DEVELOPING A SENTIMENT ANALYSIS MODEL FOR CODE-MIXED HINDI-ENGLISH (HINGLISH) TEXT

A Project Report submitted

In fulfillment of the requirements for the degree of

B.Tech (Computer Science & Engineering)

Submitted by :-

Pulakala Prithvi Raj (222025042)

Pankaj Biswas (222025043)

Pritisha Goswami (222025049)

B.Tech Computer Science and Engineering

8th Semester

Royal School of Engineering and Technology (RSET)

Under the

guidance of

Dr. Dillip Rout

Assistant Professor

Royal School of Engineering and Technology (RSET)

THE ASSAM ROYAL GLOBAL UNIVERSITY

GUWAHATI: 781035

Session: 2022-2026

CERTIFICATE OF APPROVAL

This is to certify that the project report entitled *"Developing a Sentiment Analysis Model for Code-Mixed Hindi-English (Hinglish) Text"* submitted by **Pulakala Prithvi Raj** (Roll No. 222025042), **Pankaj Biswas** (Roll No. 222025043) and **Pritisha Goswami** (Roll No. 222025049), students of B.Tech, 8th semester in the Department of Computer Science & Engineering, Royal School of Engineering and Technology (RSET), The Assam Royal Global University, Guwahati, Assam, has been completed under my supervision. This work is submitted as part of the requirements for the award of the B.Tech degree in Computer Science & Engineering and has not been submitted elsewhere for a degree.

_____ _____

Project Guide: **Signature of the External**

Dr. Dillip Rout **Name of the External**

Assistant Professor, CSE, RSET

Date:

Place: Guwahati

FORWARDING CERTIFICATE

This is to certify that the project report entitled *"Developing a Sentiment Analysis Model for Code-Mixed Hindi-English (Hinglish) Text"* submitted by **Pulakala Prithvi Raj** (Roll No. 222025042), **Pankaj Biswas** (Roll No. 222025043) and **Pritisha Goswami** (Roll No. 222025049), students of B.Tech, 8th semester in the Department of Computer Science & Engineering at Royal School of Engineering and Technology (RSET), The Assam Royal Global University, Guwahati, Assam, under the guidance of **Dr. Dillip Rout, Assistant Professor**, has been evaluated and deemed satisfactory for submission as a requirement for the degree program.

Date:

Place: Guwahati

Dr. Dillip Rout
Assistant Professor
Department of CSE

Royal School of Engineering & Technology

DECLARATION

We, **Pulakala Prithvi Raj** (Roll No. 222025042), **Pankaj Biswas** (Roll No. 222025043) and **Pritisha Goswami** (Roll No. 222025049), hereby declare that the project work entitled *"Developing a Sentiment Analysis Model for Code-Mixed Hindi-English (Hinglish) Text"* was carried out by us under the guidance and supervision of **Dr. Dillip Rout, Assistant Professor**, **Department of Computer Science & Engineering**. This project is submitted for the academic session 2022-2026. We confirm that this work, or any part of it, has not been submitted elsewhere for any other purpose to date.

Date:

Place: Guwahati

Pulakala Prithvi Raj	**Pankaj Biswas**	**Pritisha Goswami**
222025042	**222025043**	**222025049**

ACKNOWLEDGMENT

We are deeply grateful to Royal School of Engineering and Technology for providing us with the resources and environment needed to complete this project titled, *"Developing a Sentiment Analysis Model for Code-Mixed Hindi-English (Hinglish) Text"*.

We would like to extend our heartfelt thanks to our guide, Dr. Dillip Rout, Assistant Professor, Department of CSE, Royal School of Engineering and Technology, whose invaluable guidance, encouragement, and insightful feedback have been crucial throughout the project's development. His support enabled us to navigate complex challenges and explore new dimensions in the field of deep learning.

We are also grateful to all faculty members who offered their support, advice, and assistance, both directly and indirectly. Their guidance has played an essential role in shaping this work.

I wish to express my gratitude to my family and friends, whose constant support, encouragement, and patience have been a source of strength throughout this journey. Without their belief in my abilities, this work would not have been possible.

Thank you

Pulakala Prithvi Raj
222025042

Pankaj Biswas
222025043

Pritisha Goswami
222025049

ABSTRACT

Code-mixed languages, such as Hinglish which is an informal infusion of Hindi and English cause significant challenges for sentiment analysis due to inconsistent grammar, transliteration variations, and limited annotated resources. This study presents a comprehensive comparison of classical machine learning, hybrid models and deep learning architectures, including the transformers for binary sentiment classification of Hinglish text. The analysis utilizes the PRISM dataset, comprising 29,550 Hinglish samples labeled as non-hate (0) or hate (1) sources from Kaggle. The text preprocessing included removing URLs, eliminating mentions and hashtags, and normalizing whitespace. Seventeen models were implemented: Word2Vec, GloVe, FastText, USE, ELMo used with both LSTM and LightGBM for Regular Training. Whereas, GloVe+BiLSTM, FastText+BiLSTM, Word2Vec+BiLSTM, MuRIL, mBART, HingRoBERTa-mixed and MPNet these models were used for both regular training and multistage language training include six variations of multistage training specially with GloVe+BiLSTM and MuRIL. Evaluation metrics include Accuracy, Balanced Accuracy, Precision, Recall, F1-score, Specificity, and AUC-ROC to examine the robustness of the models. Multi-stage language training results indicate that MPNet achieved the highest F1 Score (0.720) and AUC (0.820), and MuRIL achieved highest accuracy (0.724) in (Hinglish->Hindi->English->Full) variation highlighting the efficacy of multilingual transformers for modeling code-mixed text. Classical models performed worse, though GloVe+BiLSTM outperformed the Word2Vec and FastText baselines. The proposed model also surpasses the projects available on Kaggle. The findings emphasize the importance of multiple model evaluations for robust sentiment classification of low-resource, code-mixed social media data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Hinglish, which is a blend of English (*Latin*) and Hindi (*Latin*), mainly used in India as informal conversations, which presents unique challenges for Natural Language Processing (NLP) because of spelling variations and informal grammar, sarcastic contexts and frequent code-switching within sentences. Preliminary research was majorly focused on creating annotating the corpuses for Hinglish to assist supervised learning approaches. These datasets commonly included social media posts, its comments, and chat messages mainly informal ones showing code-mixing at the lexical and syntactic levels. Researchers investigated language identification as an initial step, distinguishing English, Hindi, and mixed tokens, which is essential for efficient sentiment analysis. Earlier studies also used Classical Machine Learning classifier models such as Naïve Bayes, Decision Trees, and SVM using features like n-grams, part-of-speech tags, and lexicons optimized for code-mixed text. Recent studies have applied deep learning models like Long Short-Term Memory (LSTM) models, Recurrent Neural Networks (RNNs) models and Transformer-based models which are more relevant for contextual understanding in comparison to Classical Machine Learning classifier models in code-mixed sentences.

Many works have also shown the challenges in Code-Mixed categories that widely included transliteration and normalization, about Hindi words written in Latin script with irregular spelling as usually there are multiple spelling variations for single Hindi word. Various techniques such as embedding-based representations and pronunciation-based comparison have been put forward to deal with these variations effectively. Overall, the background research highlights the complexity of sentiment analysis in Hinglish due to linguistic divergence, lack of normalized spelling system, and the dynamic nature of code-switching. This has influenced the development of specialized datasets, feature extraction methods, and model architectures optimized to the distinctions of code-mixed language.

1.2 PROBLEM STATEMENT

The swift growth of social media platforms has resulted in a notable rise in user contents,

which is generally written in code-mixed languages that blend multiple languages within a single sentence. Hinglish, a mix of English and Hindi written in Latin script, is widely seen on platforms like Facebook, YouTube comments, Twitter comments, and Instagram comments. Analysing the sentiment of such code-mixed text is inherently difficult due to irregular grammar, inconsistencies in transliteration, spelling variations, and a lack of annotated datasets. Classical Machine Learning models, such as Word2Vec and FastText embeddings paired with linear classifiers, that provide computational efficiency but often struggle to capture the complex contextual information in code-mixed text as these types of models are usually of Static embeddings technique. Whereas, Deep Learning models, including recurrent neural networks (RNNs), bidirectional long short-term memory networks (BiLSTM), and transformer-based multilingual encoders (MuRIL), are proficient at complex contextual understandings.

1.3 MOTIVATION

This research offers an in-depth comparative study of machine learning and deep learning techniques applied to Hinglish code-mixing for the classification of hate and non-hate speech. It leverages the publicly accessible PRISM dataset, which includes 29950 entries [16], for hate-speech detection. The methodology proposed in this study fills the gap in comparing multiple models using various performance metrics such as Accuracy, Balanced Accuracy, Precision, Recall, Specificity, F1 Score, and ROC-AUC Score. The research involves four models that encompass both traditional and deep learning frameworks. The goal is to pinpoint models that exhibit strong performance in binary sentiment classification of Hinglish code-mixed text and to shed light on the benefits of combining different modeling approaches. Additionally, the study aims to evaluate the transformer model without training and to assess the significance of preprocessing.

1.4 OBJECTIVES

To develop and optimize NLP architectures for accurate Hinglish sentiment analysis, begin with preparing a comprehensive annotated dataset and generating multiple embedding representations, including pretrained transformer embeddings (MuRIL, mBART, HingRoBERTa, MPNet) , traditional embeddings combined with BiLSTM (Word2Vec, GloVe, FastText) and embeddings combined with LSTM (Word2Vec,

Glove, FastText, Elmo, USE) . Consistent preprocessing steps such as tokenization, transliteration normalization, and language identification are essential. Establish a baseline framework using simpler models like Word2Vec+BiLSTM or GloVe+BiLSTM alongside classical classifiers such as LIGHTGBM to set reference performance metrics. Fine-tune transformer-based models on the Hinglish dataset while optimizing hyperparameters, and similarly train and tune BiLSTM layers with traditional embeddings. For LIGHTGBM, we have used embedding features (like Word2Vec,FastText,Glove,Elmo,USE) as input and optimize boosting the parameters.

Implement a multi-phase language-specific training strategy by splitting the dataset into English, Hindi, and Hinglish subsets, sequentially training and fine-tuning models across these phases with independent validation to monitor performance shifts. Analyze cross-language adaptation by evaluating performance consistency, knowledge transfer, and adaptation capability after each retraining phase. Following multi-phase training, assess robustness and generalization on a unified test corpus, including noisy and informal samples and varied sentiment-emotion categories. Use iterative refinement to adjust architectures, embeddings, and training strategies, considering ensemble approaches that combine transformer LSTM and BiLSTM-based models to enhance accuracy and build a robust Hinglish sentiment-emotion analysis system.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

Research in code-mixed sentiment analysis, particularly for Hinglish (Hindi–English mixed text), has witnessed substantial growth over the last decade, evolving from simple lexicon-based techniques to sophisticated transformer-driven architectures. Code-mixed text poses unique challenges due to transliteration variations, inconsistent grammar, and a lack of large annotated corpora. Earlier studies primarily relied on lexicons and statistical models, while recent approaches emphasize deep learning and multilingual transformer models that better capture bilingual context and semantics.

The earliest work in this domain explored classical machine learning approaches using handcrafted linguistic and statistical features. Singh analyzed code-mixed social media text using Naïve Bayes and SVM classifiers, revealing that token-level language identification and transliteration inconsistencies greatly impacted sentiment accuracy [1]. Similarly, Thakur et al. outlined that while traditional methods offered moderate accuracy, they lacked scalability and contextual depth [2]. These early models were limited in their ability to handle non-standardized language usage and failed to capture more complex syntactic relationships.

A significant shift occurred with the adoption of embedding-based representations that moved beyond sparse lexical features. Techniques such as Word2Vec and GloVe introduced dense vector representations capable of encoding semantic similarity. The study by Agarwal (2024) demonstrated that integrating CNN and BiLSTM with pretrained embeddings significantly improved sentiment detection accuracy for Hinglish text [3]. However, these embeddings were static and unable to account for polysemy or word sense variations, leading to limited performance in diverse contexts.

In subsequent years, deep neural architectures like RNNs and LSTMs became prominent for modeling sequential dependencies within sentences. However, these models struggled with long-term contextual understanding and required large labeled corpora for practical training. The introduction of transformer-based architectures revolutionized natural language processing by replacing sequential recurrence with self-attention, enabling parallel processing of long-range dependencies. This innovation paved the way for transformer-based contextual embeddings such as BERT

and MuRIL, which excel at handling code-mixed and multilingual data.

Singh et al. presented a study on sentiments in Code-Mixed texts, validating the effectiveness of transformer-based architectures in understanding sentiment polarity and emotion intensity in bilingual texts [4]. Similarly, a hybrid attention-based mechanism that outperformed CNN and RNN baselines was proposed, proving that contextualized embeddings substantially improve cross-lingual generalization [5]. These findings confirmed that contextual modeling plays a vital role in decoding the semantics of Hinglish text, where literal translations are insufficient for accurate sentiment recognition.

The challenge of data scarcity and domain imbalance in Hinglish corpora was addressed by Yadav et al. (2024), who employed weak supervision and semi-supervised techniques to enhance dataset diversity and reduce annotation cost [6]. Similarly, Aggarwal et al. showcased how deep contextual encoders successfully captured sarcasm and implicit sentiment polarity in Hinglish [7]. Generally, traditional sentiment models often misclassify or are inefficient at capturing these aspects of sentiment analysis; however, deep learning models achieve satisfactory results. These studies highlight the evolution from sentiment-level to emotion and sarcasm-level understanding in code-mixed research.

The rise of ensemble-based architectures has further pushed the boundaries of Hinglish sentiment and hate-speech classification. Gupta et al. (2021) illustrated that integrating outputs from multiple deep learning and transformer models achieved higher recall and robustness than individual networks [8]. Similarly, a combination of several transformer models was deployed to capture the varied contextual nuances and linguistic cues, resulting in improved detection accuracy [9]. Moreover, Aloria et al. (2023) further emphasized that attention-based transformers outperform traditional CNN or RNN frameworks, underscoring the superiority of contextual understanding in handling humor, irony, and sarcasm [10].

Recent studies have extended Hinglish sentiment analysis to broader tasks, such as multilingual emotion recognition and affective computing. A study used a multilingual transformer pipeline to analyze complex code-mixed expressions, reporting enhanced accuracy through contextual embeddings and domain adaptation [11]. Similarly, Baruah et al. developed a BiLSTM-based architecture optimized for mixed-script input, achieving notable improvements in recognizing emotion intensity and polarity [12]. Both studies underline that domain-specific pretraining and attention-based architectures significantly improve emotion recognition in low-resource settings. Furthermore, Paul et al. introduced a sentiment dynamics framework that integrates

attention layers to visualize the flow of emotions in bilingual texts [17]. This research demonstrated that attention mechanisms not only improve model interpretability but also help localize sentiment-bearing tokens in code-mixed data. Likewise, a review of Code-Mixed Sentiment Analysis shows that a multi-layered CNN-BiLSTM model achieves higher accuracy on benchmark Hinglish datasets by leveraging word embeddings and sentiment lexicons [18]. Additionally, recent efforts have focused on addressing linguistic variability and transliteration inconsistencies. A study on challenges in Code-Mixed NLP highlighted the limitations of tokenization, spelling variation, and the representation of Romanized Hindi, underscoring the need for data normalization before model training [19]. This work provides an essential foundation for preprocessing strategies in Hinglish NLP pipelines.

From the literature reviewed, it is evident that the field has undergone a clear methodological evolution from feature-engineered machine learning models to embedding-based, deep learning, and transformer-driven architectures. Early models offered interpretability but struggled with the complex semantics of bilingual text. Embedding models improved word-level representation but lacked contextual flexibility. Deep neural networks, such as CNN–BiLSTM, enhanced sequential understanding but required substantial labeled data. In contrast, transformer-based multilingual encoders such as MuRIL deliver superior contextual sensitivity, cross-lingual adaptability, and robustness to transliteration noise. Furthermore, ensemble and hybrid architectures combining these models continue to outperform standalone systems, offering a comprehensive solution to the nuances of Hinglish text processing. However, despite notable progress, a gap persists in standardized benchmarking and comparative evaluation across models. Hence, the present research aims to bridge this gap by systematically evaluating diverse architectures — including MuRIL, CNN–BiLSTM, GloVe–BiLSTM, and Word2Vec–Logistic Regression — on a unified Hinglish sentiment dataset to propose a robust multimodel framework for effective sentiment and emotion classification.

Table 2.1: Literature Survey

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
1	Sentiment Analysis of Code-Mixed Social Media Text (Hinglish)	Gaurav Singh[1]	arXiv (Preprint)	2021	Early Hinglish sentiment analysis; accuracy affected by transliteration inconsistency and token errors.	The necessity for generic sentiment analysis models specifically designed to address the grammatical and lexical intricacies of code-mixed Hinglish material.
2	Current State of Hinglish	Varsha Thakur, Roshani Sahu and Somya Omer[2]	SSRN Electronic Journal	2020	Traditional models showed limited contextual depth and poor scalability	The lack of a thorough assessment and organization of existing approaches, problems, and the latest technological advancements in Hinglish sentiment analysis.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
3	Improving Sentiment Analysis	Prof Neha Agarwal, Viraj Shah , Rishikesh Sharma , Himanshu Yadav, Vaibhav Shah[3]	Educational Administration: Theory and Practice	2024	Hybrid deep learning enhanced accuracy compared to classical ML	Current models exhibit constrained accuracy in processing Hinglish; thus, there is a necessity for hybrid deep learning architectures to enhance performance.
4	**Predicting Multi Label emojis,Emotions, and sentiments in code-mixed Texts using an emojifying**	Gopendra Vikram Singh, Soumitra Ghosh, Mauajana Firdaus, Asif Ekbal, Pushpak Bhattacharya[4]	Scientific Reports	2024	Multilabel emotion and sentiment analysis improved via transformer-based contextual modeling.	Typical models usually forecast only one label (for instance, sentiment alone). There exists a deficiency in the ability to concurrently predict the interrelated aspects of emojis, various emotions, and sentiments in code-mixed language.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
5	A self-Attention hybrid emoji prediction model for code-mixed language	Gadde Satya Sai Naga Himabindu ,Rajat Rao, Divyasikha Sethiya[5]	Social Network Analysis and Mining	2022	Contextualized embeddings outperformed CNN/RNN; effective in emoji prediction for Hinglish.	Predicting emojis in code-mixed Hinglish with precision is challenging, necessitating specific models that implement self-attention strategies to grasp combined semantic meanings.
6	Leveraging weakly annotated data for hate speech detection in code-mixed Hinglish: A feasibility-driven transfer learning approach with Large Language Models. In arXiv [cs.CL].	Sargam Yadav, Abishek Kaushik, Kevin McDaid,[6]	arXiv (Preprint)	2024	Used weak supervision to improve dataset diversity and mitigate annotation cost	The critical shortage of well-labeled datasets for Hinglish hate speech requires strategies that utilize poorly annotated data through Large Language Models (LLMs) and transfer learning.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
7	"Did you really mean what you said?" : Sarcasm Detection in Hindi-English Code-Mixed Data using Bilingual Word Embeddings. In arXiv [cs.CL].	Akshita Agarwal,Anshul Wadhawan, Ashima Choudhury, Kavita Mourya, [7]	arXiv (Preprint)	2020	Captured sarcasm and implicit polarity in bilingual data effectively.	Recognition of sarcasm in Hinglish frequently does not succeed with typical monolingual word representations, necessitating the use of tailored bilingual word embeddings to understand irony across languages.
8	Ensemble based hinglish hate speech detection. 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS).	Rahul, Gupta, V., Sehra, V., & Vardhan, Y. R. [8]	ICICCS 2021 (Conference)	2021	Ensemble boosted recall and robustness compared to single models.	. Independent models show limited effectiveness in identifying hate speech within noisy Hinglish datasets; collective techniques are necessary to combine their predictive strengths.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
9	Ensemble learning-based sarcasm detection in hinglish tweets using Word2Vec embedding. 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), 1–6.	Acharya, A., & Goyal, R. [9]	IATMSI 2025 (Conference)	2025	Combined transformer outputs achieved improved sarcasm detection accuracy.	The requirement to integrate word-level meaning models (Word2Vec) with ensemble machine learning techniques to more effectively understand sarcastic subtleties in Hinglish tweets.
10	Hilarious or hidden? Detecting sarcasm in hinglish tweets using BERT-GRU. 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)	Aloria, S., Aggarwal, I., Baliyan, N., & Ghosh, M. [10]	ICCCNT 2023 (Conference)	2023	Demonstrated attention-based transformer superiority for humor and irony detection	The sequential context and deep semantics of sarcasm in Hinglish are not fully captured by traditional models, requiring advanced hybrid deep learning architectures like BERT-GRU.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
11	A comparative study of machine learning and deep learning approaches for identifying Assamese abusive comments on social media. Procedia Computer Science, 258, 981–992.	Chutia, T., Baruah, N., & Sonowal, P. [11]	Procedia Computer Science	2025	compared traditional ML and DL; BiLSTM outperformed SVM in contextual text understanding.	There is an absence of strict comparative standards between conventional machine learning methods and deep learning approaches specifically aimed at identifying abusive language in Assamese.
12	Named Entity Recognition in Assamese Language using two separate models: BiLSTM and BERT. Procedia Computer Science, 258, 242–251.	Baruah, P., Dutta, B., Sarma, S. K., & Talukdar, K [12]	Procedia Computer Science	2025	Implemented dual-model NER; contributed insights for low-resource Indian languages.	The scarcity of efficient Named Entity Recognition (NER) solutions for the under-resourced Assamese language has led to the investigation of BiLSTM and BERT functionalities.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
13	Sentiment analysis of Mizo using lexical features in low resource based models. Natural Language Processing Journal, 13(100181), 100181[13]	Lalthangmawii, M., & Singh, T. D. [13]	Natural Language Processing Journal	2025	Proposed lexical sentiment analysis for low-resource Mizo; highlighted linguistic diversity challenges	A significant deficiency of natural language processing tools and sentiment analysis frameworks for the severely low-resource Mizo language exists, necessitating customized models that utilize fundamental lexical characteristics.
14	Bidirectional LSTM-based sentiment analysis for Assamese text. American Journal of Computer Science and Technology, 7(2), 29–37	Talukdar, M., & Sarma, S. [14]	American Journal of Computer Science and Technology	2024	Achieved strong sequential understanding for sentiment prediction using BiLSTM.	The necessity for models that can grasp two-way sequential context (BiLSTM) to enhance the precision of sentiment polarity classification in Assamese language content.

