon aur chamdi ka frak karne waale bharmhanwadi kisi ke sage nahi hain	0	[throughout, state, within, patel, obc, within, idea, exist, Only, gujrat, this, sneak, right, within, Ink, this, learn, bramanwadi, ever, To you, aarackchan, want, balance, this, then, have, obc, this, mila, two, worms, Ink, hatred, map, exist, this, blood, gold, foreskin, the, crumb, meat, the ones, megalomaniacal, coating, the, sage, want, exist]
4	Sarkar banne ke bad Hindu hit me ek bhi faisla Jo bjp ke dwara liya gaya ho,bjp ko gay,gobar,mandir,masjid aur nafrat faila kar vot chahiye	1	[government, his colleagues, the, after, hindu, hit, within, I, Ink, Decision, who, bjp, the, By, else, gaya, then, bjp, this, gay, the fire, temple, mosque, gold, hatred, files, right, vot, Needed]

Figure 16: Dataset after preprocessing

5.4. Machine Learning Models

Vectorization is done before the balancing.



```
# Convert the text data to a TF-IDF matrix
tfidf = TfidfVectorizer(max_features=10000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

Figure 17: TFIDF Vectorization

As seen in the screenshot below, the dataset is imbalanced. Because of this reason, we implemented SMOTE balancing.

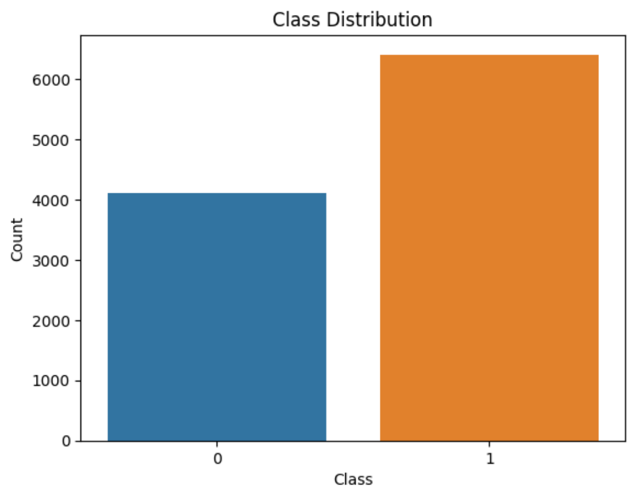


Figure 18: Before Balancing

The ratio of class 1 tweets to class 0 tweets after balancing is shown below. The dataset is now balanced.

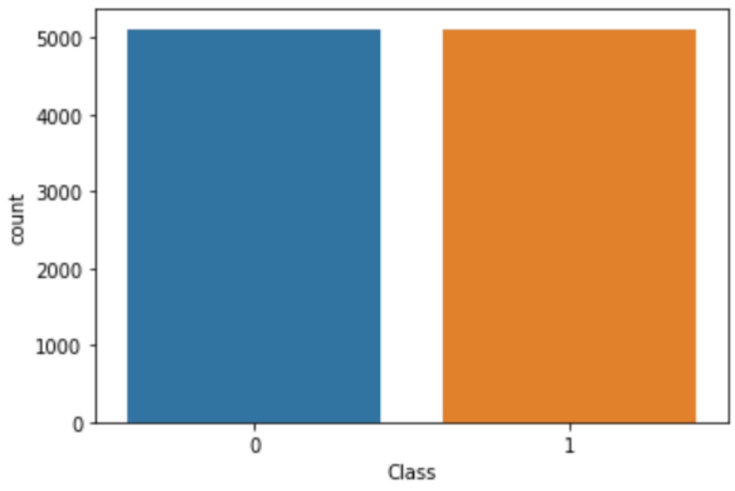
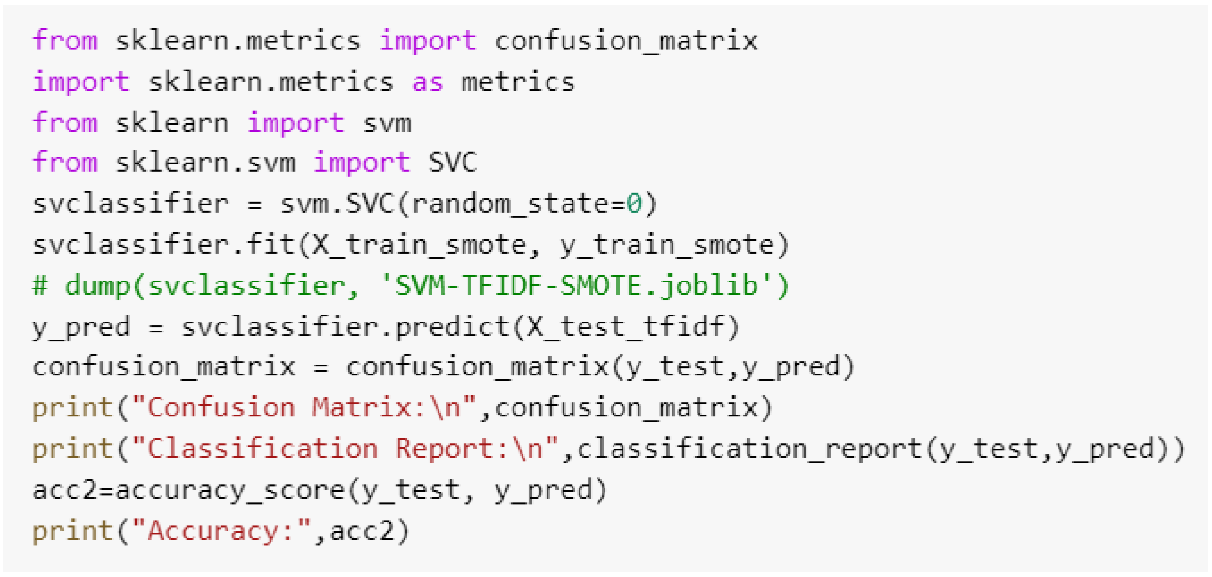


Figure 19: After SMOTE Balancing

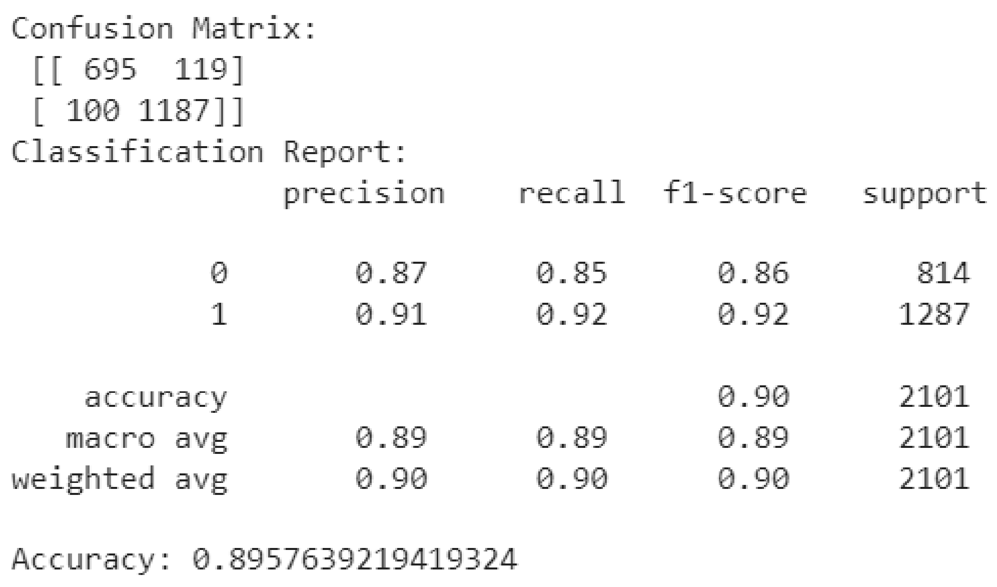
Out of all the Machine Learning models we implemented, SVM showed the most promising results. The code for the same is in the screenshot below.