Sl.NO	Title of the Article	Name of the Author	Name of the Journal	Year of publish	Findings	Research Gap
15	CodemixedNLP: An Extensible and Open NLP Toolkit for Code-Mixing. In arXiv [cs.CL].	Jayanthi, S. M., Nerella, K., Chandu, K. R., & Black, A. W. [15]	arXiv (Preprint)	2021	Introduced open-source NLP toolkit for processing code-mixed languages like Hinglish	The lack of a comprehensive, open-source, and adaptable NLP framework specifically created to manage the preprocessing, modeling, and assessment of languages that are mixed with code.
16	Code-Mixed Hinglish Hate Speech Detection Dataset. Kaggle.com	Dhekane, S. [16]	Kaggle (Dataset Repository)	2025	Publicly available dataset enabling research on Hinglish hate-speech classification.	There is a considerable shortage of publicly available, high-quality, and uniform datasets that are essential for training and evaluating Hinglish hate speech detection models.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

The overall workflow involves of three proposed methodologies that is illustrated in Fig. 3.1, Fig.3.2, and Fig3.3 which provides a high-level view of the data preprocessing, feature extraction, model training, and evaluation process. It visually summarizes the pipeline from raw dataset input to performance evaluation across mentioned models. The project workflow begins with the primary input which is raw sentiment data of English in Latin text, Hindi in Devnagiri texts and often consisting of code-mixed sentences Hindi-English (Hinglish) in Latin text. This dataset is unstructured and not immediately suitable for Tokenization and further Word Embeddings. Therefore, the first crucial step is Text Pre-processing that involves URL Removal, Lowercasing, Whitespace removal, Tokenization and Embeddings. This Text Pre-processing is critical because deep learning models such as BERT Transformer Models assign different vectors for the same word with different casings, to solve this we use Lowercasing, URLs which do not provide any context so we remove URLs and space normalization as tokens of extra spaces are also created. After preprocessing, the dataset is divided into training (60%), validation (10%), and testing (30%) sets (same for all methodologies). The processed data is then fed into respective sentiment analysis model, where the embedding techniques vary according to the methodology being implemented.

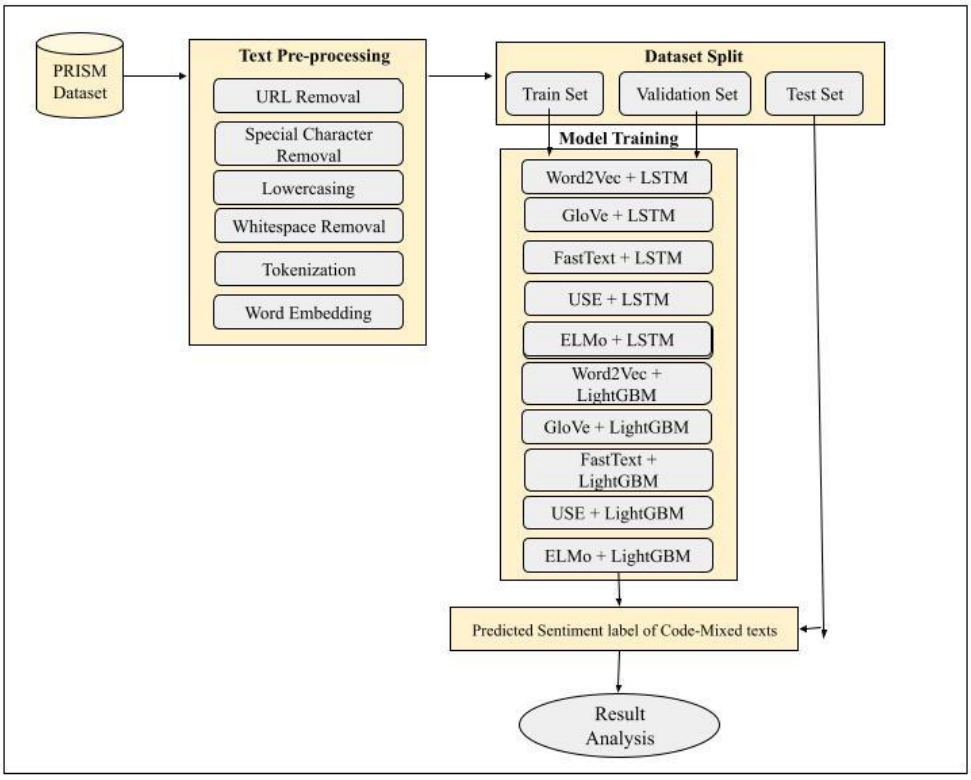


Fig 3.1: Workflow of the proposed methodology for sentiment analysis.

After preprocessing and dataset splitting, the training set is used to develop multiple hybrid sentiment classification models by combining various word embedding techniques with machine learning and deep learning algorithms. Specifically, Word2Vec, GloVe, FastText, Universal Sentence Encoder (USE), and ELMo embeddings are integrated with LSTM and LightGBM classifiers to capture the semantic and contextual information present in code-mixed text. The validation set is used for model tuning and performance optimization, while the test set is employed for final evaluation. The predicted sentiment labels generated by these embedding-classifier combinations are then analyzed to compare their effectiveness and identify the best-performing model for code-mixed sentiment analysis.

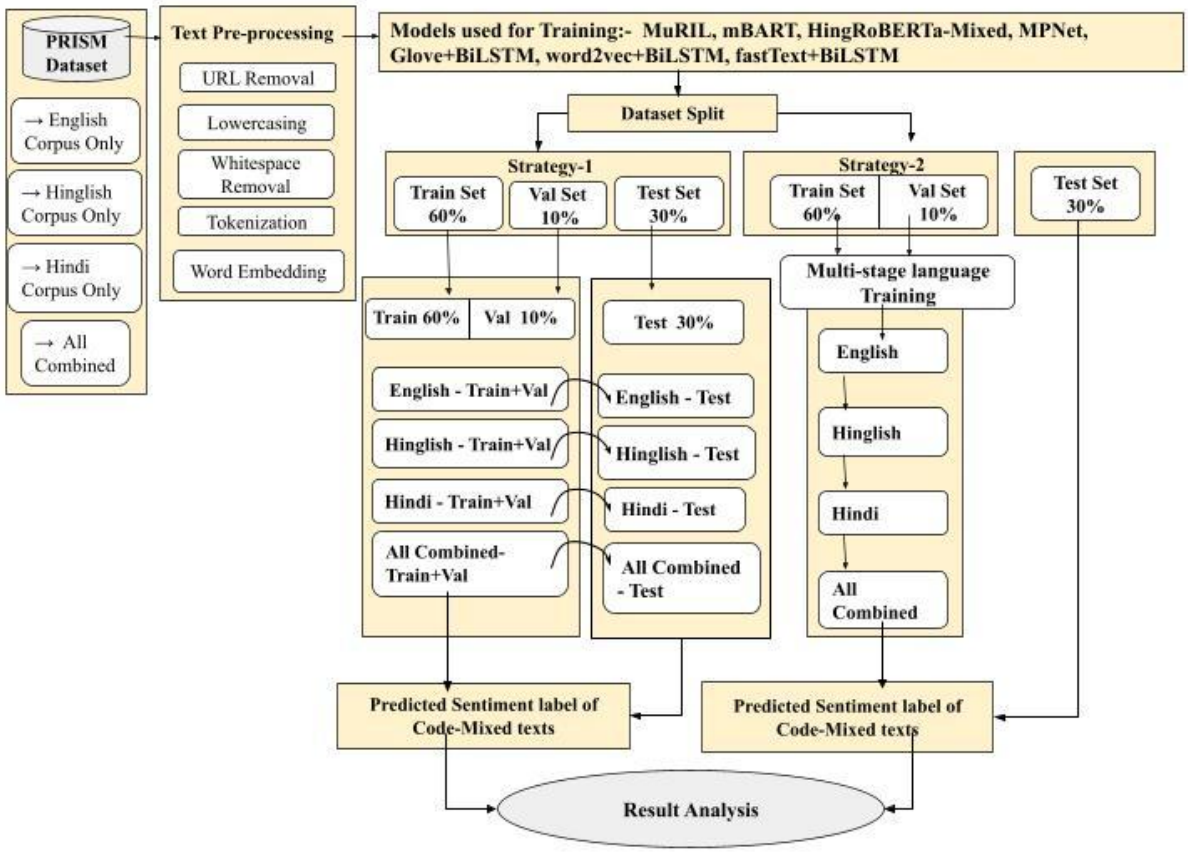


Fig 3.2: Workflow of the proposed methodology for sentiment analysis.

In the above proposed methodology Fig3.2: Two training strategies are applied. In Strategy-1, each language corpus and the combined dataset are independently trained and evaluated on the test set. In Strategy-2, multi-stage language training is performed sequentially using English, Hinglish, Hindi, and combined corpora to improve cross-lingual understanding. Finally, predicted sentiment labels for code-mixed texts are compared through result analysis to evaluate model effectiveness and across multilingual strategies.

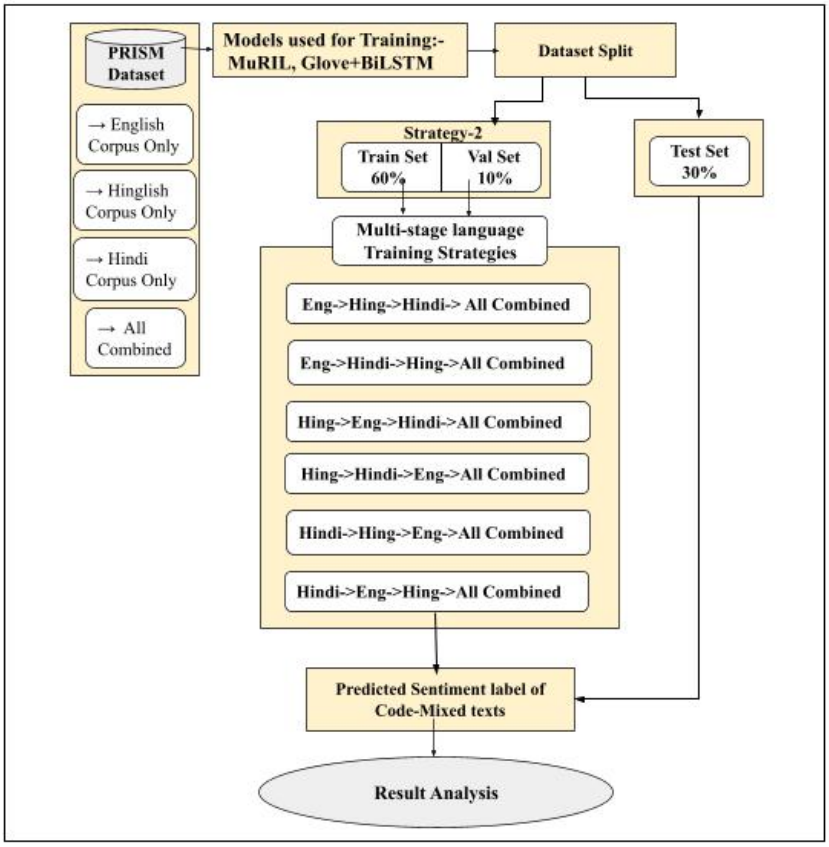


Fig 3.3: Workflow of the proposed methodology for sentiment analysis.

In the proposed methodology fig3, we performed multi-stage language training that consists of six various strategies of training done sequentially using (English, Hinglish, Hindi, all combined), (English, Hindi, Hinglish, all combined), (Hinglish, English, Hindi, all combined), (Hinglish, Hindi, English, all combined), (Hindi, Hinglish, English, all combined), and (Hindi, English, Hinglish, all combined) corpora to improve cross-lingual understanding. Finally, predicted sentiment labels for code-mixed texts are compared through result analysis to evaluate model effectiveness and across multilingual strategies.

3.2 DATA COLLECTION

The research utilized the "combined_hate_speech_dataset" that is publically available on Kaggle. This dataset includes 29,550 labeled text entries, predominantly featuring code-mixed Hinglish sentences. For the purpose of analysis, two main columns were preserved: the text content and the hate_label. The target variable is binary, with 0 indicating non-hate content and 1 signifying hate content. The dataset shows a slight imbalance in class

distribution, with non-hate samples being more prevalent. The following table clearly provides data description.

Table 3.1: Dataset Details

Dataset Name	**combined_hate_speech_dataset (PRISM) – Kaggle**
Total Samples	29,550
Classification Type	Binary (0 = Non-hate, 1 = Hate)
Languages Covered	English, Hindi (Devanagari), Hinglish
Data Composition	English: 15,000
	Hindi: 9,767
	Hinglish: 4,783
Label Distribution	Non-hate: 15,825
	Hate: 13,725
Profanity Lexicon	209 offensive terms with severity scores
Application	Hate-speech / Toxicity detection in multilingual text

3.3 DATA PREPROCESSING

The dataset originally consisted of 29,550 entries, with 9 attributes such as text, hate_label, source, profanity_score, language, dataset_version, combined_date, text_length and word_count. This dataset included samples in English (Latin Script), Hindi(Devnagiri Script), and Hinglish(Latin Script), for the purpose of classifying hate speech. To enhance the quality of the data, thorough preprocessing and noise analysis were conducted prior to tokenization and embedding. During the analysis, it was found that duplicate rows, repeated texts, URLs, mentions, hashtags, elongated words, and social media attachments were significant sources of noise. Regex-based techniques were employed to identify and eliminate these noisy elements. Further preprocessing involved converting text to lowercase, removing extra spaces, normalizing elongated words, and deleting URLs and HTML entities. The following table shows the general noise statistics that are found during preprocessing.

Table 3.2: List Count of URL Noise

URL / Social Noise Type	Count
HTTP/HTTPS Links	356
WWW Links	7
pic.twitter Links	106
YouTube Links	22
Facebook Links	0
Hungama Links	1
Attached URLs	33

The preprocessing stage significantly improved the overall quality of the dataset by removing noisy and redundant textual patterns. Duplicate text entries, URLs, and elongated words were identified as major sources of inconsistency that could negatively affect model learning and classification performance. Cleaning operations such as duplicate removal, URL elimination, and text normalization helped create a more consistent and standardized corpus for embedding. As a result, the dataset size was slightly reduced while preserving meaningful information required for hate speech detection. The following table shows the comparison of the dataset before and after cleaning.

Table 3.3: Data Metrics Before and After Cleaning

Metric	Before Cleaning	After Cleaning
Total Rows	29,550	29,506
Duplicate Texts	11	0
Texts with URLs	458	0
Texts with Elongated Words	2,176	0

After preprocessing, essential features were extracted to prepare the dataset for hate speech classification. The refined dataset retained only relevant attributes required for model training and analysis. The clean_text feature contains normalized textual content,

while hate_label represents the target classification variable. The language feature identifies the language category of each sample, and additional statistical features such as text_length and word_count were included to capture textual characteristics. These extracted features help improve data representation and support effective downstream modeling. The final processed dataset consisted of 29,506 samples with 5 important features.

3.4 EDA

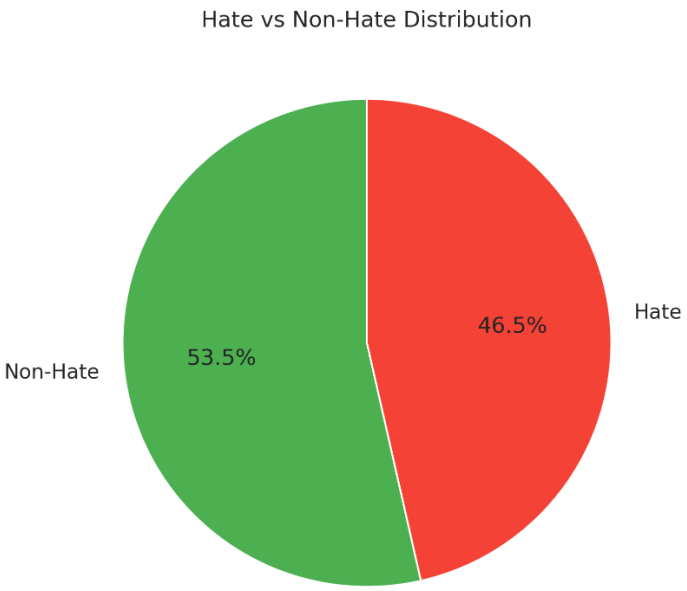


Fig:3.4 Hate vs Non-Hate Class distribution

The following pie chart illustrates the distribution of hate and non-hate samples in the PRISM dataset. The dataset contains approximately balanced class representations, where non-hate samples account for **53.5%** and hate samples account for **46.5%** of the total data. Maintaining a balanced class distribution is important for reducing model bias and improving classification performance.

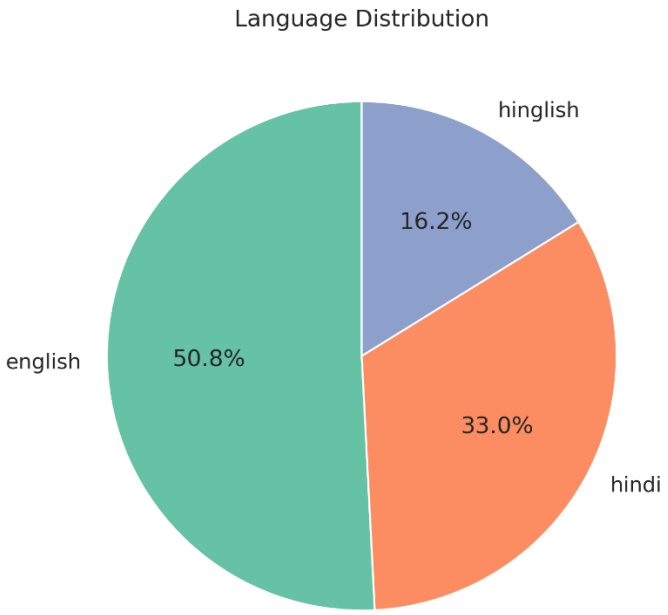


Fig:3.5 Language Distribution

The following pie chart presents the language distribution of the dataset across English, Hindi, and Hinglish corpora. English samples constitute **50.8%**, Hindi samples represent **33.0%**, and Hinglish samples contribute **16.2%** of the dataset. This multilingual distribution enables the models to learn diverse linguistic patterns for multilingual hate speech detection.

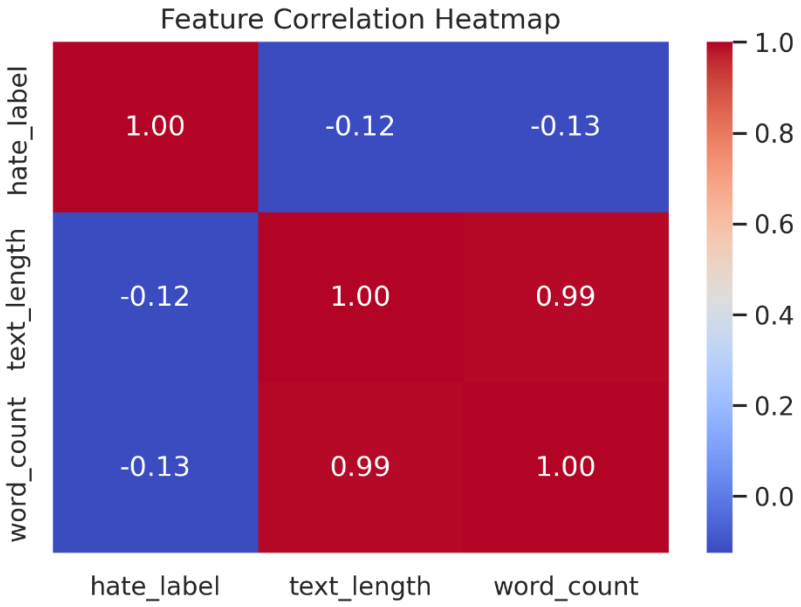


Fig:3.6 Feature Correlation Heatmap

The correlation heatmap illustrates the relationship among numerical features such as hate_label, text_length, and word_count. A very strong positive correlation (0.99) is observed between text length and word count, indicating that longer texts generally contain more words. In contrast, hate_label shows a weak negative correlation with both text length and word count, suggesting that text size has minimal influence on hate speech classification.

```
===== TEXT LENGTH STATISTICS =====
count    29506.000000
mean       150.524436
std        189.746688
min          1.000000
25%         53.000000
50%         94.000000
75%        167.000000
90%        289.000000
95%        480.000000
99%       1063.950000
max       1926.000000
Name: text_length, dtype: float64
```

```
===== WORD COUNT STATISTICS =====
count    29506.000000
mean        28.386260
std         34.559779
min          1.000000
25%         10.000000
50%         18.000000
75%         32.000000
90%         56.000000
95%         90.000000
99%        195.000000
max        300.000000
Name: word_count, dtype: float64
```

Fig:3.7 Text length Statistics and Word count Statistics

The following figures present the statistical distribution of text length and word count in the final cleaned dataset. The text length analysis shows a mean of 150.52 characters and a median of 94 characters, with some samples reaching a maximum length of 1926 characters. Similarly, the word count distribution shows an average of 28.38 words and a median of 18 words, while the maximum word count reaches 300 words. The 95th percentile values of 480 characters and 90 words indicate the presence of long textual samples and high variance within the dataset. These observations are important for determining appropriate padding and truncation limits in transformer-based models to ensure efficient training and balanced sequence representation.

3.5 DATASET SPLITTING

The cleaned PRISM dataset was analyzed to understand the distribution of hate labels and language categories before model training. The dataset contains English, Hindi, and Hinglish samples with a relatively balanced hate speech distribution across languages. To ensure reliable model evaluation, the dataset was divided into training, validation, and testing subsets using a stratified splitting approach. Initially, 70% of the data was reserved as the training pool and 30% as the independent test set. The training pool was further

divided into 60% training data and 10% validation data. Only the clean_text and hate_label columns were used during model training. Language-wise class distributions were maintained across all subsets to preserve dataset balance and reduce sampling bias. The following table shows the language wise data splitting.

Table 3.4: Dataset Splits Table

Category			**Combined**	**English**		**Hindi**		**Hinglish**	
Total Samples			29506	14994		9738		4774	
Label Distribution	Non-hate(0)		15,799	7495		5393		2911	
	Hate(1)		13,707	7499		4345		1863	
Split	**Train (60%)**	Non-hate(0)	18617	4478	9446	3237	6143	1764	3011
		Hate(1)		4485		2622		1117	
	Val (10%)	Non-hate(0)	2086	525	1050	378	683	204	335
		Hate(1)		525		305		131	
	Test (30%)	Non-hate(0)	8865	2297	4499	1597	2926	846	1434
		Hate(1)		2248		1303		561	

3.6 MODEL SELECTION

3.6.1 MODEL DESCRIPTION

3.6.1.1 TRANSFORMER BASED MODELS

1. **MuRIL:** MuRIL is a multilingual transformer model developed by Google for Indian languages and code-mixed text. It generates contextual embeddings that capture semantic relationships across multiple languages. MuRIL was selected because the dataset contains Hindi, English, and Hinglish text, making it highly effective for multilingual hate speech classification.
2. **mBART:** mBART is a multilingual encoder–decoder transformer model designed for cross-lingual understanding and contextual language representation. It was used because of its strong capability to learn multilingual semantic patterns from diverse textual inputs.
3. **HingRoBERTa:** HingRoBERTa is a RoBERTa-based transformer specifically adapted for Hinglish and code-mixed language processing. It was selected because it effectively handles transliterated and mixed-language text commonly found in social media hate speech datasets.
4. **MPNet:** MPNet combines masked language modeling with permuted positional encoding to improve contextual understanding. It was chosen because of its strong sentence representation capability and effectiveness in capturing complex hate speech semantics.