```
from sklearn.metrics import confusion_matrix
import sklearn.metrics as metrics
from sklearn import svm
from sklearn.svm import SVC
svclassifier = svm.SVC(random_state=0)
svclassifier.fit(X_train_smote, y_train_smote)
# dump(svclassifier, 'SVM-TFIDF-SMOTE.joblib')
y_pred = svclassifier.predict(X_test_tfidf)
confusion_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n",confusion_matrix)
print("Classification Report:\n",classification_report(y_test,y_pred))
acc2=accuracy_score(y_test, y_pred)
print("Accuracy:",acc2)
```

Figure 20: SVM Model

The classification report for the SVM model is shown below.

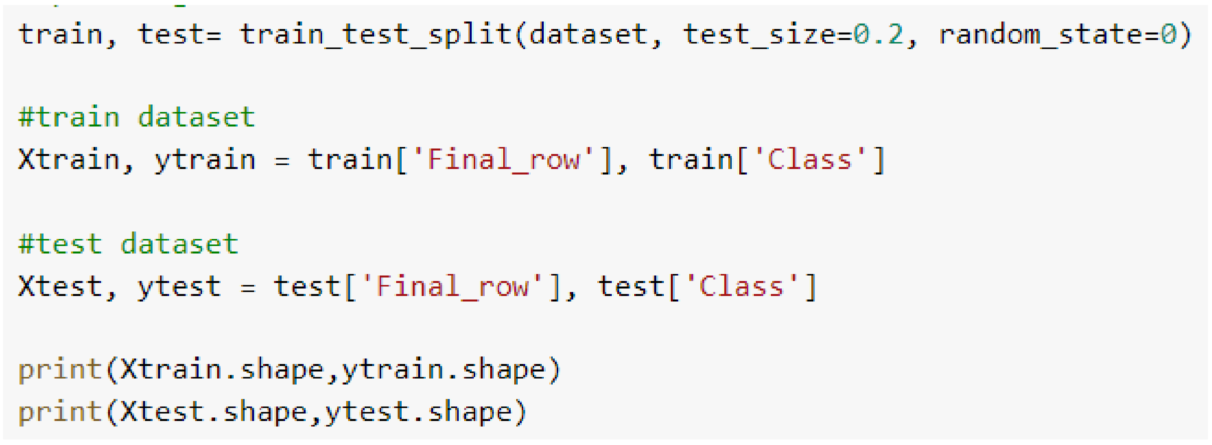


```
Confusion Matrix:
 [[ 695  119]
 [ 100 1187]]
Classification Report:
               precision    recall  f1-score   support

           0       0.87      0.85      0.86       814
           1       0.91      0.92      0.92      1287

    accuracy                           0.90      2101
   macro avg       0.89      0.89      0.89      2101
weighted avg       0.90      0.90      0.90      2101

Accuracy: 0.8957639219419324
```

Figure 21: SVM Results

5.5 Deep Learning Models

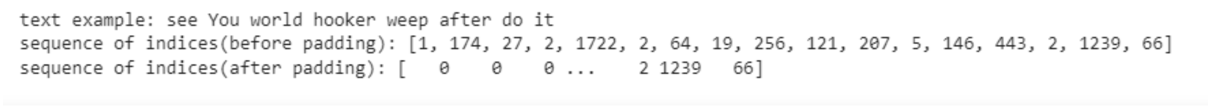
Splitting of data into train and test sets is shown below.



```
train, test= train_test_split(dataset, test_size=0.2, random_state=0)

#train dataset
Xtrain, ytrain = train['Final_row'], train['Class']

#test dataset
Xtest, ytest = test['Final_row'], test['Class']

print(Xtrain.shape,ytrain.shape)
print(Xtest.shape,ytest.shape)
```

Figure 22: Train Test Split

Padding and tokenization was done as well

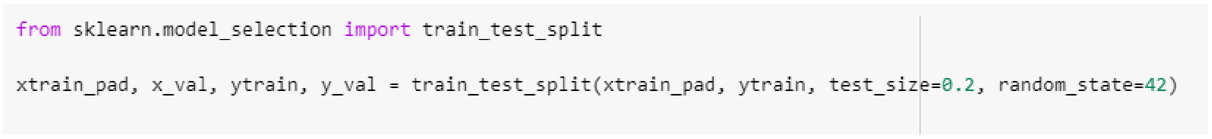