3.6.1.2 DEEP LEARNING MODELS

5. **FastText + BiLSTM:** This model combines FastText embeddings with a Bidirectional LSTM network. FastText captures subword information and spelling variations, while BiLSTM learns contextual dependencies from both forward and backward directions. It was selected because Hinglish and social media text often contain noisy and misspelled words.
6. **Word2Vec + BiLSTM:** Word2Vec provides semantic word embeddings, and BiLSTM captures sequential contextual information from text. This model was used to evaluate the effectiveness of predictive word embeddings for multilingual hate speech detection.
7. **GloVe + BiLSTM:** GloVe embeddings capture global word co-occurrence information, while BiLSTM models sequential text patterns. This architecture was selected to compare static embedding-based contextual learning against transformer models.
8. **Word2Vec + LSTM:** This model combines Word2Vec embeddings with LSTM networks for sequence learning. It was selected because LSTM effectively captures long-term dependencies in textual data.
9. **GloVe + LSTM:** GloVe embeddings with LSTM were used to analyze the effectiveness of global semantic representations in hate speech classification tasks.
10. **FastText + LSTM:** FastText embeddings combined with LSTM were selected

because FastText handles subword-level variations effectively, which is useful for multilingual and code-mixed text.

11. **USE+LSTM:** This model uses Universal Sentence Encoder embeddings with LSTM networks. USE captures sentence-level semantic meaning, making it useful for understanding contextual hate speech patterns.
12. **ELMo+LSTM:** ELMo in conjunction with LSTM integrates advanced contextualized word representations with a sequential deep learning architecture, which enhances its effectiveness in grasping intricate syntactic and semantic frameworks throughout a text.

3.6.1.3 MACHINE LEARNING MODELS

13. **Word2Vec + LightGBM:** This model combines Word2Vec embeddings with the LightGBM classifier. It was selected because LightGBM efficiently handles vectorized textual features using gradient boosting techniques.
14. **GloVe + LightGBM:** GloVe embeddings were used as feature inputs for LightGBM to evaluate how global semantic vectors perform with boosting-based classification.
15. **FastText + LightGBM:** FastText embeddings combined with LightGBM were selected because FastText captures subword semantics effectively while LightGBM provides efficient classification performance.
16. **USE + LightGBM:** This model uses USE sentence embeddings with LightGBM classification. It was selected to evaluate sentence-level semantic representations using boosting methods.
17. **ELMo+ LightGBM:** ELMo combined with LightGBM derives constant contextual feature vectors from text through ELMo and feeds them into a highly efficient, gradient-boosted decision tree classifier, providing a resource-saving method for classification tasks that resemble tabular data.

3.6.2 MODEL ALGORITHMS

The selected models use supervised learning for hate speech classification. Transformer models employ fine-tuning with contextual embeddings and attention mechanisms for sequence understanding. BiLSTM and LSTM architectures process text sequentially to capture contextual dependencies from embeddings. LightGBM models use gradient boosting decision trees for efficient feature-based classification, while Logistic Regression applies a linear decision boundary over Word2Vec embeddings. Cross-Entropy and Binary Cross-Entropy losses were used for binary classification tasks, and optimizers such as Adam, AdamW, and Gradient Boosting strategies were applied for

stable convergence and improved learning performance.

Table 3.5: Model Algorithms

Model	**Core Concept**	**Training Method**	**Loss function**	**Optimizer**
MuRIL	Transformer-based model, contextual multilingual embeddings	Supervised sequence fine-tuning + linear warmup–decay scheduler	Cross-Entropy Loss	AdamW
mBART	Transformer-based encoder–decoder model with fully contextual multilingual embeddings	Supervised sequence fine-tuning with linear warmup–decay scheduler	Cross-Entropy Loss	Adam Optimizer
HingRoBERTa	Transformer-based model (RoBERTa variant), contextual embeddings adapted for Hinglish/code-mixed text	Supervised sequence fine-tuning + linear warmup–decay scheduler	Cross-Entropy Loss	AdamW
MPNet	Transformer-based model using masked language modeling with permuted position encoding for enhanced contextual representation	Supervised sequence fine-tuning + linear warmup–decay scheduler	Cross-Entropy Loss	AdamW
FastText + BiLSTM	Subword (character n-gram) embeddings + Bidirectional LSTM contextual modeling	Supervised mini-batch training with backpropagation	Cross-Entropy Loss	Adam Optimizer

Word2Vec + BiLSTM	Predictive word embeddings (CBOW/Skip-gram) + Bidirectional LSTM contextual modeling	Supervised mini-batch training with backpropagation	Cross-Entropy Loss	Adam Optimizer
GloVe + BiLSTM	Static global co-occurrence embeddings + Bidirectional LSTM contextual modeling	Supervised mini-batch training with backpropagation	Cross-Entropy Loss	Adam Optimizer
Word2Vec + LSTM	Uses Word2Vec embeddings (semantic similarity) + LSTM to capture sequential dependencies in text	Pretrained Word2Vec embeddings fed into LSTM, trained end-to-end on labeled data	Binary Cross-Entropy	Adam
GloVe + LSTM	Uses GloVe embeddings (global word co-occurrence statistics) + LSTM for sequence learning	Pretrained GloVe vectors used as embedding layer, then LSTM training on dataset	Binary Cross-Entropy	Adam
FastText + LSTM	FastText captures subword information (handles misspellings, Hinglish variations) + LSTM	Pretrained FastText embeddings → LSTM trained on sequence data	Binary Cross-Entropy	Adam

USE + LSTM	Universal Sentence Encoder provides sentence-level embeddings + LSTM for deeper sequence modeling	USE embeddings generated → passed to LSTM for classification training	Binary Cross-Entropy	Adam
Word2Vec + LightGBM	Gradient Boosting decision trees + Word2Vec feature vectors	Word2Vec embeddings averaged or pooled → fed into LightGBM classifier	Binary Log Loss	Gradient Boosting Decision Trees
GloVe + LightGBM	GloVe embeddings used as input features + LightGBM for classification	GloVe vectors aggregated → used to train LightGBM model	Binary Log Loss	Stochastic Gradient Descent (+ Negative Sampling)
FastText + LightGBM	FastText embeddings (handles subwords well) + LightGBM classifier	FastText vectors → feature input → LightGBM training	Binary Log Loss	Stochastic Gradient Descent (+ Negative Sampling + Subword learning)
USE + LightGBM	Sentence-level embeddings from USE + LightGBM classification	USE embeddings → directly fed to LightGBM	Binary Log Loss	AdaGrad

ELMo+ LSTM	ELMo gives GBDT	Supervised DL (hybrid)	Binary Cross entropy	SGD
ELMo+ LightGBM	Word2Vec embeddings + Logistic Regression classifier	Supervised ML (linear model)	Binary Log Loss	Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB)

3.6.3 MODEL HYPERPARAMETERS

Hyperparameters were selected to balance computational efficiency and model performance. For Transformer models we set max sequence length = 128 to capture sufficient contextual information while maintaining manageable memory usage. A learning rate of 2e-5 with AdamW optimizer and weight decay of 0.01 was used to ensure stable fine-tuning and prevent overfitting. For BiLSTM and LSTM-based models, embedding dimensions of 100–300 and hidden dimensions up to 256 were chosen to learn rich semantic representations. Dropout values between 0.2–0.5 were applied to reduce overfitting. Batch sizes of 16 and 32 were selected for balanced training stability and GPU utilization. LightGBM hyperparameters such as max_depth, num_leaves, and n_estimators were tuned to improve classification performance while controlling model complexity. Lower learning rates and regularization parameters were used to achieve stable gradient updates and better generalization.

Table 3.6: Model Hyperparameters

Model	**Core Components / Embeddings**	**Key Hyperparameters**	**Optimizer**	**Epochs**	**Batch Size**

MuRIL	Transformer (google/muril-base-cased)	max_seq_len=128, weight_decay=0.01, warmup_ratio=0.1	AdamW, lr=2e-5	8	16
mBART	Transformer (facebook/mbart-large-50)	max_seq_len = 128, weight_decay = 0.01, warmup_ratio = 0.1	AdamW, lr = 2e-5	8	16
HingRoBERTa	Transformer (RoBERTa-based Hinglish-adapted model)	max_seq_len = 128, weight_decay = 0.01, warmup_ratio = 0.1	AdamW, lr = 2e-5	8	16
MPNet	Transformer (microsoft/mpnet-base)	max_seq_len = 128, weight_decay = 0.01, warmup_ratio = 0.1	AdamW, lr = 2e-5	8	16
FastText + BiLSTM	BiLSTM + Pre-trained FastText (subword) embeddings	max_seq_len = 128, embedding_dim = 300, hidden_dim = 256, dropout = 0.5	Adam, lr = 1e-3, weight_decay = 0	10	32
Word2Vec + BiLSTM	BiLSTM + Pre-trained Word2Vec embeddings	max_seq_len = 128, embedding_dim = 300, hidden_dim = 256, dropout = 0.5	Adam, lr = 1e-3, weight_decay = 0	10	32
GloVe + BiLSTM	BiLSTM + Pre-trained GloVe embeddings	max_seq_len = 128, embedding_dim = 300, hidden_dim = 256, dropout = 0.5	Adam, lr = 1e-3, weight_decay = 0	10	32
LSTM	Word2vec	max_seq_len =100 , embedding_dim =100 , hidden_dim = 128, dropout = 0.2	Adam, lr = 1e-2, weight_decay = 0	30	16

LSTM	GLOVE	max_seq_len =100 , embedding_dim =100 , hidden_dim = 128, dropout = 0.2	Adam, lr = 1e-2, weight_decay = 0	30	16
LSTM	FASTTEXT	max_seq_len =100 , embedding_dim =100 , hidden_dim = 128, dropout = 0.2	Adam, lr = 1e-2, weight_decay = 0	30	16
LSTM	USE	max_seq_len =100 , embedding_dim =100 , hidden_dim = 128, dropout = 0.2	Adam, lr = 1e-2, weight_decay = 0	30	16
LightGBM	Word2vec	learning_rate = 0.01 n_estimators = 500 max_depth = 8 num_leaves = 63 subsample = 0.8 colsample_bytree = 0.8 reg_alpha = 0.1 reg_lambda = 0.2	Gradient Boosting Decision Trees	20	16
LightGBM	GLOVE	learning_rate = 0.01 n_estimators = 500 max_depth = 8 num_leaves = 63 subsample = 0.8 colsample_bytree = 0.8 reg_alpha = 0.1 reg_lambda = 0.2	Stochastic Gradient Descent (+ Negative Sampling)	20	16
LightGBM	FASTTEXT	learning_rate = 0.01 n_estimators = 500 max_depth = 8 num_leaves = 63 subsample = 0.8 colsample_bytree = 0.8 reg_alpha = 0.1 reg_lambda = 0.2	Stochastic Gradient Descent (+ Negative Sampling + Subword learning)	20	16

LightGBM	USE	learning_rate = 0.01 n_estimators = 500 max_depth = 8 num_leaves = 63 subsample = 0.8 colsample_bytree = 0.8 reg_alpha = 0.1 reg_lambda = 0.2	AdaGrad	20	16
LightGBM	ELMo	learning_rate = 0.01 n_estimators = 500 max_depth = 8 num_leaves = 63 subsample = 0.8 colsample_bytree = 0.8 reg_alpha = 0.1 reg_lambda = 0.2	Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB)	30	16
LSTM	ELMo	max_seq_len =200 , embedding_dim =200 , hidden_dim = 128, dropout = 0.2	SGD (Stochastic Gradient Descent)., lr = 1e-2, weight_decay = 0	30	16

3.7 EVALUATION METRICS

As the Dataset consists of two classes i.e, 0 (non-hate) and 1 (hate) The performance of the proposed system was rigorously evaluated using 7 well-established metrics that includes Accuracy, Balanced Accuracy, Precision, Recall, Specificity, F1 and AUC-ROC.

Accuracy: Accuracy measures the overall percentage of correctly classified samples among the total predictions. It evaluates how well the model performs on both hate and non-hate classes collectively. Accuracy was used to measure the general classification performance of the models.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Balanced Accuracy: Balanced Accuracy computes the average recall obtained for each class and is useful for handling class imbalance. It ensures that both hate and non-hate classes contribute equally to evaluation. This metric was used to provide unbiased performance measurement across classes.

$$Balanced\ Accuracy = \frac{Recall\ + Specificity}{2}$$

Precision: Precision measures the proportion of correctly predicted hate samples among all samples predicted as hate. It evaluates the model's ability to reduce false positive predictions. Precision was used because false hate predictions can negatively affect classification reliability.

$$Precision = \frac{TP}{TP + FP}$$

Recall (Sensitivity): Recall measures the proportion of actual hate samples correctly identified by the model. It evaluates the model's ability to detect hate speech effectively. Recall was important because missing harmful content may reduce system effectiveness.

$$Recall = \frac{TP}{TP + FN}$$

Specificity: Specificity measures the proportion of correctly identified non-hate samples. It evaluates how effectively the model avoids false hate predictions for normal text. This metric was used to ensure balanced non-hate classification performance.

$$Specificity = \frac{TN}{TN + FP}$$

F1-Score: F1-Score is the harmonic mean of Precision and Recall. It provides a balanced evaluation when both false positives and false negatives are important. F1-score was used because hate speech datasets often require balanced detection capability.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

AUC-ROC: AUC-ROC measures the model's ability to distinguish between hate and non-hate classes across different classification thresholds. Higher AUC values indicate better discrimination capability. This metric was used to evaluate overall classification robustness and threshold-independent performance.

3.8 TOOLS AND FRAMEWORKS

The development and evaluation of the Hindi-English code-mixed involved a combination of tools, libraries, and frameworks from both speech processing and natural language processing domains. The following are the major tools and frameworks utilized throughout the project:

1. Python was used as the primary programming language for dataset preprocessing, model implementation, training, and evaluation.
2. Google Colab was used for executing experiments with GPU support and cloud-based computation.
3. Jupyter Notebook was used for interactive coding, experimentation, and result visualization.
4. Pandas was used for data loading, preprocessing, cleaning, and tabular data manipulation.
5. NumPy was used for numerical computations and array-based operations.
6. Regex was used for detecting and removing URLs, mentions, hashtags, and noisy textual patterns.
7. NLTK was used for tokenization and text preprocessing operations.
8. Scikit-learn was used for dataset splitting, evaluation metrics, and machine learning utilities.
9. PyTorch was used for implementing transformer-based and deep learning models.
10. TensorFlow and Keras were used for implementing LSTM, BiLSTM, and neural network architectures.
11. Transformers was used for loading and fine-tuning transformer models such as MuRIL, mBART, MPNet, and HingRoBERTa.
12. Gensim was used for generating Word2Vec and FastText embeddings.
13. LightGBM was used for machine learning-based classification using boosted decision trees.
14. Universal Sentence Encoder was used for generating sentence-level semantic embeddings.
15. Matplotlib and Seaborn were used for plotting graphs, pie charts, and correlation heatmaps for dataset analysis and visualization.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 RESULTS

4.1.1 REGULAR TRAINING

The performance of the Code-Mixed dataset model was evaluated using Accuracy, Balanced Accuracy, Precision, Recall, Specificity, F1-Score and AUC-ROC across different hybrid models for comparison for the ground truth. The USE+LSTM model initially performed highest on the code-mixed dataset with 68% of accuracy and 59%approx in f1-score. GloVe+LightGBM scored the least in the scale comparison to other hybrid models with 65% accuracy and 57% f1 score.

Table 4.1: Regular Result

Models	accuracy	bal_acc	precision	recall	specificity	f1_score	auc_roc
Word2vec+LSTM	**0.6675**	**0.6573**	**0.6914**	**0.5133**	**0.8012**	**0.5892**	**0.7294**
GloVe+ LSTM	**0.6779**	**0.6627**	**0.7591**	**0.4491**	**0.8763**	**0.5644**	**0.7532**
FastText+LSTM	**0.661**	**0.6496**	**0.6906**	**0.4895**	**0.8097**	**0.5729**	**0.7208**
USE+LSTM	**0.6805**	**0.6684**	**0.7283**	**0.4981**	**0.8388**	**0.5916**	**0.7555**
ELMo+LSTM	**0.6632**	**0.6503**	**0.7084**	**0.4674**	**0.8331**	**0.5632**	**0.7327**
Word2Vec+LightGBM	**0.6665**	**0.6548**	**0.7025**	**0.4893**	**0.8203**	**0.5768**	**0.7397**
GloVe+LightGBM	**0.6527**	**0.6403**	**0.6863**	**0.465**	**0.8156**	**0.5544**	**0.717**
FastText+LightGBM	**0.676**	**0.6643**	**0.7173**	**0.4993**	**0.8293**	**0.5888**	**0.7478**
USE+LightGBM	**0.6739**	**0.6619**	**0.716**	**0.4937**	**0.8302**	**0.5844**	**0.7462**
ELMo+LightGBM	**0.6744**	**0.6636**	**0.6808**	**0.6744**	**0.8156**	**0.6658**	**0.7469**

4.1.2 LANUAGE WISE REGULAR TRAINING

This is a one of a kind strategy technique that involves Language wise regular finetuning and it used the same performance metrics that is being used in Table 4.1. This work consists of various deep learning models with different background architectures and hybrid models. Here MPNet achieved highest accuracy of 74% and highest f1-score with 72% when all languages combined.

Table 4.2: English Strategy

Model Name	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.835074**	**0.835074**	**0.834813**	**0.835556**	**0.834593**	**0.835184**	**0.912011**
MuRIL	**0.814625**	**0.81463**	**0.828996**	**0.792889**	**0.836372**	**0.810541**	**0.896677**
HingRoBERTa	**0.835519**	**0.835526**	**0.858159**	**0.804**	**0.867052**	**0.830197**	**0.917912**
MPNet	**0.822183**	**0.822181**	**0.817426**	**0.829778**	**0.814584**	**0.823555**	**0.899569**
GloVe+BiLSTM	**0.776238**	**0.776579**	**0.756138**	**0.808274**	**0.744885**	**0.781337**	**0.854612**
Word2Vec+BiLSTM	**0.719049**	**0.719050**	**0.722072**	**0.712444**	**0.725656**	**0.717226**	**0.797342**
FastText+BiLSTM	**0.752355**	**0.750867**	**0.740920**	**0.798956**	**0.702778**	**0.768844**	**0.825146**

Table 4.3: Hindi Strategy

Model Name	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.567762**	**0.590282**	**0.510024**	**0.799847**	**0.380717**	**0.622872**	**0.622511**
MuRIL	**0.607803**	**0.608876**	**0.554258**	**0.618865**	**0.598888**	**0.584783**	**0.67028**

HingRoBERTa	**0.605407**	**0.605746**	**0.55254**	**0.608896**	**0.602596**	**0.579351**	**0.631553**
MPNet	**0.599932**	**0.597676**	**0.549306**	**0.576687**	**0.618665**	**0.562664**	**0.611597**
GloVe+BiLSTM	**0.550690**	**0.500000**	**0.000000**	**0.000000**	**1.000000**	**0.000000**	**0.486025**
Word2Vec+BiLSTM	**0.600274**	**0.579456**	**0.578161**	**0.385736**	**0.773177**	**0.462741**	**0.604954**
FastText+BiLSTM	**0.645194**	**0.622978**	**0.612500**	**0.467780**	**0.778175**	**0.530447**	**0.673262**

Table 4.4: Hinglish Strategy

Model	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.706909**	**0.671073**	**0.662005**	**0.50805**	**0.834096**	**0.574899**	**0.732349**
MuRIL	**0.679693**	**0.669716**	**0.583612**	**0.624329**	**0.715103**	**0.603284**	**0.745**
HingRoBERTa	**0.73552**	**0.709035**	**0.688285**	**0.588551**	**0.829519**	**0.634523**	**0.778005**
MPNet	**0.717376**	**0.681911**	**0.679907**	**0.520572**	**0.843249**	**0.589666**	**0.753845**
GloVe+BiLSTM	**0.697939**	**0.638326**	**0.772000**	**0.344029**	**0.932624**	**0.475956**	**0.705756**
Word2Vec+BiLSTM	**0.706211**	**0.662119**	**0.682540**	**0.461538**	**0.862700**	**0.550694**	**0.740799**
FastText+BiLSTM	**0.691358**	**0.619592**	**0.710843**	**0.318919**	**0.920266**	**0.440299**	**0.688354**

Table 4.5: Combined Strategy

Model	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC

mBART	**0.734184**	**0.734024**	**0.706504**	**0.731761**	**0.736287**	**0.718911**	**0.827426**
MuRIL	**0.721532**	**0.721742**	**0.690934**	**0.724708**	**0.718776**	**0.707418**	**0.804714**
HingRoBERTa	**0.748531**	**0.7432**	**0.761364**	**0.668045**	**0.818354**	**0.711658**	**0.83382**
MPNet	**0.74164**	**0.741131**	**0.716694**	**0.733949**	**0.748312**	**0.725219**	**0.82881**
GloVe+BiLSTM	**0.683009**	**0.677217**	**0.681793**	**0.595574**	**0.758861**	**0.635774**	**0.763671**
Word2Vec+BiLSTM	**0.670357**	**0.662825**	**0.676418**	**0.556663**	**0.768987**	**0.610726**	**0.736175**
FastText+BiLSTM	**0.676610**	**0.665650**	**0.710953**	**0.511679**	**0.819620**	**0.595076**	**0.754570**

4.1.3 MULTI-STAGE LANGUAGE TRAINING

This work includes the multi-stage language training with different models by which it means that instead of regular fine tuning this training goes through sequential starts initial from English then Hinglish then Hindi and at last all combined. GloVe+BiLSTM has gained the highest accuracy with 82% and 80% f1-score.