```
text example: see You world hooker weep after do it
sequence of indices(before padding): [1, 174, 27, 2, 1722, 2, 64, 19, 256, 121, 207, 5, 146, 443, 2, 1239, 66]
sequence of indices(after padding): [   0    0    0 ...    2 1239   66]
```

Figure 22: Tokenization and Padding

This above output signifies the tokenization and padding done on the text data to prepare it for training on a neural network and shows the difference of input before and after padding, that is adding zeros to make the length of input vectors same.

The training data was divided into train and validation sets further to avoid data leakage with training so that the test set is completely unknown to the model.

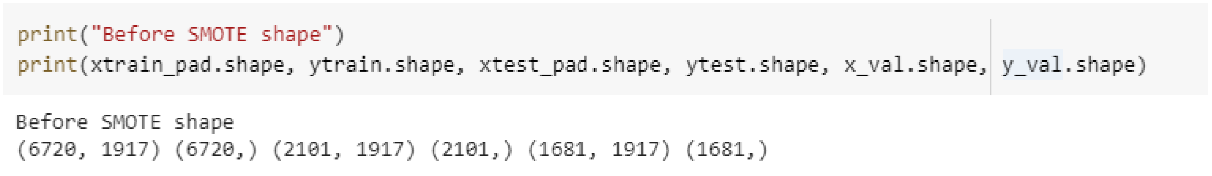


```
from sklearn.model_selection import train_test_split

xtrain_pad, x_val, ytrain, y_val = train_test_split(xtrain_pad, ytrain, test_size=0.2, random_state=42)
```

Figure 23: Splitting into train and validation set

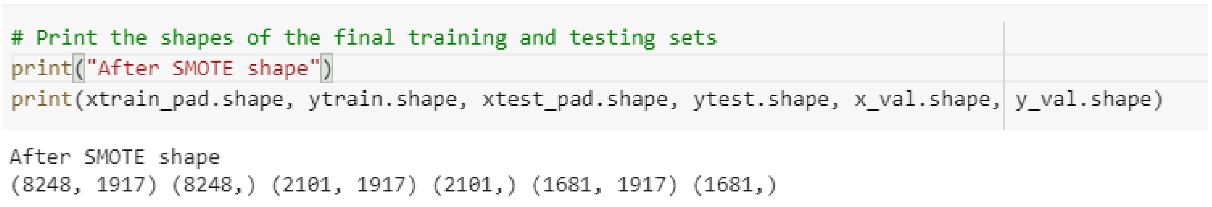
SMOTE balancing was done before implementing deep learning models as well.

Below are the screenshots of the data shapes before and after applying SMOTE for balancing.



```
print("Before SMOTE shape")
print(xtrain_pad.shape, ytrain.shape, xtest_pad.shape, ytest.shape, x_val.shape, y_val.shape)

Before SMOTE shape
(6720, 1917) (6720,) (2101, 1917) (2101,) (1681, 1917) (1681,)
```

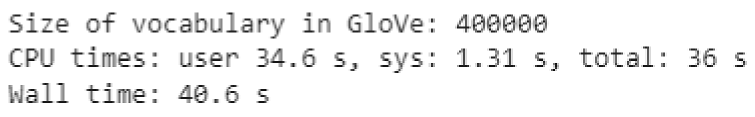
Figure 24: Before SMOTE Data Shape



```
# Print the shapes of the final training and testing sets
print("After SMOTE shape")
print(xtrain_pad.shape, ytrain.shape, xtest_pad.shape, ytest.shape, x_val.shape, y_val.shape)