Table 4.6: English Strategy

Model Name	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.829483**	**0.829266**	**0.840185**	**0.809164**	**0.849369**	**0.824383**	**0.897722**
MuRIL	**0.796480**	**0.796017**	**0.820650**	**0.753114**	**0.838920**	**0.785433**	**0.880163**
HingRoBERTa	**0.832563**	**0.832665**	**0.823401**	**0.842082**	**0.823248**	**0.832637**	**0.906317**
MPNet	**0.816502**	**0.816566**	**0.809545**	**0.822509**	**0.810623**	**0.815975**	**0.893683**
GloVe+BiLSTM	**0.6106**	**0.6258**	**0.5532**	**0.8407**	**0.4110**	**0.6673**	**0.6250**

Word2Vec+BiLSTM	**0.587438**	**0.588573**	**0.550975**	**0.60457**	**0.572574**	**0.576531**	**0.636518**

Table 4.7: Hindi Strategy

Model Name	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.600345**	**0.594817**	**0.556962**	**0.540292**	**0.649343**	**0.5485**	**0.645617**
MuRIL	**0.598621**	**0.588589**	**0.561126**	**0.489639**	**0.687539**	**0.522951**	**0.637667**
HingRoBERTa	**0.598966**	**0.595967**	**0.552395**	**0.566385**	**0.625548**	**0.559303**	**0.631097**
MPNet	**0.605172**	**0.602944**	**0.55826**	**0.580967**	**0.624922**	**0.569387**	**0.632868**
GloVe+BiLSTM	**0.5276**	**0.5382**	**0.4939**	**0.6885**	**0.3880**	**0.5752**	**0.5192**
Word2Vec+BiLSTM	**0.624492**	**0.624139**	**0.591543**	**0.619163**	**0.629114**	**0.605038**	**0.673203**
FastText+BiLSTM	**0.585889**	**0.585743**	**0.514705**	**0.584725**	**0.586762**	**0.547486**	**0.612223**

Table 4.8: Hinglish Strategy

Model	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.687278**	**0.660987**	**0.627368**	**0.531194**	**0.79078**	**0.57529**	**0.696596**
MuRIL	**0.697939**	**0.657843**	**0.678947**	**0.459893**	**0.855792**	**0.548353**	**0.691189**
HingRoBERTa	**0.707889**	**0.702147**	**0.623762**	**0.673797**	**0.730496**	**0.647815**	**0.750452**
MPNet	**0.685856**	**0.676619**	**0.601019**	**0.631016**	**0.722222**	**0.615652**	**0.738968**
GloVe+BiLSTM	**0.5198**	**0.5147**	**0.4816**	**0.4431**	**0.5863**	**0.4616**	**0.5225**
Word2Vec+BiLSTM	**0.566539**	**0.536258**	**0.72**	**0.109436**	**0.96308**	**0.189994**	**0.577701**

FastText+BiLSTM	**0.728395**	**0.651575**	**0.884057**	**0.329729**	**0.973421**	**0.480314**	**0.687276**

Table 4.9: Combined Strategy

Model	Accuracy	Bal Acc	Precision	Recall	Specificity	F1 Score	AUC-ROC
mBART	**0.731812**	**0.72878**	**0.722592**	**0.686041**	**0.771519**	**0.703842**	**0.802136**
MuRIL	**0.715996**	**0.710274**	**0.723184**	**0.629621**	**0.790928**	**0.673167**	**0.795080**
HingRoBERTa	**0.736218**	**0.735923**	**0.709502**	**0.731761**	**0.740084**	**0.720460**	**0.820400**
MPNet	**0.726502**	**0.726061**	**0.699929**	**0.719844**	**0.732278**	**0.709747**	**0.810777**
GloVe+BiLSTM	**0.8204**	**0.8186**	**0.8149**	**0.7935**	**0.8437**	**0.8041**	**0.9139**
Word2Vec+BiLSTM	**0.730456**	**0.726676**	**0.72639**	**0.673395**	**0.779958**	**0.698889**	**0.809126**
FastText+BiLSTM	**0.701762**	**0.700688**	**0.676668**	**0.685554**	**0.715822**	**0.681082**	**0.770482**

4.1.4 MULTI-STAGE LANGUAGE TRAINING WITH SIX VARIATIONS ON MuRIL AND GloVe + BiLSTM

This work involves two models MuRIL and Glove+BiLSTM for model training that includes multi-stage language training of six variations using (English, Hinglish, Hindi, all combined), (English, Hindi, Hinglish, all combined), (Hinglish, English, Hindi, all combined), (Hinglish, Hindi, English, all combined), (Hindi, Hinglish, English, all combined), and (Hindi, English, Hinglish, all combined) to check which variation performs well on which model. So, we have achieved the highest 72% accuracy with Hinglish, Hindi, English, all combined variation from MuRIL model and highest accuracy with 66% in Hindi, Hinglish, English, all combined variation using GloVe+BiLSTM model.

Table 4.10: Combined Strategy

Model	Strategy	Phase	Accuracy	Balanced Acc	Precision	Recall	Specificity	F1	ROC-AUC
BiLSTM+GloVe	**E -> HG -> H -> F**	**E**	**0.7498**	**0.7503**	**0.7243**	**0.7980**	**0.7027**	**0.7594**	**0.8190**
		HG	**0.4222**	**0.4862**	**0.3906**	**0.8021**	**0.1702**	**0.5254**	**0.4919**
		H	**0.4759**	**0.5007**	**0.4498**	**0.7460**	**0.2555**	**0.5612**	**0.5025**
		F	**0.6080**	**0.6195**	**0.5554**	**0.7821**	**0.4570**	**0.6496**	**0.7023**

	E -> H -> HG -> F	E	0.7391	0.7392	0.7301	0.7496	0.7288	0.7397	0.8147
		HG	0.4890	0.5081	0.4053	0.6025	0.4137	0.4846	0.4989
		H	0.5731	0.5428	0.5569	0.2441	0.8416	0.3394	0.5863
		F	0.6449	0.6399	0.6305	0.5693	0.7105	0.5983	0.7163
	HG -> E -> H -> F	E	0.7393	0.7392	0.7370	0.7353	0.7431	0.7361	0.8203
		HG	0.5942	0.5202	0.4728	0.1551	0.8853	0.2336	0.5822
		H	0.6079	0.5937	0.5817	0.4536	0.7339	0.5097	0.6320
		F	0.6732	0.6661	0.6770	0.5669	0.7654	0.6171	0.7438
	HG -> H -> E -> F	E	0.6854	0.6855	0.6754	0.7006	0.6704	0.6878	0.7586
		HG	0.6347	0.5581	0.6516	0.1800	0.9362	0.2821	0.5664
		H	0.6372	0.6248	0.6187	0.5019	0.7477	0.5542	0.6598
		F	0.6615	0.6553	0.6574	0.5666	0.7439	0.6087	0.7110
	H -> HG -> E -> F	E	0.5963	0.5965	0.5878	0.6152	0.5777	0.6012	0.6335
		HG	0.6084	0.5362	0.5260	0.1800	0.8924	0.2683	0.5194
		H	0.6331	0.6257	0.5995	0.5526	0.6988	0.5751	0.6660
		F	0.6103	0.6053	0.5884	0.5360	0.6747	0.5610	0.6455
	H -> E -> HG -> F	E	0.7193	0.7188	0.7359	0.6744	0.7632	0.7038	0.8011
		HG	0.5572	0.4940	0.3835	0.1818	0.8061	0.2467	0.5129
		H	0.6272	0.6165	0.6002	0.5104	0.7226	0.5516	0.6606
		F	0.6634	0.6562	0.6648	0.5552	0.7572	0.6051	0.7297
MuRIL	E -> HG -> H -> F	E	0.7965	0.7960	0.8206	0.7531	0.8389	0.7854	0.8802
		HG	0.6979	0.6578	0.6789	0.4599	0.8558	0.5484	0.6912
		H	0.5986	0.5886	0.5611	0.4896	0.6875	0.5230	0.6377
		F	0.7160	0.7103	0.7232	0.6296	0.7909	0.6732	0.7951
	E -> H -> HG -> F	E	0.7985	0.7977	0.8422	0.7291	0.8663	0.7816	0.8645
		HG	0.6645	0.6478	0.5817	0.5651	0.7305	0.5732	0.7036
		H	0.5793	0.5803	0.5285	0.5902	0.5704	0.5577	0.6034
		F	0.7054	0.7025	0.6906	0.6627	0.7424	0.6763	0.7782
	HG -> E -> H -> F	E	0.8090	0.8089	0.8103	0.8016	0.8163	0.8059	0.8318
		HG	0.6660	0.6318	0.6061	0.4635	0.8002	0.5253	0.6622
		H	0.5931	0.5868	0.5494	0.5249	0.6487	0.5369	0.6083
		F	0.7155	0.7124	0.7045	0.6678	0.7570	0.6856	0.7442
	HG -> H -> E -> F	E	0.7985	0.7977	0.8422	0.7291	0.8663	0.7816	0.8779
		HG	0.6830	0.6547	0.6242	0.5152	0.7943	0.5645	0.7030
		H	0.6279	0.6165	0.6029	0.5035	0.7295	0.5487	0.6506
		F	0.7242	0.7179	0.7389	0.6284	0.8074	0.6792	0.7981
	H -> HG -> E -> F	E	0.7901	0.7894	0.8304	0.7233	0.8555	0.7732	0.8516
		HG	0.6802	0.6590	0.6086	0.5544	0.7636	0.5802	0.6945
		H	0.6238	0.6179	0.5849	0.5602	0.6756	0.5723	0.6334
		F	0.7181	0.7135	0.7175	0.6486	0.7785	0.6813	0.7770
	H -> E ->	E	0.7668	0.7654	0.8502	0.6415	0.8894	0.7312	0.8083

	HG -> F	**HG**	**0.6915**	**0.6486**	**0.6749**	**0.4367**	**0.8605**	**0.5303**	**0.6898**
		H	**0.6193**	**0.5971**	**0.6268**	**0.3776**	**0.8165**	**0.4713**	**0.6455**
		F	**0.7065**	**0.6948**	**0.7662**	**0.5299**	**0.8597**	**0.6265**	**0.7508**

4.1.5 SARVAM MODEL *

We have used SARVAM AI Model which is an Indian startup specially made on and made for Indian languages. It is a LLM capable of speech-to-text, and text-to-speech. SARVAM has different models with different parameters but we have implemented Sarvam-1 that consists of 2 Billion parameters. It supports 22+ Indian languages with different scripts. Sarvam can also well handle the code-mixed texts so we implemented our dataset with Sarvam for model training an evaluation as an experimental work with llms and we have achieved highest performance in all 6 metrics.

Table 4.11: Performance Evaluation

Metrics	**Value**
Accuracy	**0.9373**
Balanced Accuracy	**0.9284**
Precision	**0.9486**
Recall	**0.8877**
Specificity	**0.9691**
F1-Score	**0.9171**
ROC-AUC	**0.9326**

4.2 VISUALIZATION

4.2.1 REGULAR TRAINING

This work involves total 39 figures for visualisation that uses two variations of hybrid models one is (Word2Vec, GloVe, FastText, USE and ELMo) with LSTM and another with LightGBM. Fig:4.1 – Fig: 4.10 consists of Train vs Validation Accuracy Curve and Train vs Validation Loss Curve, Fig:4.11 – Fig: 4.20 consists of AUC-ROC Curves, FIG:4.21 – Fig:4.30 consists of Confusion Matrix and Fig:4.31-Fig:4.39 is t-SNE Visualisation.