After SMOTE shape
(8248, 1917) (8248,) (2101, 1917) (2101,) (1681, 1917) (1681,)
```

Figure 25: After SMOTE Data Shape

GLoVe embeddings were done. The embedding_vectors dictionary is initialized to store the word embeddings. The pre-trained GloVe embedding file is loaded using the open function and then read line by line. For each line in the file, the word and its corresponding weights (embedding vector) are extracted and stored in the embedding_vectors dictionary.

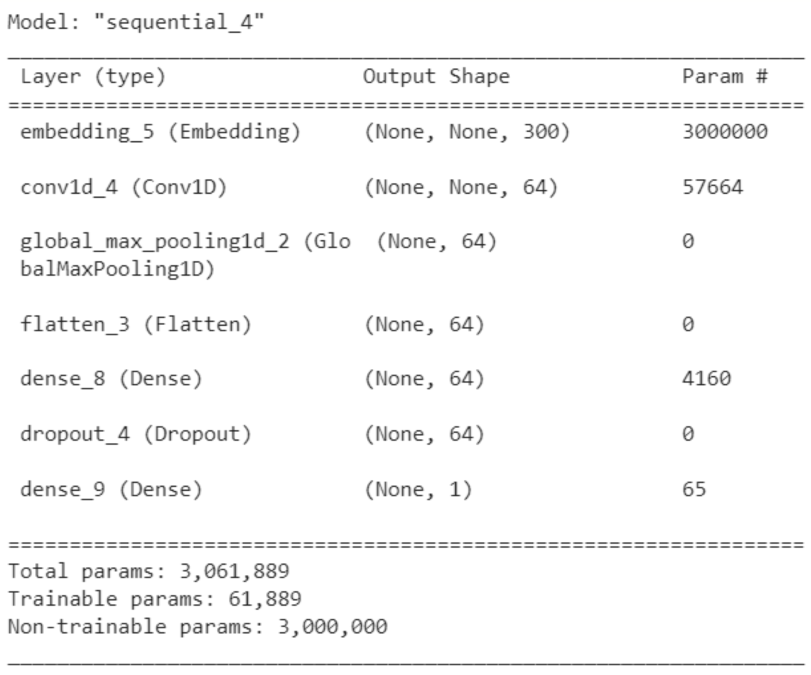


```
Size of vocabulary in GloVe: 400000
CPU times: user 34.6 s, sys: 1.31 s, total: 36 s
Wall time: 40.6 s
```

Figure 26: GLoVe Embeddings

GloVe embeddings contain 400,000 unique words, indicating the size of the vocabulary covered by the embeddings.

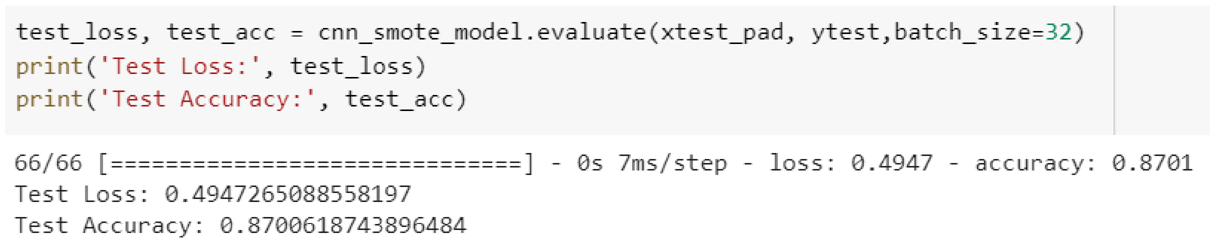
For deep learning, out of all models implemented, CNN showed us the best results. The architecture of the same is visible in the screenshot below.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, None, 300)         3000000

 conv1d_4 (Conv1D)           (None, None, 64)          57664

 global_max_pooling1d_2 (Glo  (None, 64)               0
 balMaxPooling1D)

 flatten_3 (Flatten)         (None, 64)                0

 dense_8 (Dense)             (None, 64)                4160

 dropout_4 (Dropout)         (None, 64)                0

 dense_9 (Dense)             (None, 1)                 65

=================================================================
Total params: 3,061,889
Trainable params: 61,889
Non-trainable params: 3,000,000
_________________________________________________________________
```

Figure 27: CNN Architecture

Evaluation of the model is shown below with the test loss and test accuracy displayed.