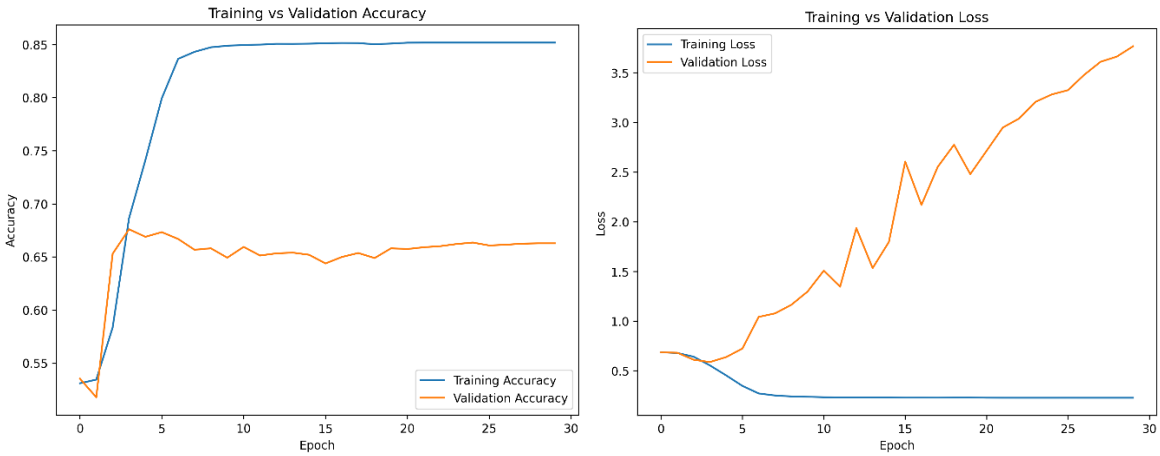


Fig:4.1 Training vs Validation Loss and Accuracy Word2Vec+LSTM

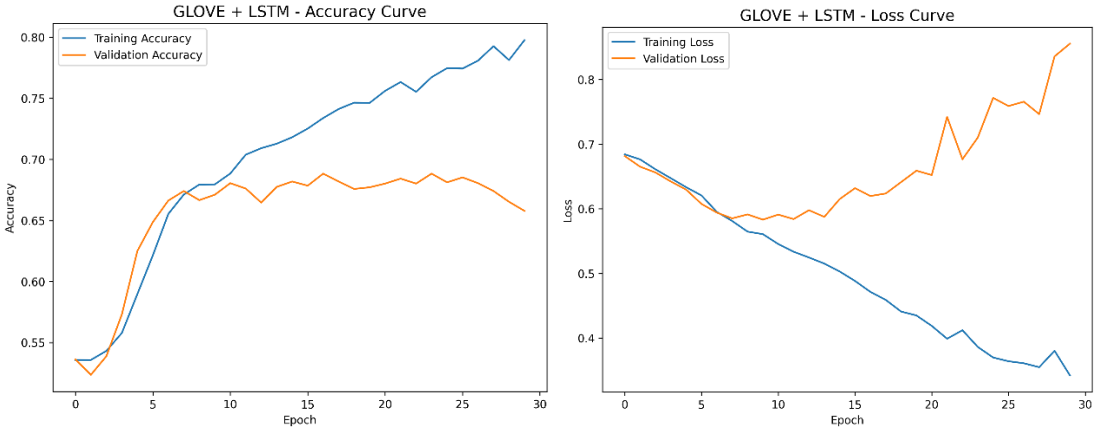


Fig:4.2 Training vs Validation Loss and Accuracy GloVe+LSTM

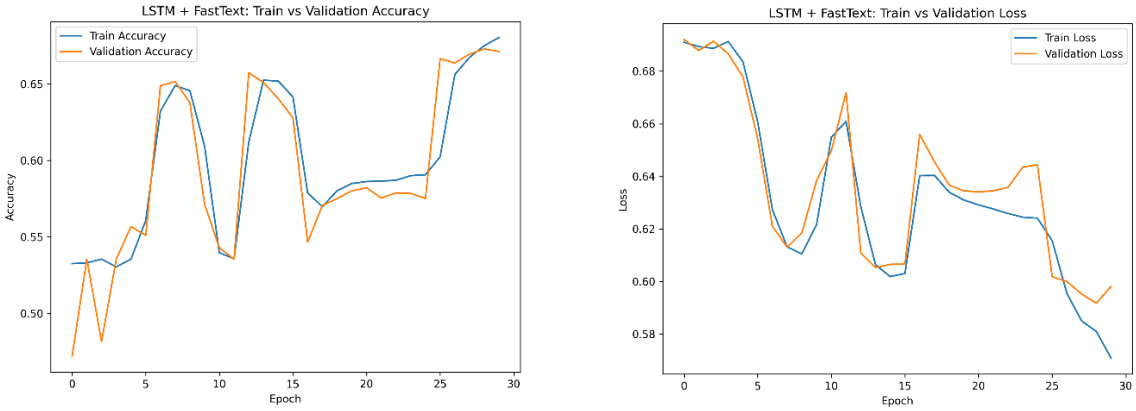


Fig:4.3 Training vs Validation Loss and Accuracy FastText+LSTM

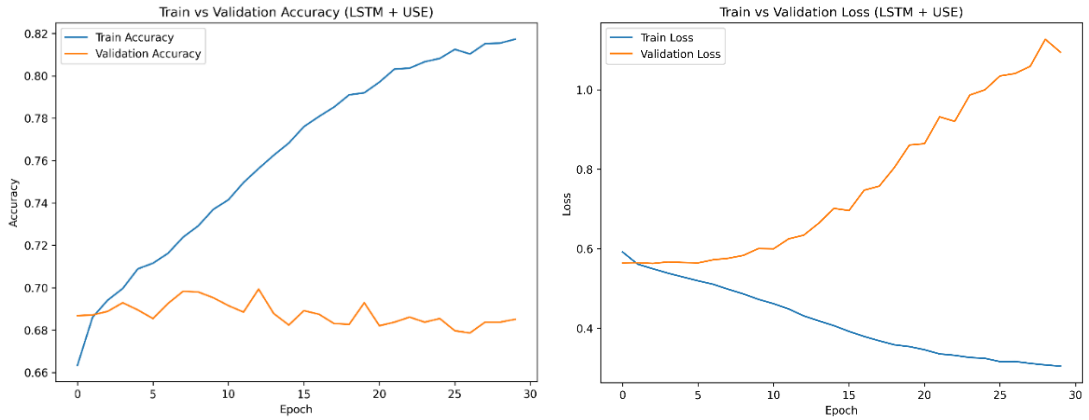


Fig:4.4 Training vs Validation Loss and Accuracy USE+LSTM

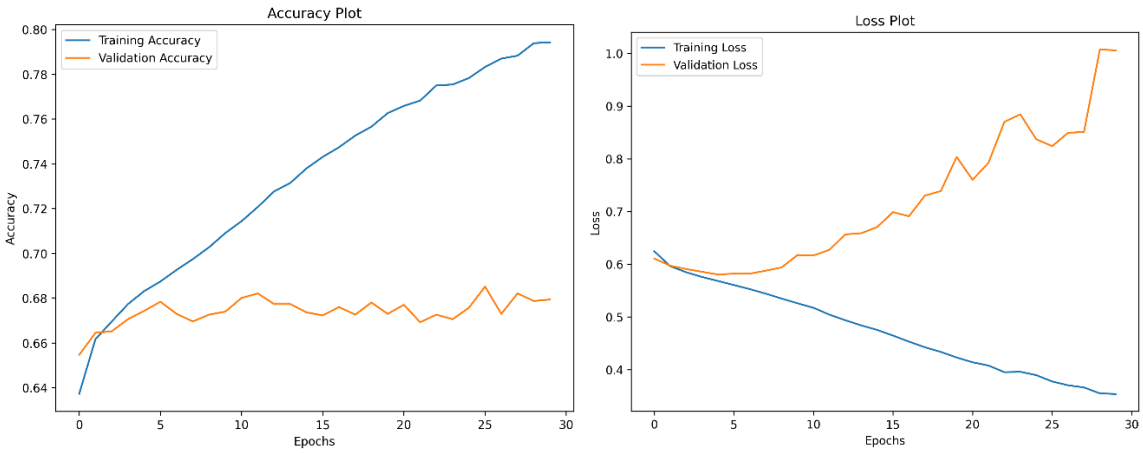


Fig:4.5 Training vs Validation Loss and Accuracy ELMo+LSTM

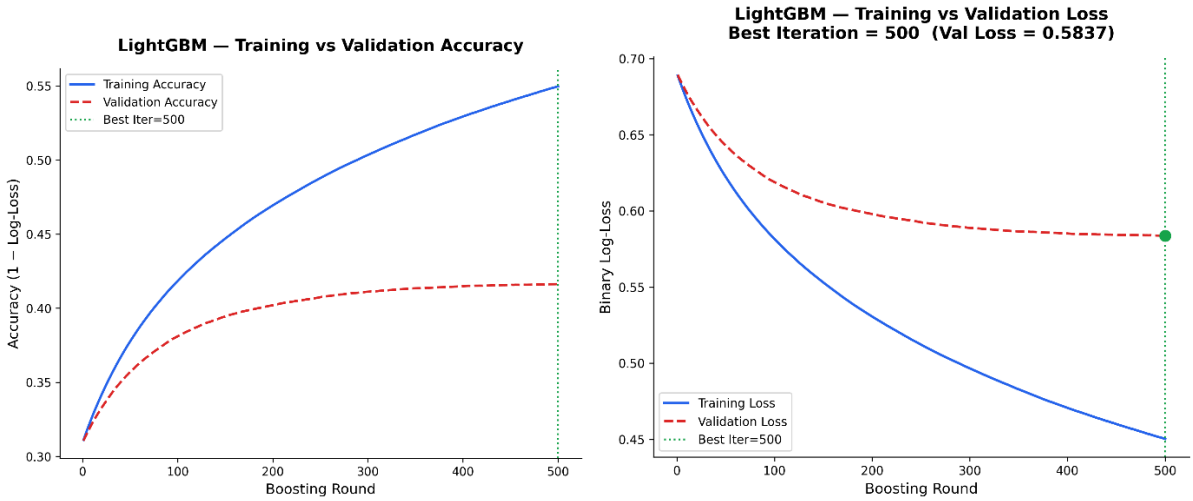


Fig:4.6 Training vs Validation Loss and Accuracy Word2Vec+LightGBM

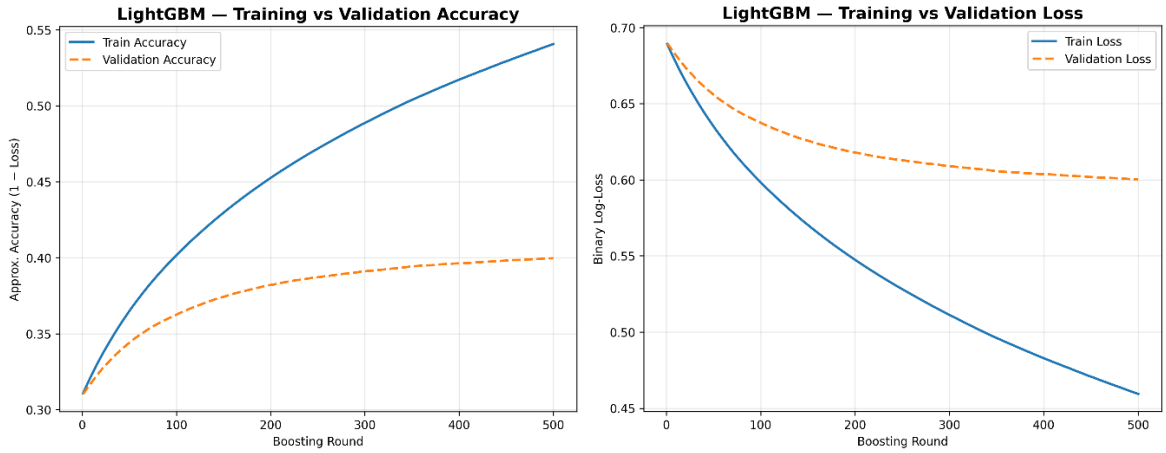


Fig:4.7Training vs Validation Loss and Accuracy GloVe+LightGBM

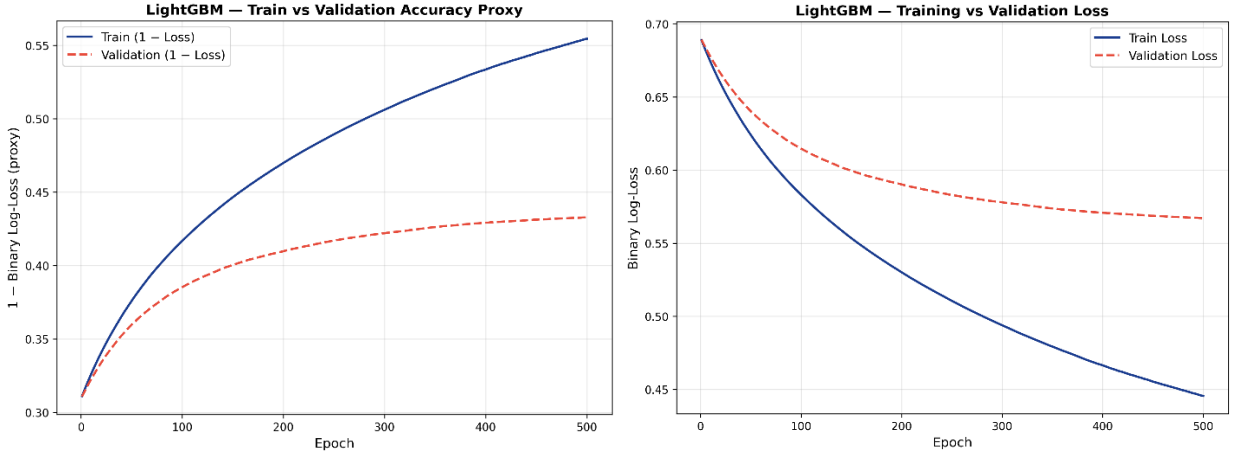


Fig:4.8 Training vs Validation Loss and Accuracy FastText+LightGBM

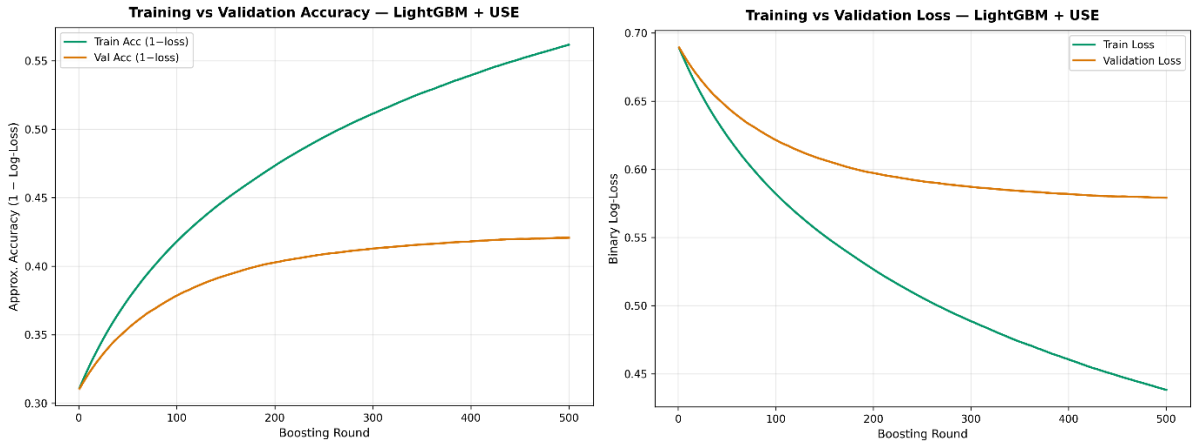


Fig:4.9 Training vs Validation Loss and Accuracy USE+LightGBM

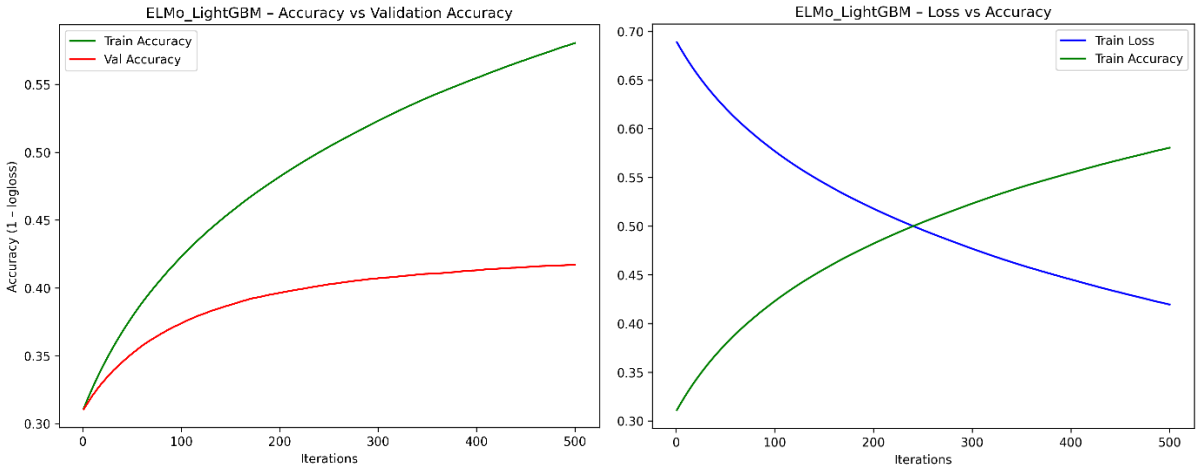


Fig:4.10 Training vs Validation Loss and Accuracy ELMo+LightGBM

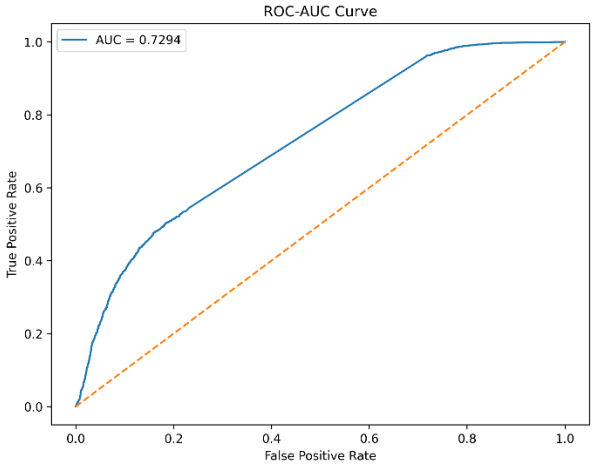


Fig:4.11 ROC-AUC Curve Word2Vec+LSTM

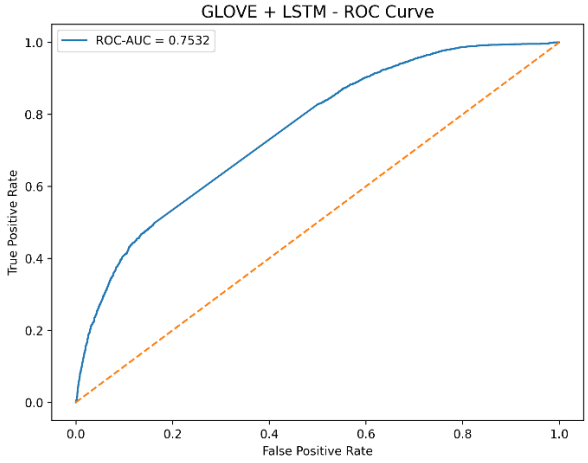


Fig:4.12 ROC-AUC Curve GloVe+LSTM

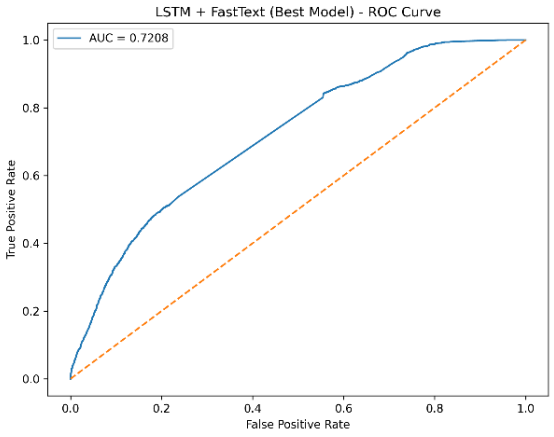


Fig:4.13 ROC-AUC Curve FastText+LSTM

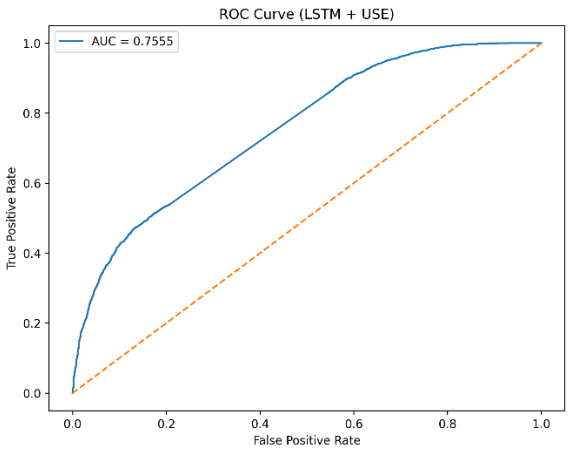


Fig:4.14 ROC-AUC Curve USE+LSTM

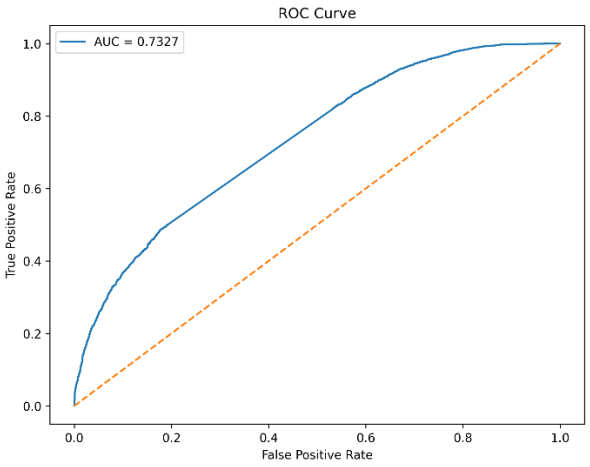


Fig:4.15 ROC-AUC Curve ELMo+~~LSTM~~

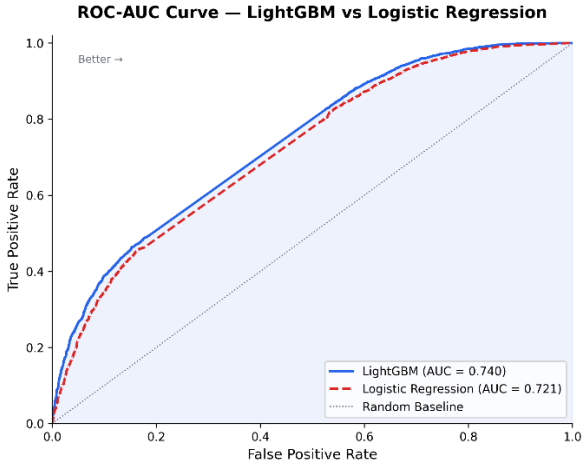


Fig:4.16 ROC-AUC Curve Word2Vec+LightGBM

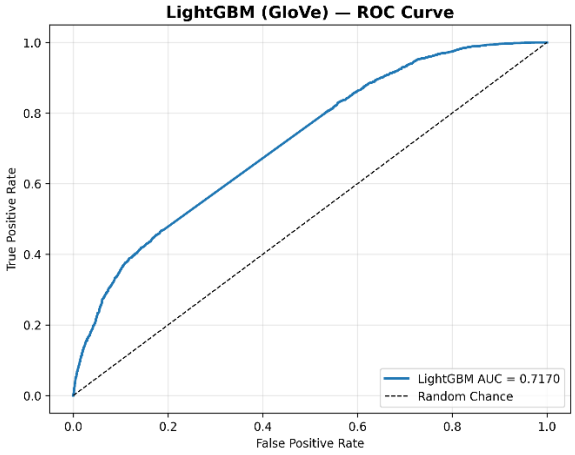


Fig:4.17 ROC-AUC Curve GloVe+LightGBM

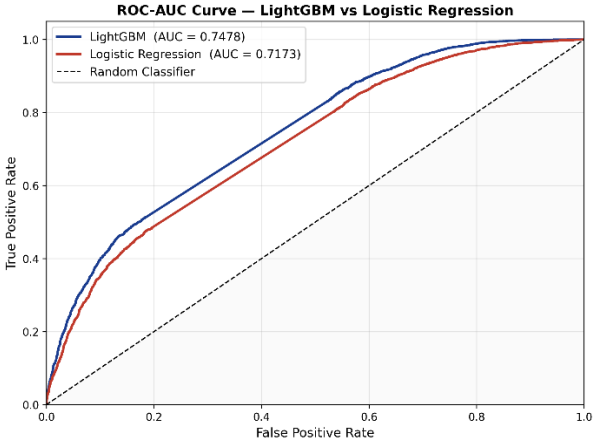


Fig:4.18 ROC-AUC Curve FastText+LightGBM

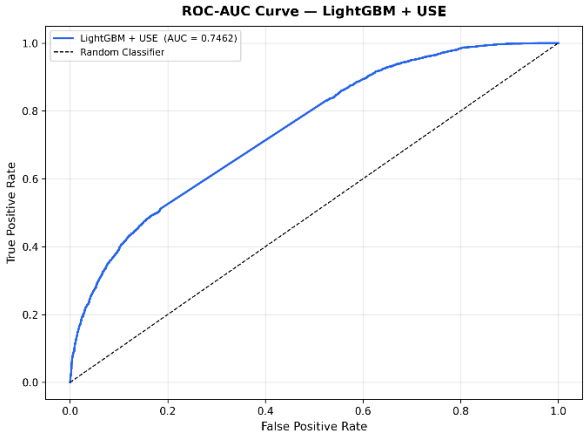


Fig:4.19 ROC-AUC Curve USE+LightGBM

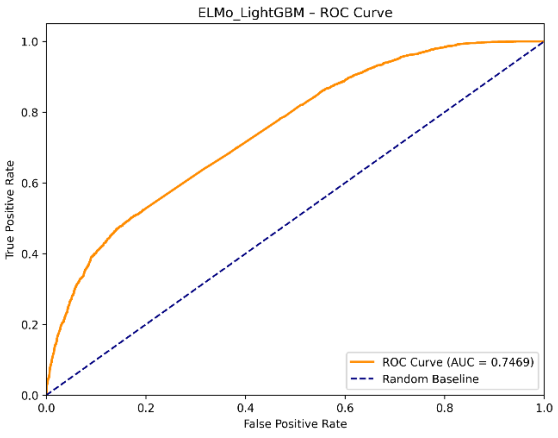


Fig:4.20 ROC-AUC Curve ELMo+LightGBM

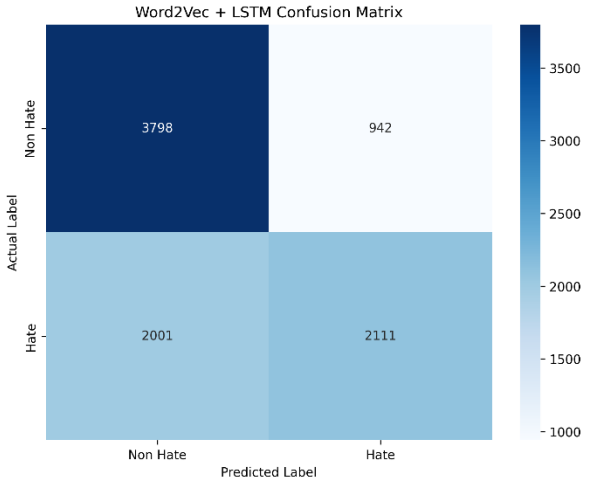


Fig:4.21 Confusion Matrix Word2Vec+LSTM

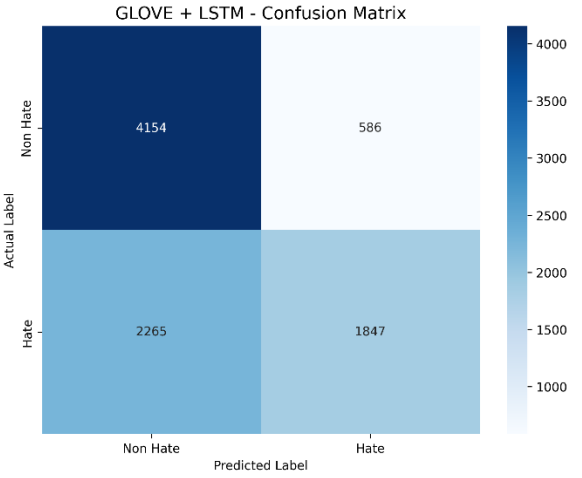


Fig:4.22 Confusion Matrix GloVe+LSTM

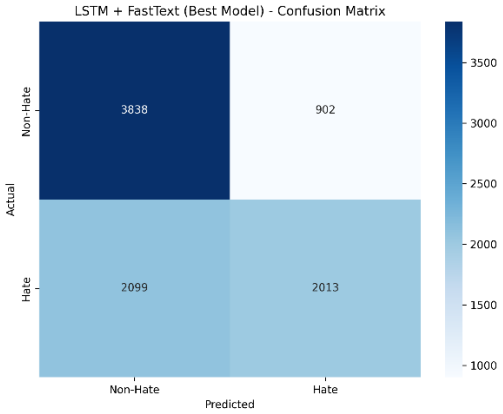


Fig:4.23 Confusion Matrix FastText+LSTM

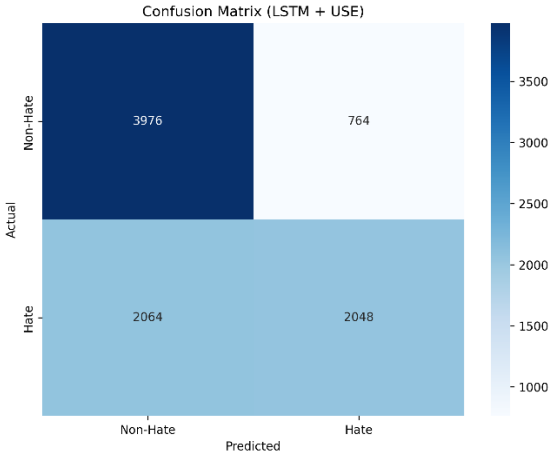


Fig:4.24 Confusion Matrix USE+LSTM

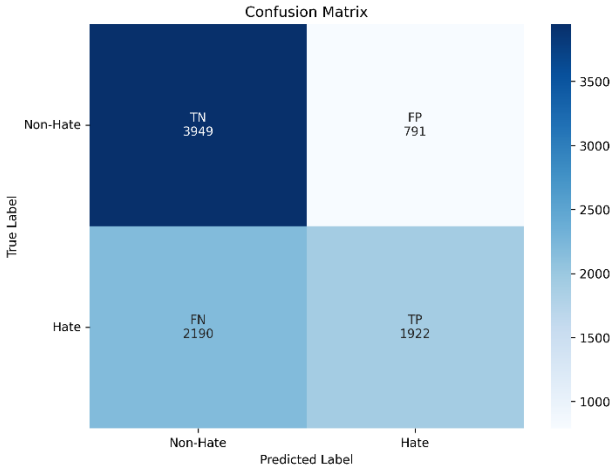


Fig:4.25 Confusion Matrix ELMo+LSTM

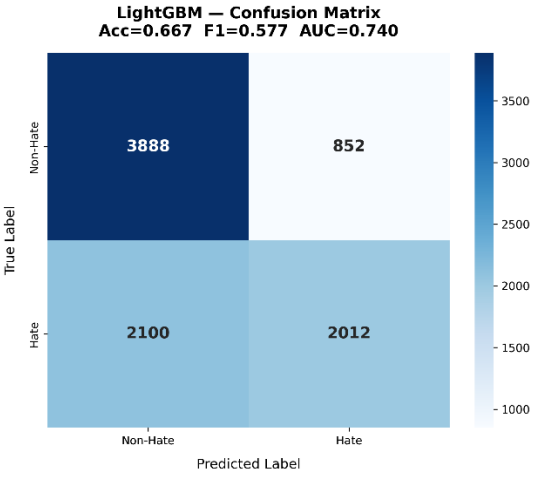
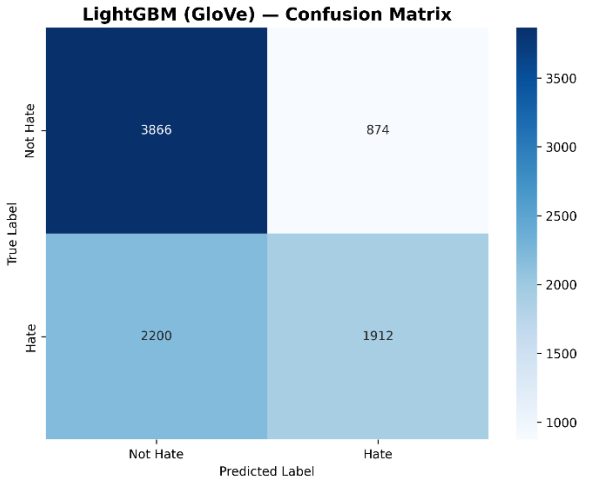


Fig:4.26 Confusion Matrix Word2Vec+LightGBM



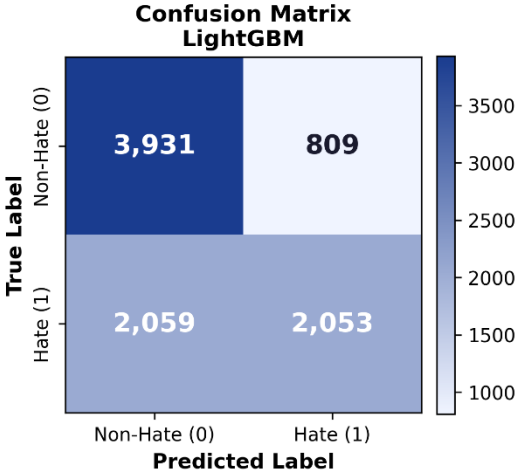
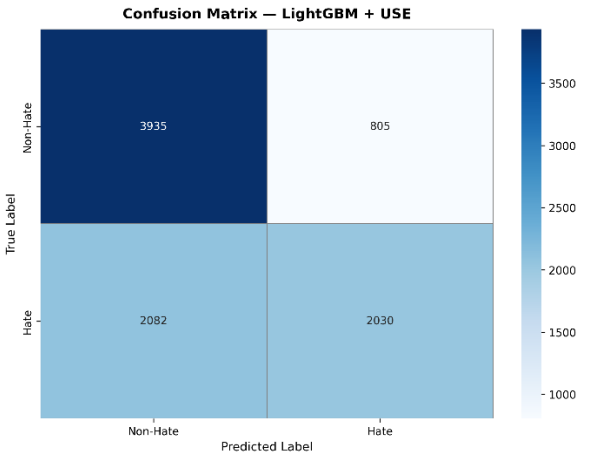


Fig:4.27 Confusion Matrix GloVe+LightGBM Fig:4.28 Confusion Matrix FastText+LightGBM



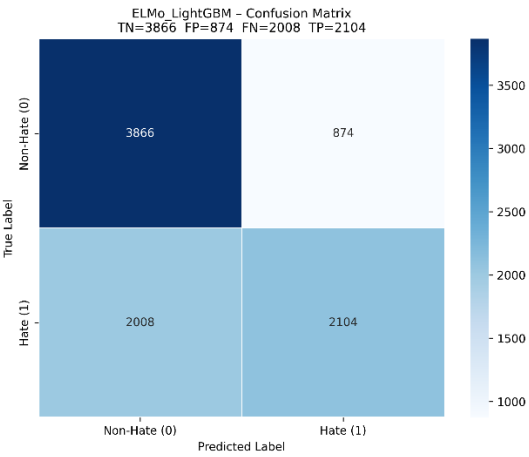
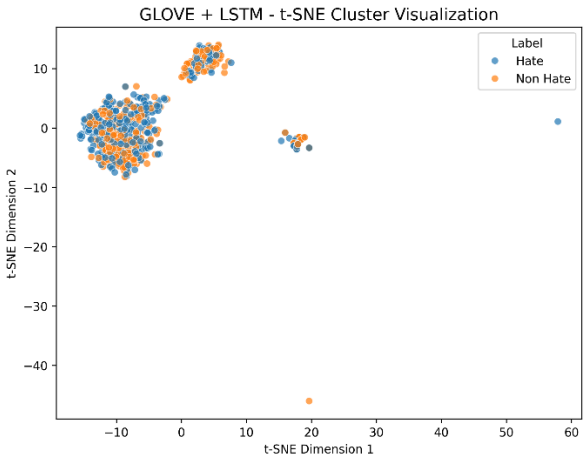