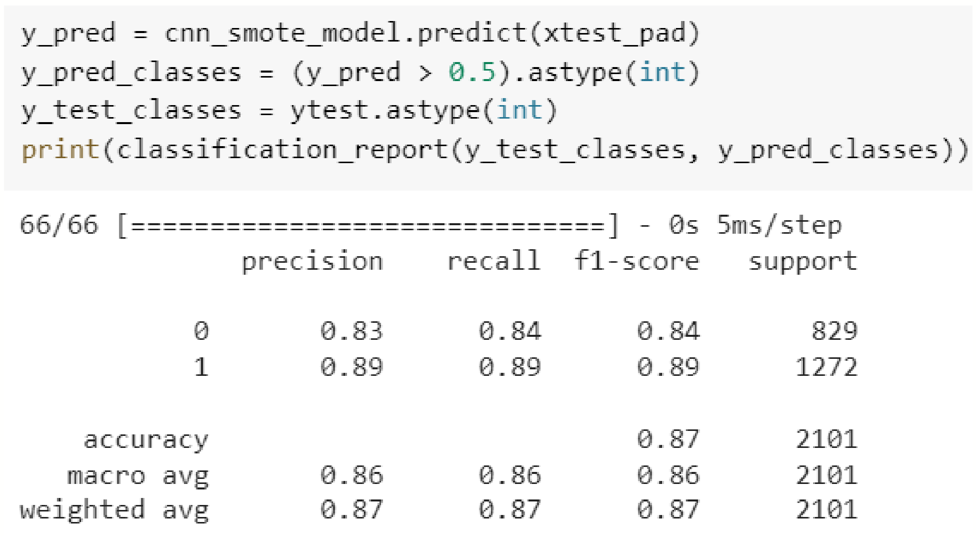


```
test_loss, test_acc = cnn_smote_model.evaluate(xtest_pad, ytest,batch_size=32)
print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
```

```
66/66 [==============================] - 0s 7ms/step - loss: 0.4947 - accuracy: 0.8701
Test Loss: 0.4947265088558197
Test Accuracy: 0.8700618743896484
```

Figure 28: CNN Results on Test Set

The classification report for CNN is shown below.



```
y_pred = cnn_smote_model.predict(xtest_pad)
y_pred_classes = (y_pred > 0.5).astype(int)
y_test_classes = ytest.astype(int)
print(classification_report(y_test_classes, y_pred_classes))
```

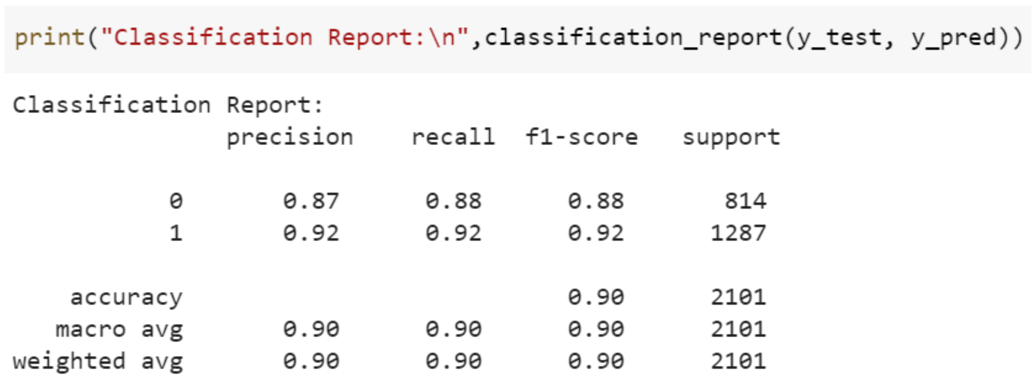
```
66/66 [==============================] - 0s 5ms/step
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       829
           1       0.89      0.89      0.89      1272

    accuracy                           0.87      2101
   macro avg       0.86      0.86      0.86      2101
weighted avg       0.87      0.87      0.87      2101
```

Figure 29: CNN Classification Report

5.6 Ensemble Models

Out of all ensemble models created, the stacking ensemble showed the best results. The code is as seen below, using RF and LR as estimators and SVM as the meta estimator.

The classification report of the stacking ensemble is shown below.



```
print("Classification Report:\n",classification_report(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.87      0.88      0.88       814
           1       0.92      0.92      0.92      1287

    accuracy                           0.90      2101
   macro avg       0.90      0.90      0.90      2101
weighted avg       0.90      0.90      0.90      2101
```

Figure 31: Stacking Classification Report

6 COMPARATIVE STUDY

6.1. Other Framework

For comparison, we implemented another framework on our dataset and compared the results. The framework we implemented is shown below.

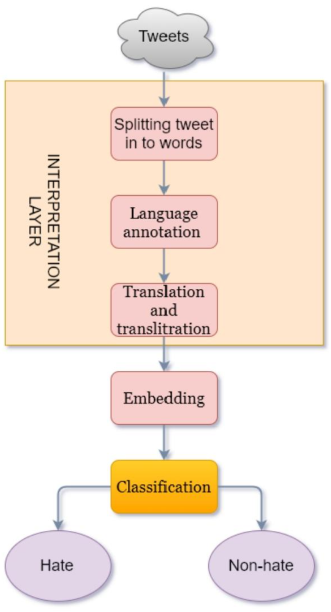


Figure 32: Other Framework

This framework was implemented by [3].

6.2. Result Comparison

The results achieved in both the frameworks are specified in the table below:

Model	Accuracy	
	Our Framework	Other Framework
CNN	0.87	0.79
LSTM	0.82	0.76
GRU	0.85	0.77

Table II. comparison

As observed in the table, our framework yields better results as compared to the previously created and tried framework on the dataset created by us. This may be due to many reasons

- This framework did not handle emojis and other noise such as converting certain special characters used inside the words to letters and hence this could have been an issue when translating which we have dealt with in our framework.
- The dataset this framework was tested on was a much smaller dataset and hence failed to handle the noise of a larger dataset.
- A lot of words lose their meaning while transliteration to devanagari text due to grammatical errors occurring while changing from roman to devanagari script.

7 RESULTS AND DISCUSSIONS

7.1. Final Results on Proposed Framework

Model	**Accuracy**	**F1 Score**	**Recall**	**Precision**
Machine Learning				
LR	0.88	0.88	0.88	0.89
RF	0.88	0.89	0.89	0.89
SVM	**0.89**	**0.90**	**0.90**	**0.90**
Deep Learning				
LSTM	0.82	0.83	0.83	0.83
GRU	0.85	0.85	0.85	0.85
CNN	**0.87**	**0.87**	**0.87**	**0.87**
RNN	0.84	0.85	0.85	0.85
Machine Learning Ensemble				
Bagging	0.89	0.90	0.90	0.90
Stacking	**0.90**	**0.90**	**0.90**	**0.90**

Table III. final results

7.2. Challenges:

- **Context dependency:**

  The meaning of words in hindi is context dependent, meaning the same word might have different meanings if used differently. For example, the term “pakka” can mean “sure” in some sentences while it can also mean “cooked” if used differently.

- **Similar words in english and hindi:**

  Some words in hindi that are written in roman script are similar to already existing english words and hence the translation here gets confused. For example, the term “uss” in hindi means “that” in english but is also very similar to “us” in english, and hence while detecting the language of this word and translating, it does not get translated to that because it is already considered as an english word.

- **No standardization:**

  There are no fixed spellings for hinglish and hence users write and use words according to their convenience. For example- “mein” and “mei” both mean me but have been written with different spellings and hence do not get translated accurately

8 CONCLUSION

8.1. Conclusion and Future Avenues

The proposed approach investigates hate speech identification in a Hindi-English code-mixed Twitter data set. In this project, we have created a large dataset consisting of hinglish hate speech tweets that was not present earlier and has been mentioned as a drawback in most of the previously conducted research in this domain. We proposed the Translation Based Preprocessing Framework for hate speech identification that translates the entire hinglish tweet to monolingual English roman script for better results on the problem at hand.

Since we have addressed the issue of a large dataset present for this research, in the near future researchers can use this dataset and try different frameworks that are better at translation and can address the challenges we faced. The use of transformer based models has proved to provide better results on similar tasks and can also be tried on this dataset to see if it outperforms the models that we have tried. Use of character level embeddings can also be tried in the future for this task.

REFERENCES

[1] Si, S., Datta, A., Banerjee, S., & Naskar, S. K. (2019, July). Aggression detection on multilingual social media text. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-5). IEEE.

[2] Gupta, V., Sehra, V., & Vardhan, Y. R. (2021, May). Ensemble Based Hinglish Hate Speech Detection. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1800-1806). IEEE.

[3] Biradar, S., Saumya, S., & Chauhan, A. (2021, December). Hate or non-hate: Translation based hate speech identification in code-mixed hinglish data set. In *2021 IEEE International Conference on Big Data (Big Data)* (pp. 2470-2475). IEEE.

[4] Malte, A., & Ratadiya, P. (2019, October). Multilingual cyber abuse detection using advanced transformer architecture. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)* (pp. 784-789). IEEE.