Fig:4.29 Confusion Matrix USE+LightGBM Fig:4.30 Confusion Matrix ELMo+LightGBM



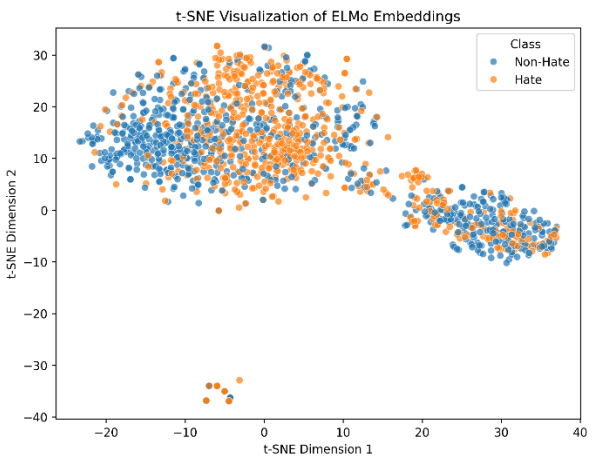


Fig: 4.31 t-SNE for GloVe + LSTM Fig: 4.32 t-SNE for ELMo + LSTM

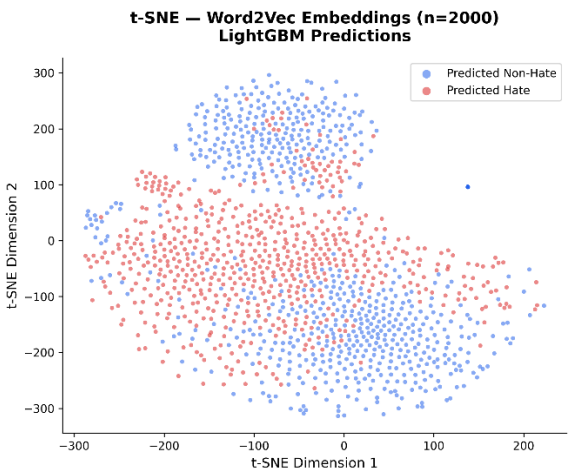


Fig 4.33: t-SNE for Word2Vec+ LightGBM

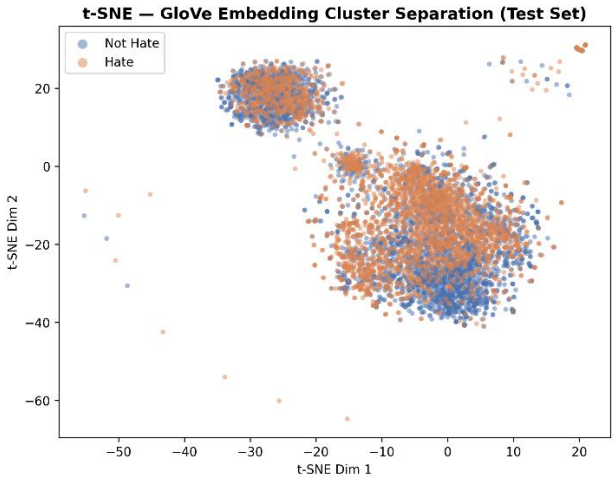


Fig:4.34: t-SNE for GloVe + LightGBM

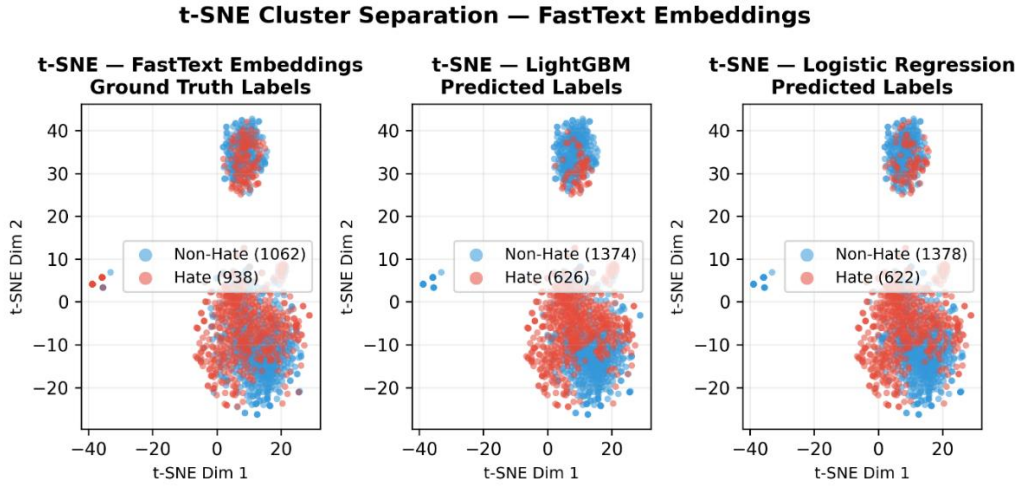


Fig:4.35 t-SNE for Comparison FastText + LSTM

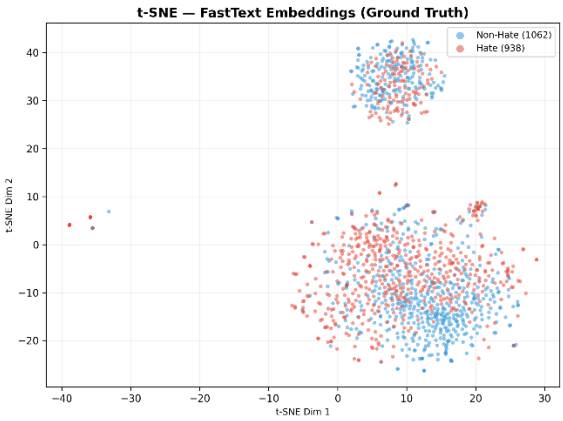


Fig 4.36: t-SNE for FastText + LSTM

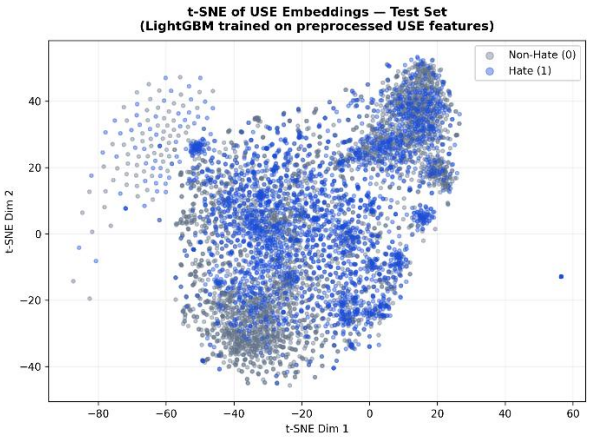


Fig:4.37 t-SNE for Use + LightGBM

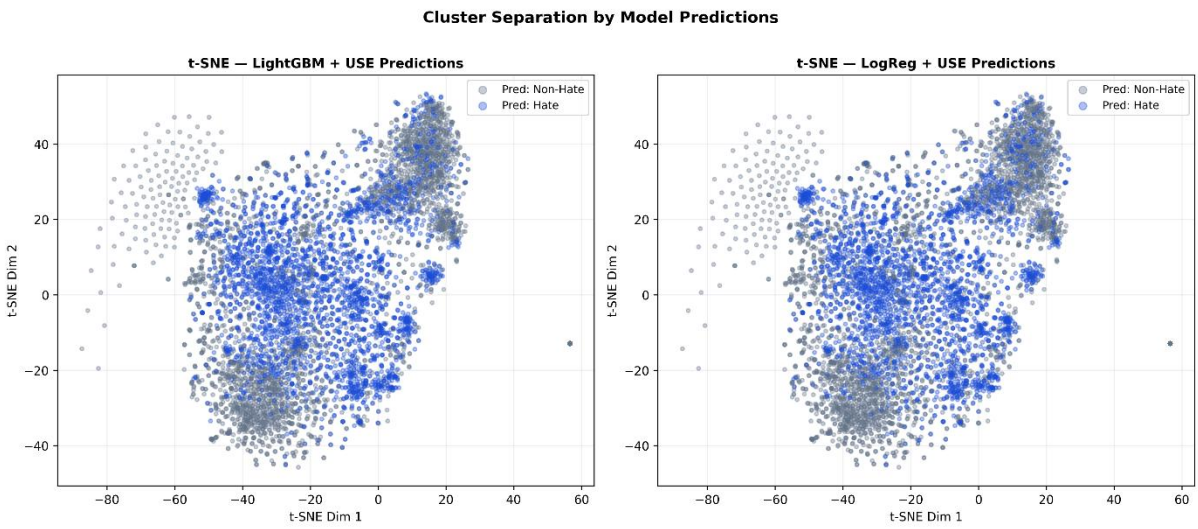


Fig:4.38 t-SNE for cluster comparison USE + LightGBM

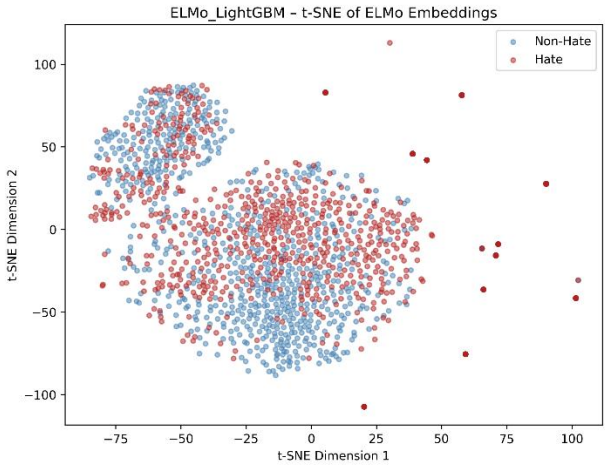


Fig: 4.39 t-SNE for FastText + LSTM

4.2.2 LANGUAGE WISE REGULAR TRAINING

This work involves total 19 figures for visualisation that uses multiplevariations of hybrid models (GloVe+BiLSTM, FastText+BiLSTM, Word2Vec+BiLSTM) and deep learning models(Mbart, MuRIL, HingRoBERTa-mixed, MPNet).but we have used only all combined(Full dataset) visualisations Fig:4.40 – Fig: 4.46 consists of Train vs Validation Accuracy Curve and Train vs Validation Loss Curve, Fig:4.47 – Fig: 4.53 consists of AUC-ROC Curves, FIG:4.54 – Fig:4.59 consists of Confusion Matrix.