[5] Kumar, V., Pasari, S., Patil, V. P., & Seniaray, S. (2020, July). Machine Learning based Language Modelling of Code Switched Data. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)* (pp. 552-557). IEEE.

[6] Maity, K., Saha, S., & Bhattacharyya, P. (2022). Emoji, Sentiment and Emotion Aided Cyberbullying Detection in Hinglish. *IEEE Transactions on Computational Social Systems*.

[7] Banerjee, S., Chakravarthi, B. R., & McCrae, J. P. (2020, December). Comparison of pretrained embeddings to identify hate speech in Indian code-mixed text. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)* (pp. 21-25). IEEE.

[8] Gupta, V., Sehra, V., & Vardhan, Y. R. (2021, April). Hindi-English Code Mixed Hate Speech Detection using Character Level Embeddings. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1112-1118). IEEE.

[9] Schmidt, A., & Wiegand, M. (2017, April). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media* (pp. 1-10).

[10] Ganfure, G. O. (2022). Comparative analysis of deep learning based Afaan Oromo hate speech detection. *Journal of Big Data*, *9*(1), 1-13.

[11] Nikhil, N., Pahwa, R., Nirala, M. K., & Khilnani, R. (2018, August). Lstms with attention for aggression detection. In *Proceedings of the first workshop on trolling, aggression and cyberbullying (TRAC-2018)* (pp. 52-57).

[12] Paschalides, D., Stephanidis, D., Andreou, A., Orphanou, K., Pallis, G., Dikaiakos, M. D., & Markatos, E. (2020). Mandola: A big-data processing and visualization platform for monitoring and detecting online hate speech. *ACM Transactions on Internet Technology (TOIT)*, *20*(2), 1-21.

[13] Kovács, G., Alonso, P., & Saini, R. (2021). Challenges of hate speech detection in social media. *SN Computer Science*, *2*(2), 1-15.

[14] Corazza, M., Menini, S., Cabrio, E., Tonelli, S., & Villata, S. (2020). A multilingual evaluation for online hate speech detection. *ACM Transactions on Internet Technology (TOIT)*, *20*(2), 1-22.

[15] Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In *Proceedings of the 26th international conference on World Wide Web companion* (pp. 759-760).

[16] Khan, M. M., Shahzad, K., & Malik, M. K. (2021). Hate speech detection in roman urdu. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, *20*(1), 1-19.

[17] Alorainy, W., Burnap, P., Liu, H., & Williams, M. L. (2019). "The enemy among us" detecting cyber hate speech with threats-based othering language embeddings. *ACM Transactions on the Web (TWEB)*, *13*(3), 1-26.

[18] Joshi, R., Karnavat, R., Jirapure, K., & Joshi, R. (2021, April). Evaluation of deep learning models for hostility detection in hindi text. In *2021 6th International Conference for Convergence in Technology (I2CT)* (pp. 1-5). IEEE.

[19] Saleem, A. M. S. M., & Gawande, K. (2021, June). Coercive Behaviour Catching: A Real Time Analysis Approach. In *2021 International Conference on Communication information and Computing Technology (ICCICT)* (pp. 1-5). IEEE.

[20] Herodotou, H., Chatzakou, D., & Kourtellis, N. (2021, April). Catching them red-handed: Real-time aggression detection on social media. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (pp. 2123-2128). IEEE.

[21] Mathur, P., Shah, R., Sawhney, R., & Mahata, D. (2018, July). Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media* (pp. 18-26).

[22] Mathur, P., Sawhney, R., Ayyar, M., & Shah, R. (2018, October). Did you offend me? classification of offensive tweets in hinglish language. In *Proceedings of the 2nd workshop on abusive language online (ALW2)* (pp. 138-148).

[23] Chopra, S., Sawhney, R., Mathur, P., & Shah, R. R. (2020, April). Hindi-english hate speech detection: Author profiling, debiasing, and practical perspectives. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 01, pp.

386-393).

[24] Mullah, N. S., & Zainon, W. M. N. W. (2021). Advances in machine learning algorithms for hate speech detection in social media: a review. *IEEE Access*

[25] Biradar, S., & Saumya, S. (2022). Fighting hate speech from a bilingual hinglish speaker's perspective, a transformer-and translation-based approach. Social Network Analysis and Mining, 12(1),