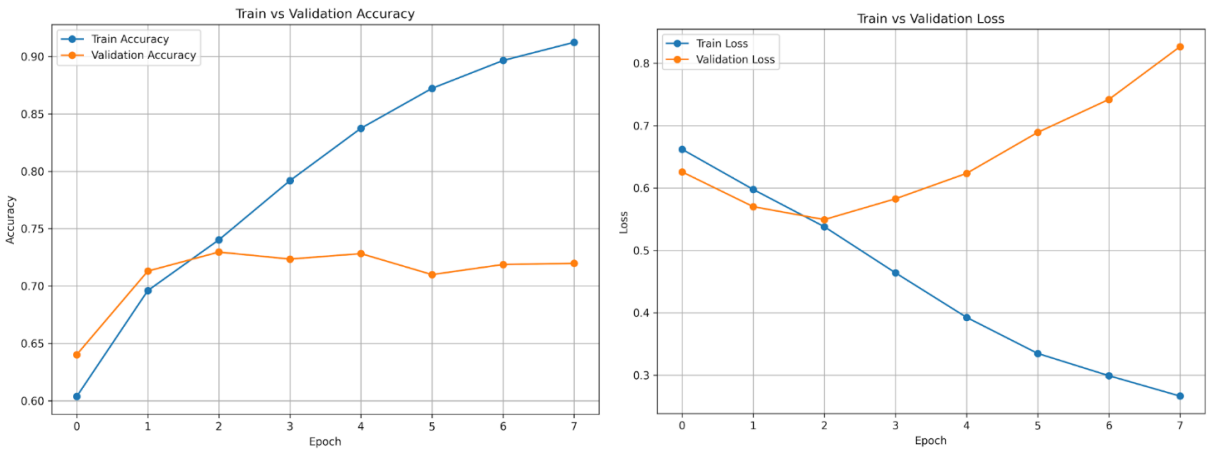


Fig:4.40 Training vs Validation Loss and Accuracy MuRIL Full dataset

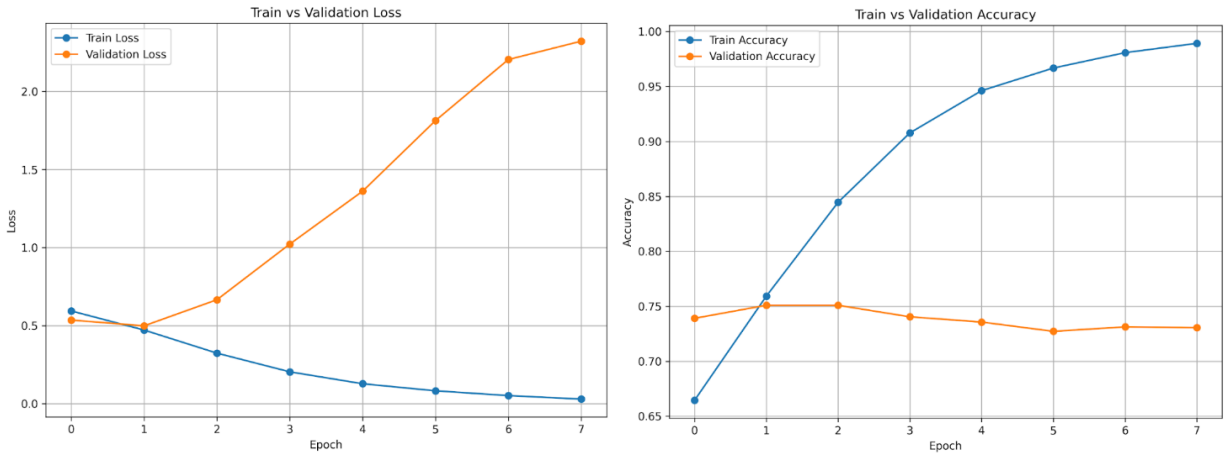


Fig:4.41 Training vs Validation Loss and Accuracy mBART Full dataset

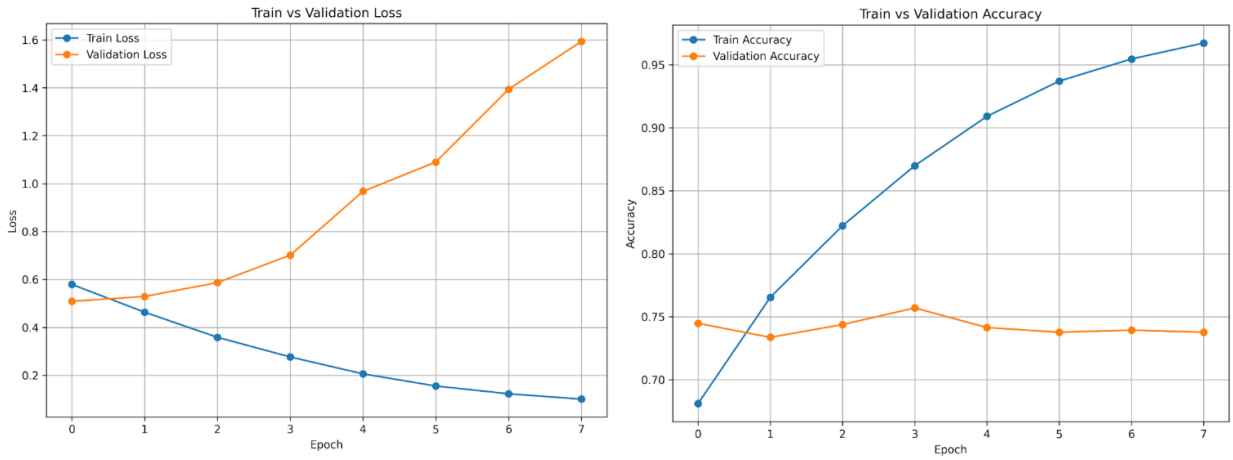


Fig:4.42 Training vs Validation Loss and Accuracy HingRoBERTa Full dataset

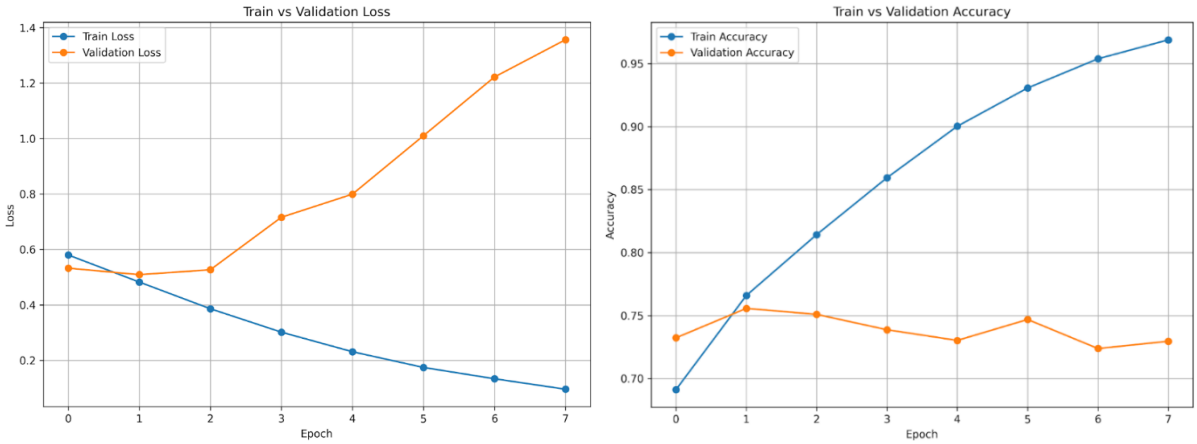


Fig:4.43 Training vs Validation Loss and Accuracy MPNet Full dataset

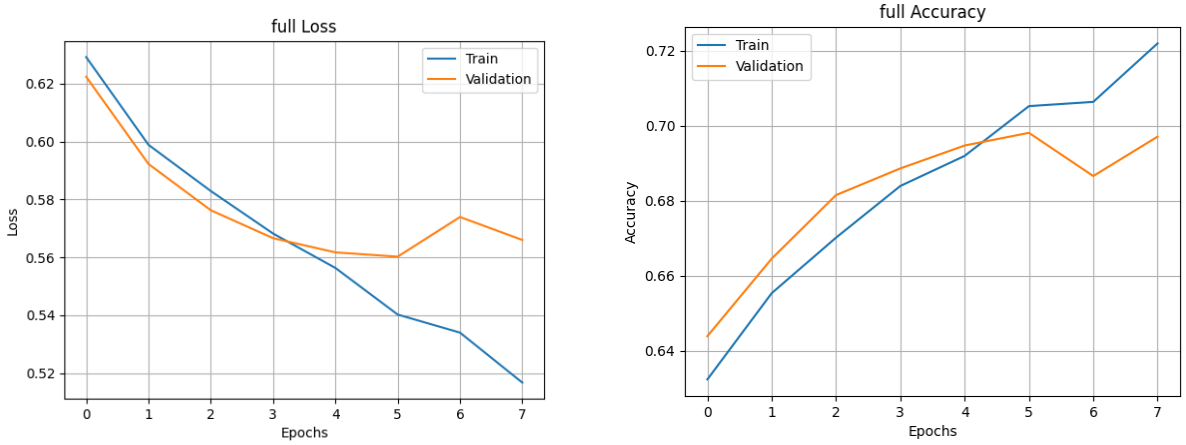


Fig:4.44 Training vs Validation Loss and Accuracy GloVe+BiLSTM Full dataset

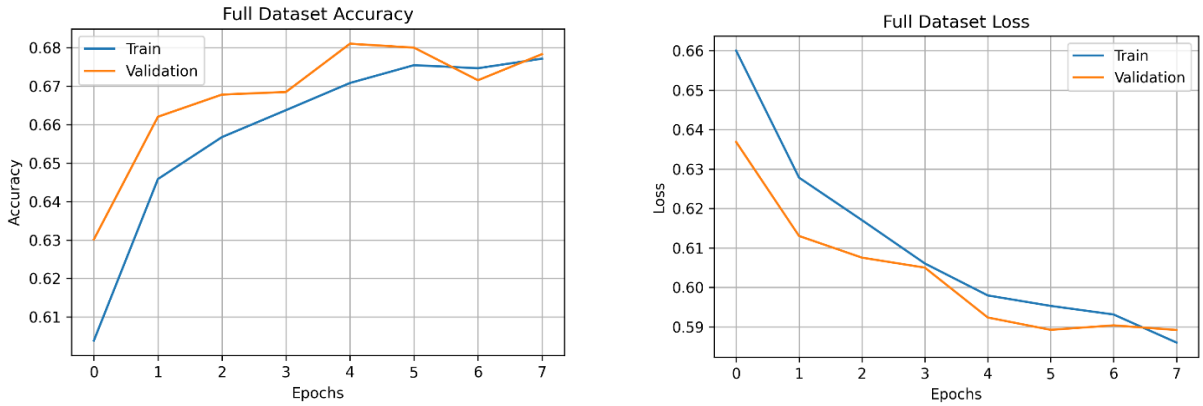


Fig:4.45 Training vs Validation Loss and Accuracy Word2Vec+BiLSTM Full dataset

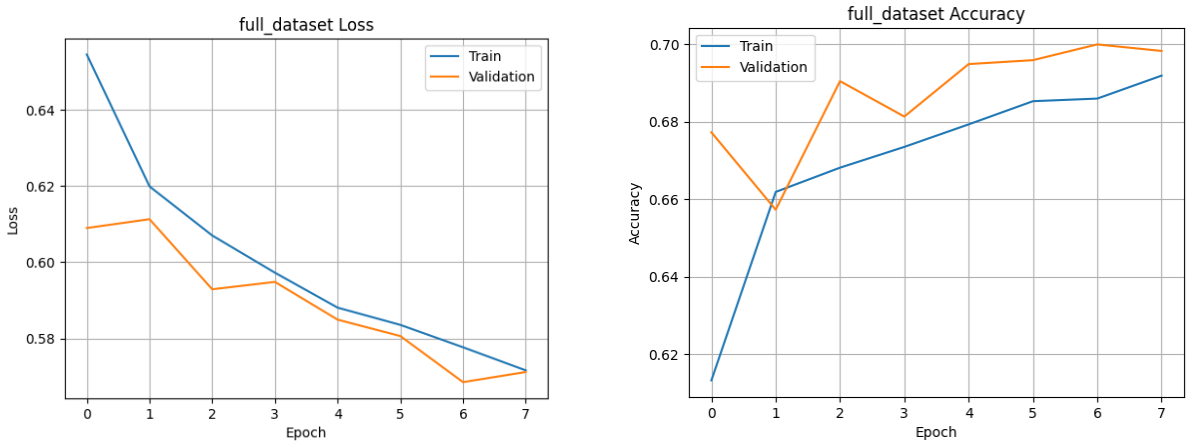


Fig:4.46 Training vs Validation Loss and Accuracy FastText +BiLSTM Full dataset

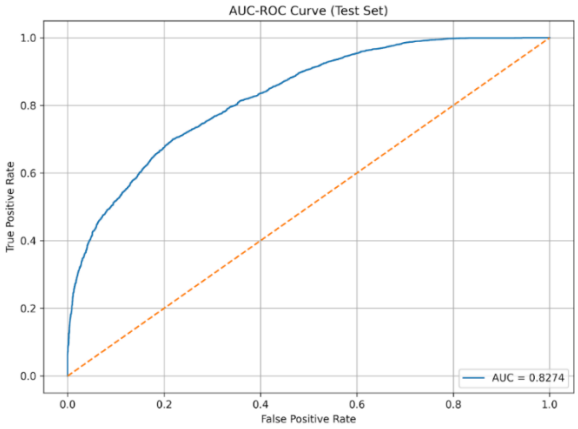


Fig:4.47 ROC-AUC Curve MuRIL

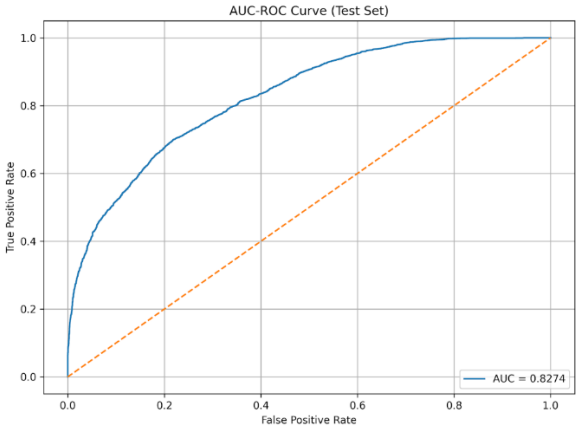


Fig:4.48 ROC-AUC Curve mBART

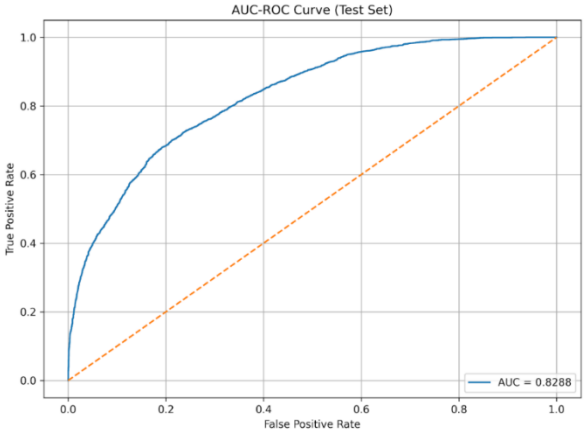


Fig 4.49 ROC-AUC Curve HingRoBERTa

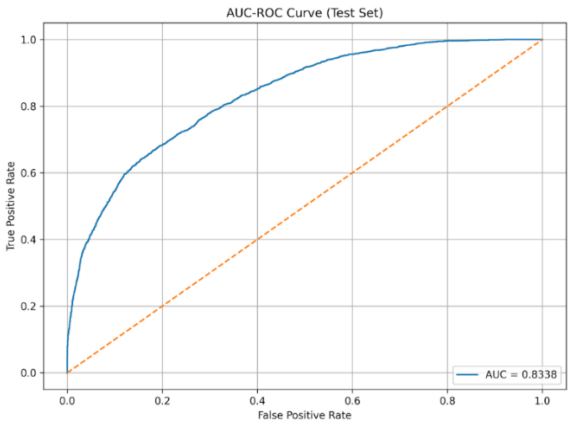


Fig: 4.50 ROC-AUC Curve MPNet

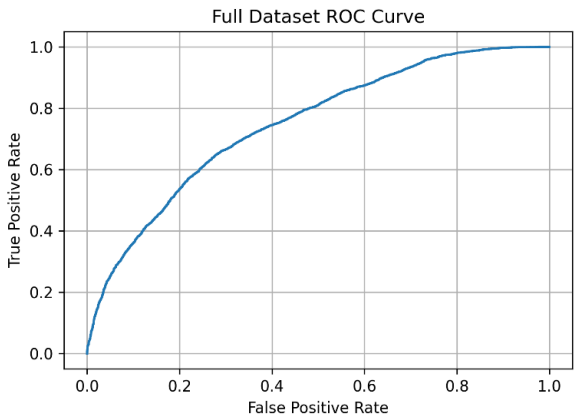


Fig:4.51 ROC-AUC Curve GloVe+BiLSTM

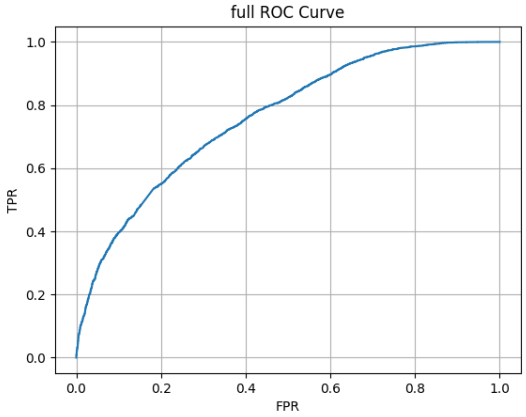


Fig:4.52 ROC-AUC Curve Word2Vec+BiLSTM

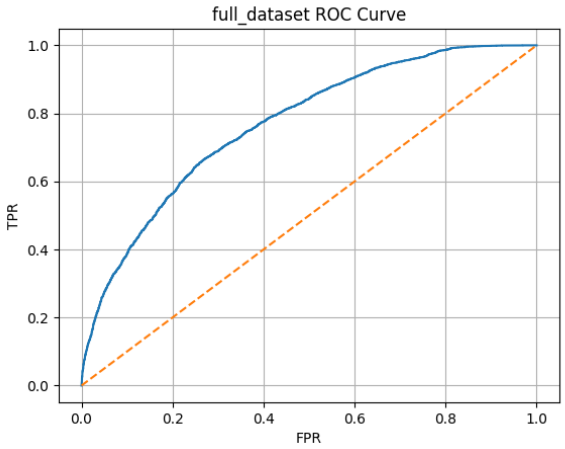


Fig:4.53 ROC-AUC Curve FastText +BiLSTM

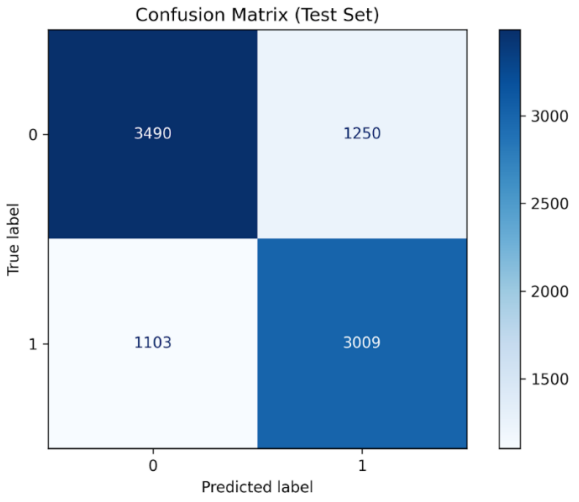


Fig:4.54 Confusion Matrix MuRIL

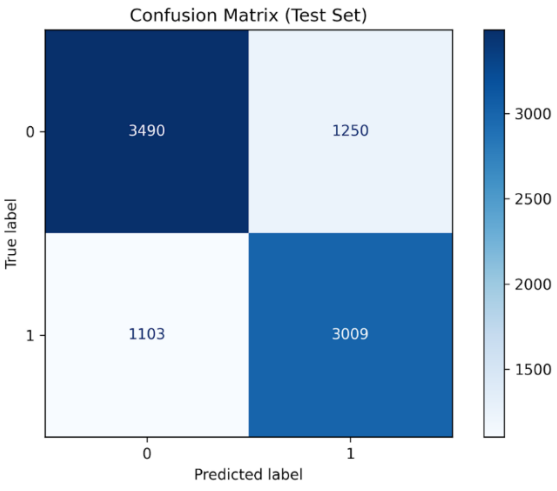


Fig:4.55 Confusion Matrix mBART

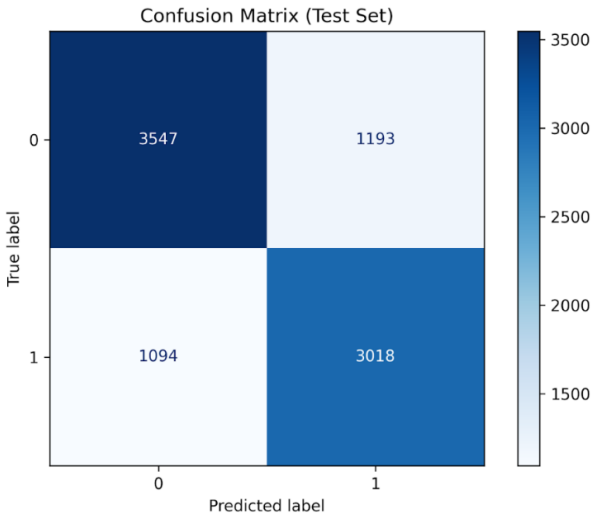


Fig 4.56: Confusion Matrix HingRoBERTa

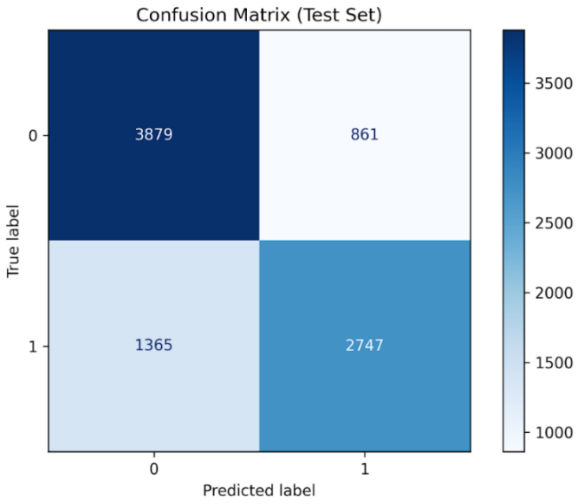


Fig:4.57 Confusion Matrix MPNet

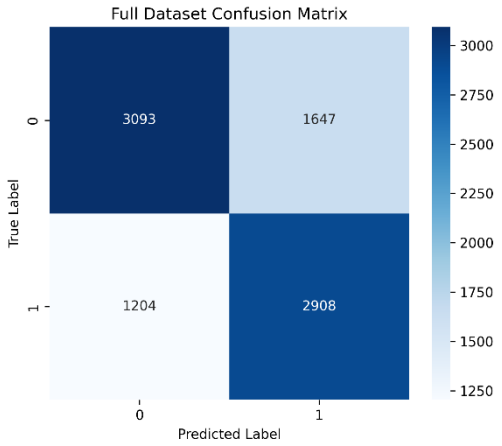


Fig:4.58 Confusion Matrix GloVe + BiLSTM

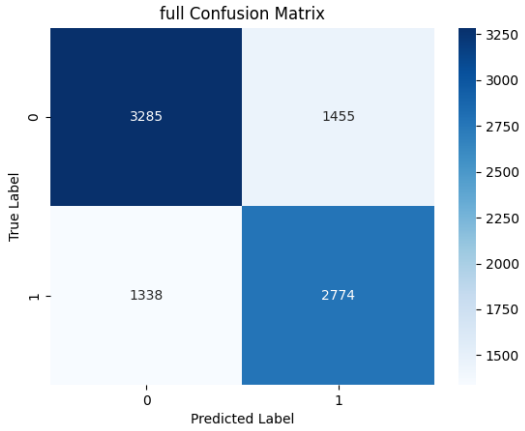


Fig:4.58 Confusion Matrix Word2Vec + BiLSTM

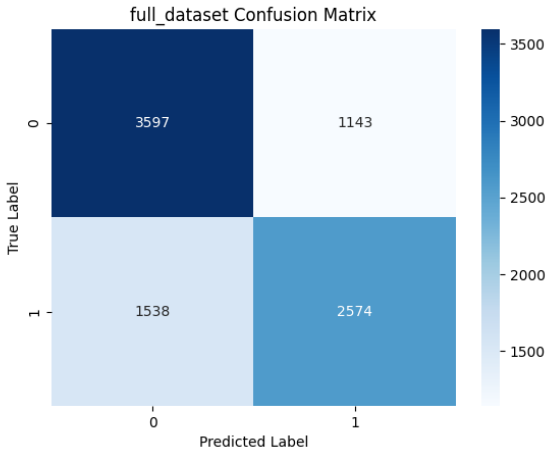


Fig:4.59 Confusion Matrix FastText + BiLSTM

4.2.3 MULTI-STAGE LANGUAGE TRAINING

This work involves total 13 figures for visualisation that uses multiple variations of hybrid models (GloVe+BiLSTM, FastText+BiLSTM, Word2Vec+BiLSTM) and deep learning models (Mbart, MuRIL, HingRoBERTa-mixed, MPNet). For multistage language training on English-> Hinglish -> Hindi -> All combined, and we have used only all combined(Full dataset) visualisations Fig:4.60 – Fig: 4.66 consists of Train vs Validation Accuracy Curve and Train vs Validation Loss Curve, Fig:4.67 – Fig: 4.73 consists of AUC-ROC Curves and Fig:4.74- Fig:4.80 consists of Confusion Matrix.

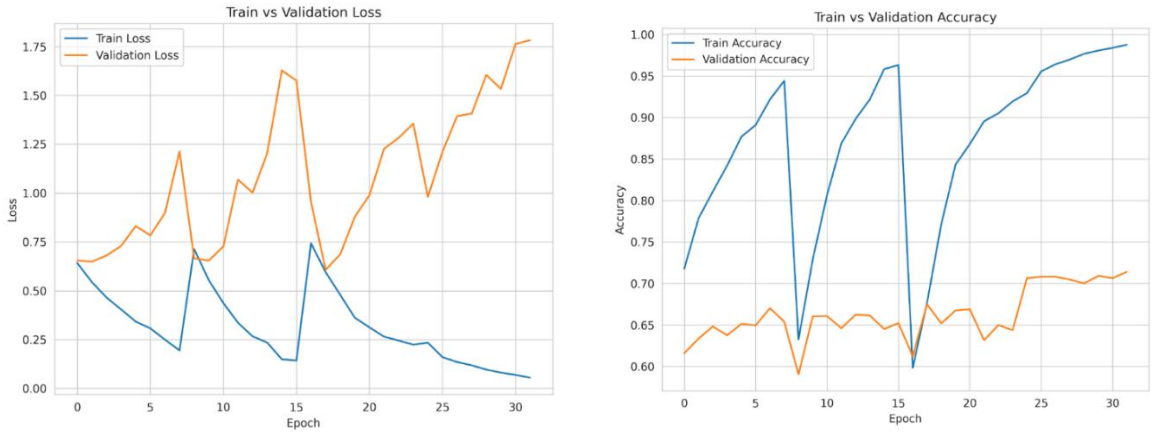


Fig:4.60 Training vs Validation Loss and Accuracy MuRIL Full dataset

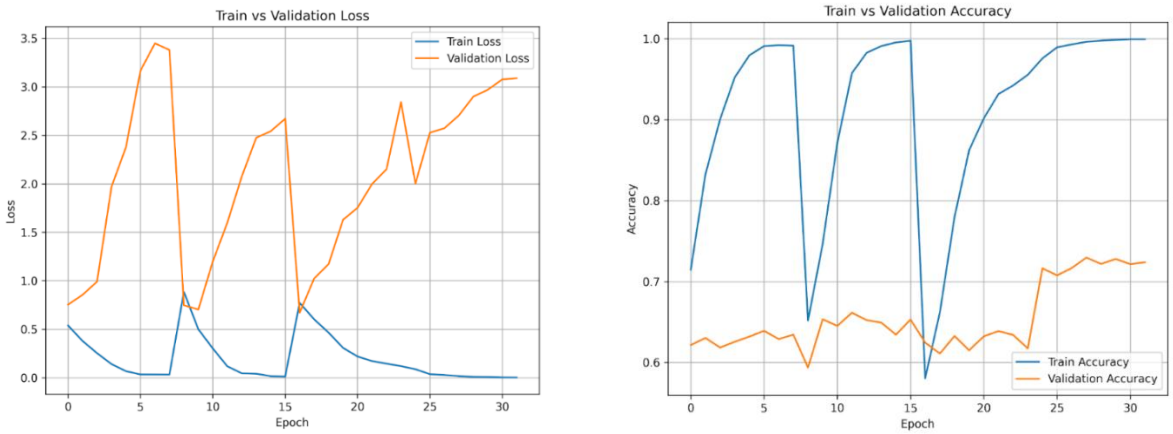


Fig:4.61 Training vs Validation Loss and Accuracy mBART Full dataset

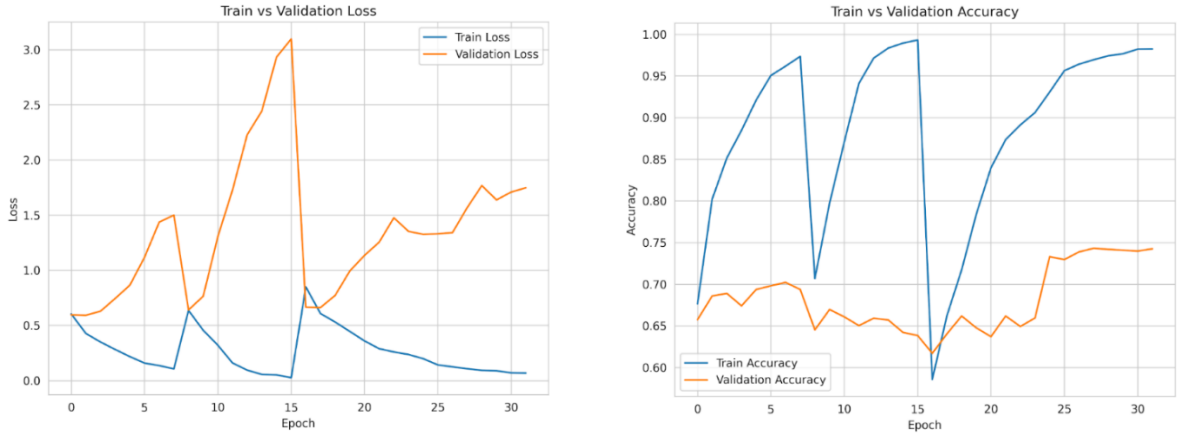


Fig:4.62 Training vs Validation Loss and Accuracy HingRoBERTa Full dataset

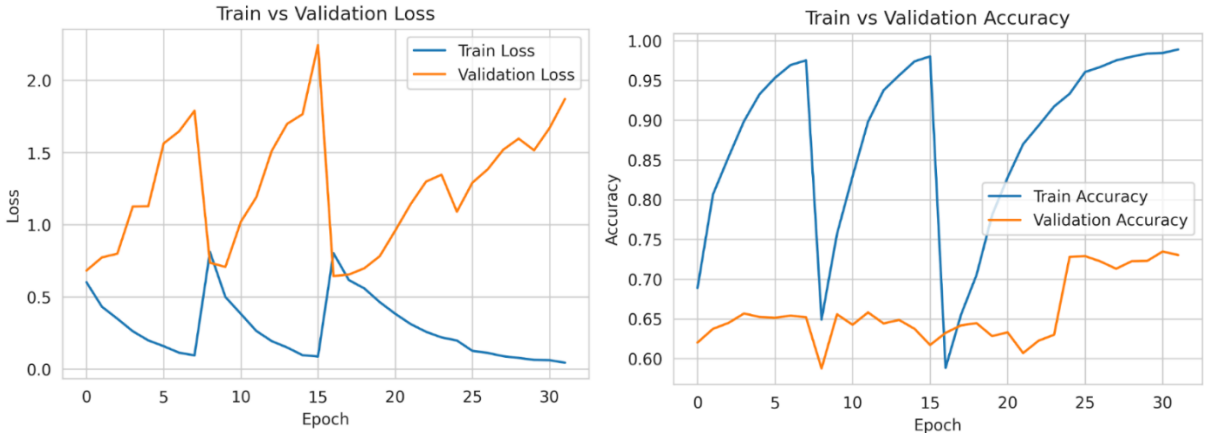


Fig:4.63 Training vs Validation Loss and Accuracy MPNet Full dataset

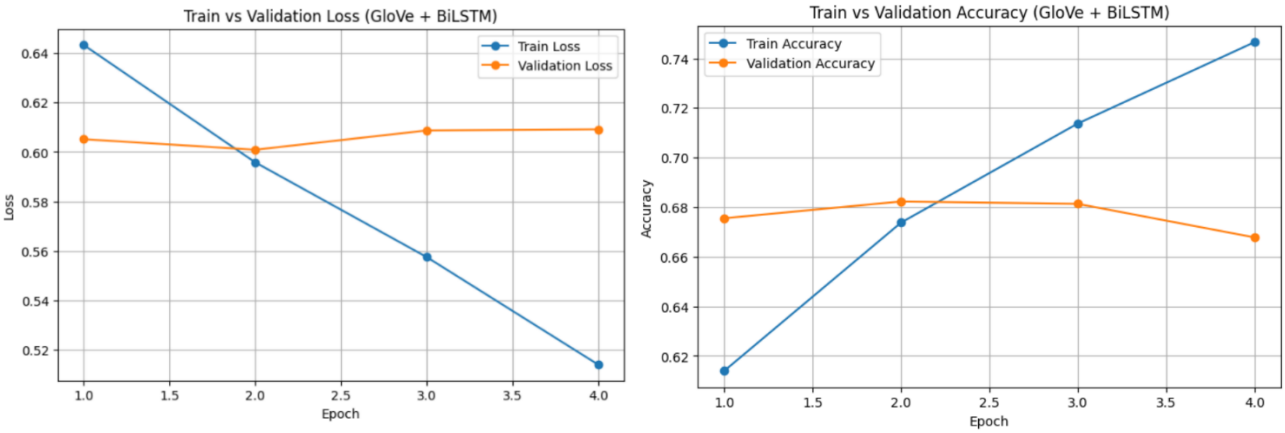


Fig:4.64 Training vs Validation Loss and Accuracy GloVe + BiLSTM Full dataset

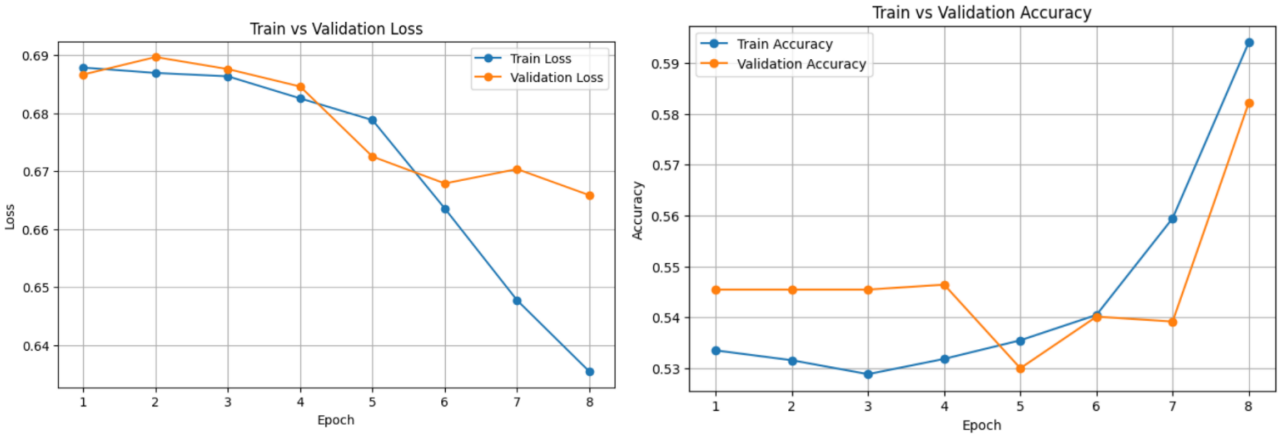


Fig:4.65 Training vs Validation Loss and Accuracy Word2Vec + BiLSTM Full dataset

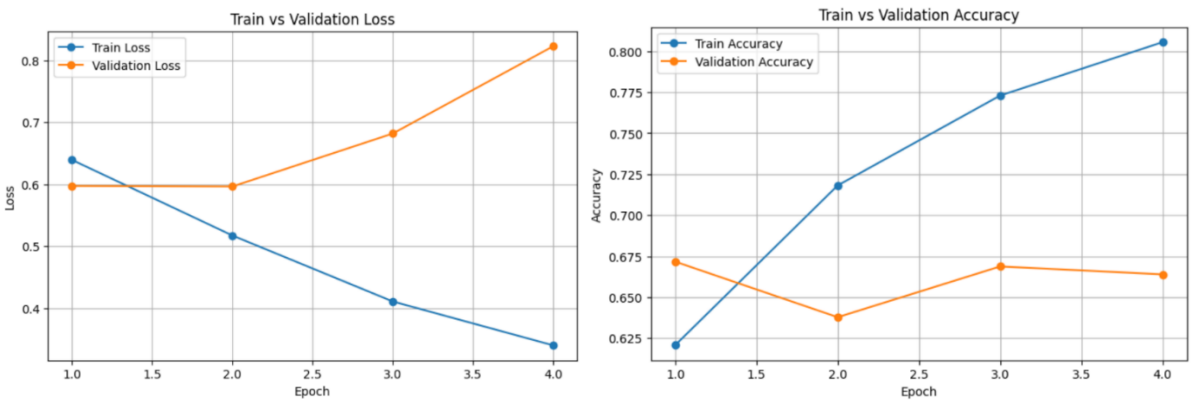


Fig:4.66 Training vs Validation Loss and Accuracy FastText + BiLSTM Full dataset

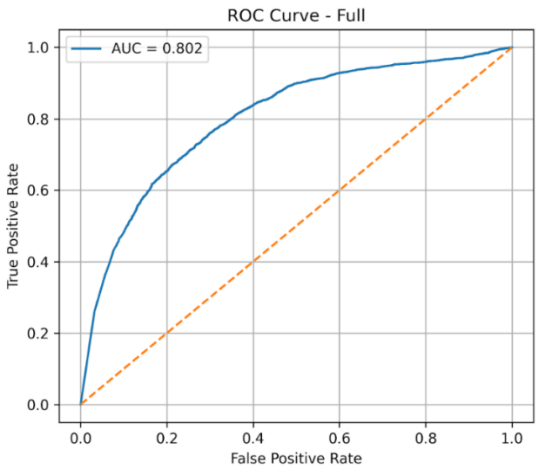


Fig:4.67 ROC-AUC Curve MuRIL Full Dataset

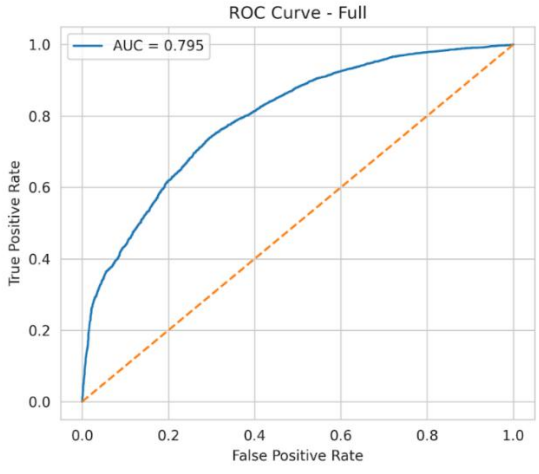


Fig:4.68 ROC-AUC Curve mBART Full Dataset

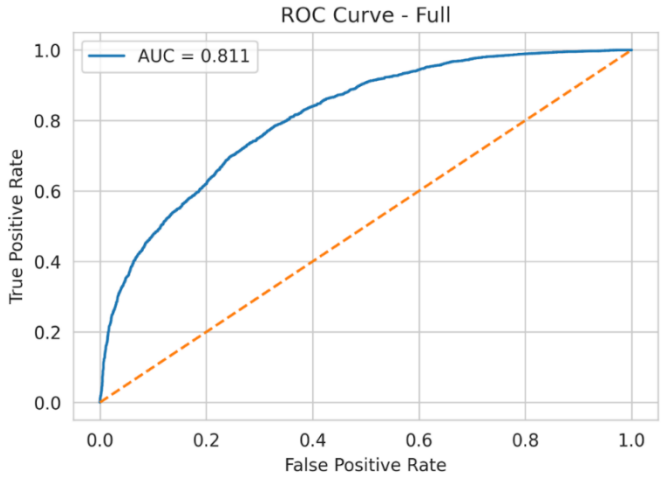


Fig:4.69 ROC-AUC Curve HingRoBERTa Full Dataset

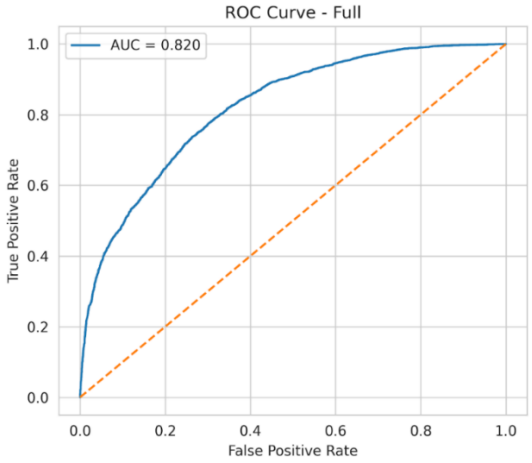


Fig: 4.70 ROC-AUC Curve MPNet Full Dataset

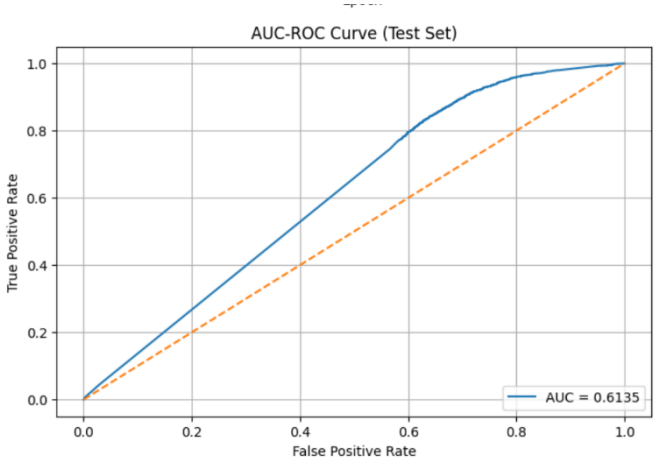


Fig4.71: ROC-AUC Curve GloVe + BiLSTM

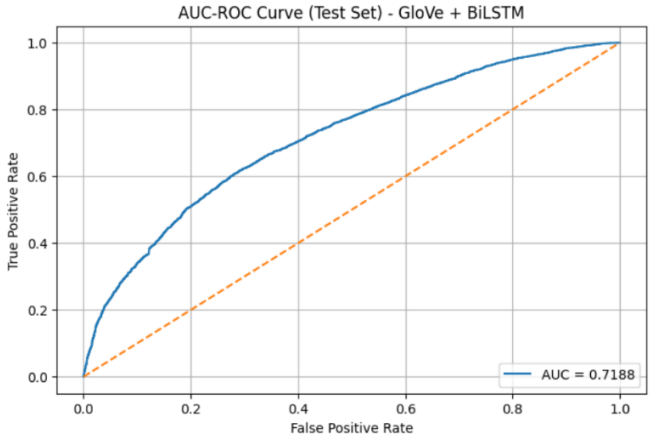


Fig:4.72 ROC-AUC Curve Word2Vec + BiLSTM

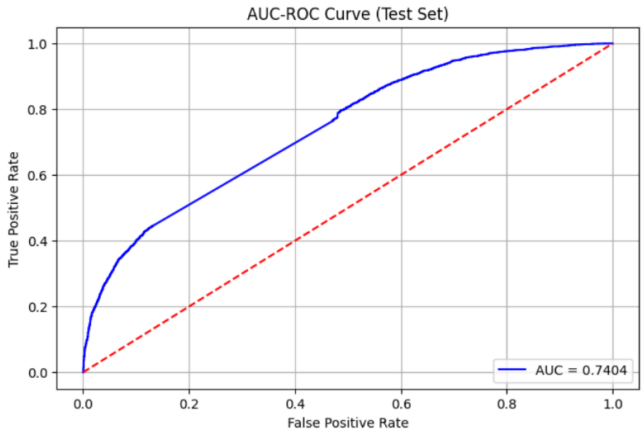


Fig:4.73 ROC-AUC Curve FastText + BiLSTM Full Dataset

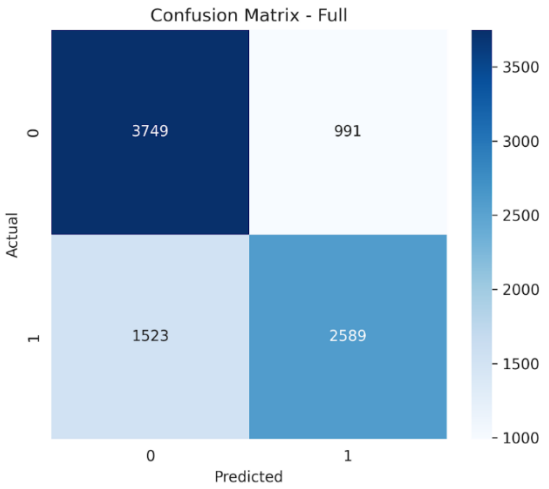


Fig:4.74 Confusion Matrix MuRIL

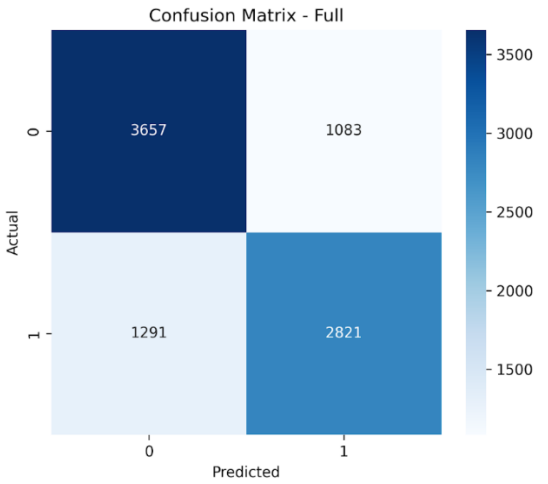


Fig:4.75 Confusion Matrix mBART

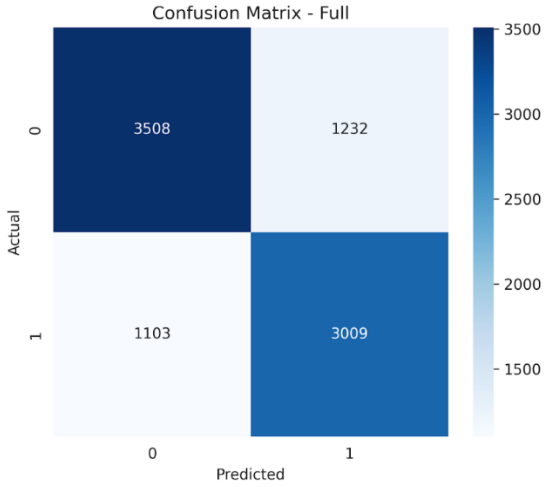


Fig:4.76 Confusion Matrix HingRoBERTa

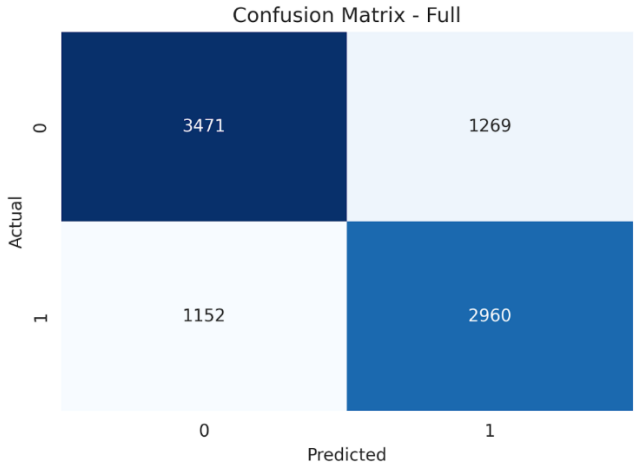


Fig:4.77 Confusion Matrix MPNet

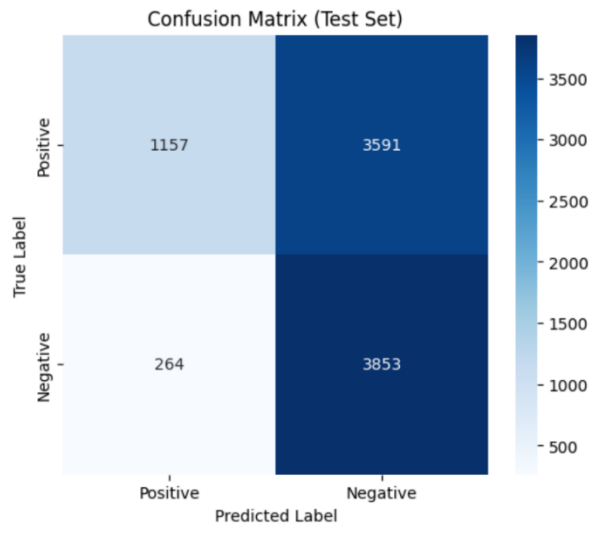


Fig:4.78 Confusion Matrix Word2Vec + BiLSTM

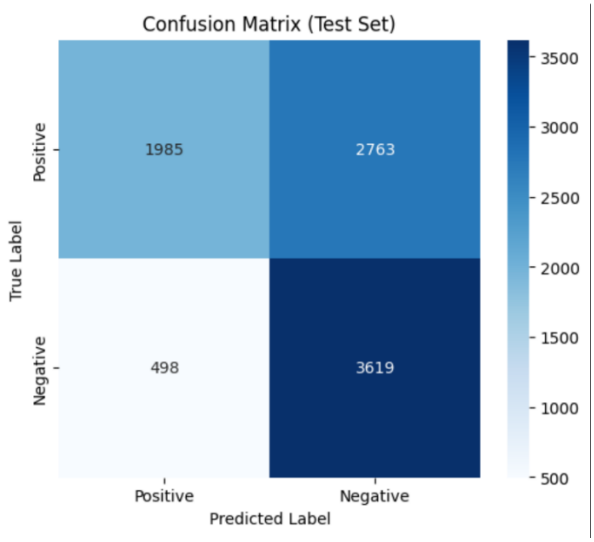


Fig:4.79 Confusion Matrix FastText + BiLSTM

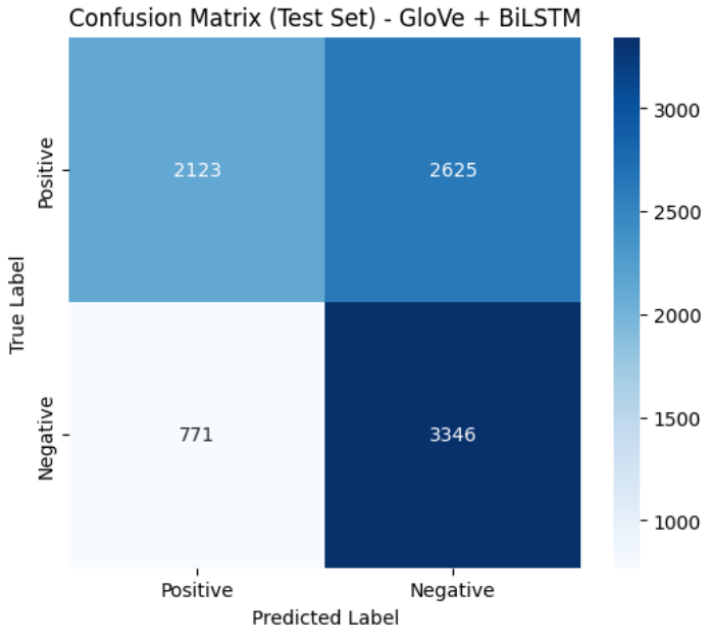


Fig:4.80 Confusion Matrix GloVe + BiLSTM

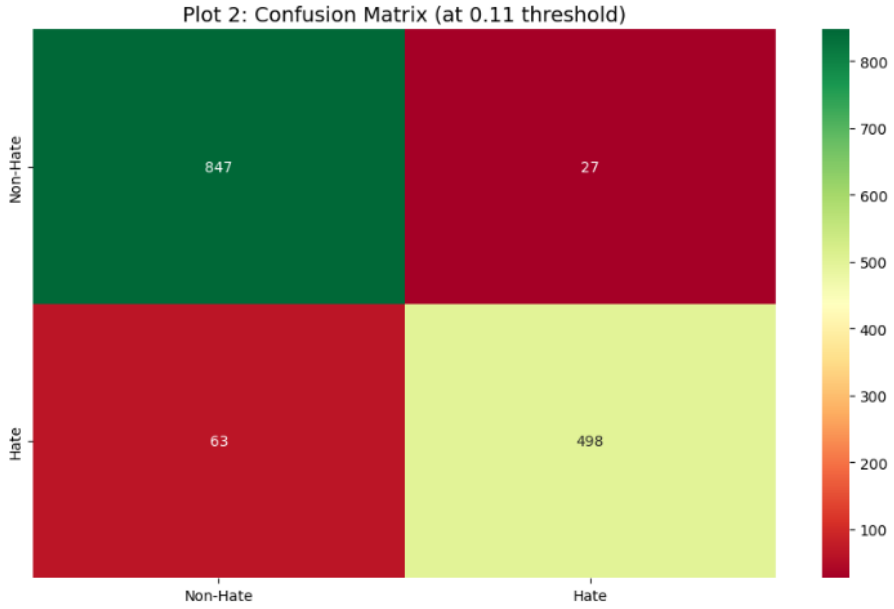


Fig: 4.81 Confusion Matrix for Sarvam *

4.3 DISCUSSION

The overall workflow consists of three methodologies in total, one with regular fine-tuning for hybrid models, second methodology involves of two different strategies one with regular fine-tuning and one with sequential training (English-> Hinglish-> Hindi-> Full) on both hybrid models and deep learning models and the third methodology involves of multiple variations sequential training (English, Hinglish, Hindi, all combined), (English, Hindi, Hinglish, all combined), (Hinglish, English, Hindi, all combined), (Hinglish, Hindi, English, all combined), (Hindi, Hinglish, English, all combined), and (Hindi, English, Hinglish, all combined) corpora to improve cross-lingual understanding. So, in our entire workflow we observed that third methodology took more training time in comparison to other two methodologies and gained very good accuracy and f1-score overall which was achieved by GloVe+BiLSTM model which is a classical hybrid model but performed better than deep learning models we used. But, in regular trainings deep learning models outperformed every other hybrid models. We have also observed Hindi(devnagiri) made overall performance degrade due to its less contextual understanding and tokens mishandle. For an extra activity we worked with Sarvam AI which is a LLM and we achieved highest of all metrics in our all workflows combined.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

This project successfully addressed the critical need for an effective and domain-specific Code-mixed sentiment analysis system, which is an obvious unexplored area in the field of natural language processing and speech technology. The outcomes of this project open up several promising opportunities for future research and development in better contextual understandings and natural language processing for code-mixed. One of the most immediate areas of expansion is the enhancement of the dataset. Our primary aim was that a good LLM or a model with a lot of billions of parameter can handle, can understand the sarcastic contexts, but the computational cost of these models are very height. We tried to make a small differentiate with developing a sentimental analysis prediction system with those models who has less parameters and achieve a near good prediction with good performance our overall works has gained the accuracy between 72%-80% which is a good start but we will move further and works with different collections of architectures to understand the depth of mechanism for enhancements. In our study we were also introduced with Explainable AI which is a modern trend model that tends to discover the happenings inside the black box. We have worked with some types of XAI that includes Shap, Lime, Captum, Integrated Gradients and all these are good powerful models. Our future scope is to study more on hybrid models as they somehow manage to perform well then deep learning models if selected smartly and we will combine the explainable AI in order to for analyzing and handling of misclassifications happening in training.

REFERENCE

[1] Singh, G. (2021). Sentiment analysis of code-mixed social media text (Hinglish). In *arXiv [cs.CL]*. https://doi.org/10.48550/ARXIV.2102.12149

[2] Thakur, V., Sahu, R., & Omer, S. (2020). Current state of hinglish text sentiment analysis. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3614442

[3] Agarwal, P. N. (2024). Improving sentiment analysis accuracy in hinglish text using hybrid deep learning approaches. *Educational Administration: Theory and Practice*, 741–750. https://doi.org/10.53555/kuey.v30i11.8739

[4] Singh, G. V., Ghosh, S., Firdaus, M., Ekbal, A., & Bhattacharyya, P. (2024). Predicting multi-label emojis, emotions, and sentiments in code-mixed texts using an emojifying sentiments framework. *Scientific Reports*, *14*(1), 12204. https://doi.org/10.1038/s41598-024-58944-5

[5] Himabindu, G. S. S. N., Rao, R., & Sethia, D. (2022). A self-attention hybrid emoji prediction model for code-mixed language: (Hinglish). *Social Network Analysis and Mining*, *12*(1). https://doi.org/10.1007/s13278-022-00961-1

[6] Yadav, S., Kaushik, A., & McDaid, K. (2024). Leveraging weakly annotated data for hate speech detection in code-mixed Hinglish: A feasibility-driven transfer learning approach with Large Language Models. In *arXiv [cs.CL]*. http://arxiv.org/abs/2403.02121

[7] Aggarwal, A., Wadhawan, A., Chaudhary, A., & Maurya, K. (2020). "Did you really mean what you said?" : Sarcasm Detection in Hindi-English Code-Mixed Data using Bilingual Word Embeddings. In *arXiv [cs.CL]*. https://doi.org/10.48550/ARXIV.2010.00310

[8] Rahul, Gupta, V., Sehra, V., & Vardhan, Y. R. (2021). Ensemble based hinglish hate speech detection. *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*.

[9] Acharya, A., & Goyal, R. (2025). Ensemble learning-based sarcasm detection in hinglish tweets using Word2Vec embedding. *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, 1–6.

[10] Aloria, S., Aggarwal, I., Baliyan, N., & Ghosh, M. (2023). Hilarious or hidden? DetectiSarcasmasm Hinglish Tweets Usinging BERT-GRU. *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*.

[11] Chutia, T., Baruah, N., & Sonowal, P. (2025). A comparative study of machine learning and deep learning approaches for identifying Assamese abusive comments on

social media. *Procedia Computer Science*, *258*, 981–992. https://doi.org/10.1016/j.procs.2025.04.335
[12] Baruah, P., Dutta, B., Sarma, S. K., & Talukdar, K. (2025). Named Entity Recognition in Assamese Language using two separate models: BiLSTM and BERT. *Procedia Computer Science*, *258*, 242–251. https://doi.org/10.1016/j.procs.2025.04.262
[13] Lalthangmawii, M., & Singh, T. D. (2025). Sentiment analysis of Mizo using lexical features in low resource based models. *Natural Language Processing Journal*, *13*(100181), 100181. https://doi.org/10.1016/j.nlp.2025.100181
[14] Talukdar, M., & Sarma, S. (2024). Bidirectional LSTM-based sentiment analysis for Assamese text. *American Journal of Computer Science and Technology*, *7*(2), 29–37. https://doi.org/10.11648/j.ajcst.20240702.11
[15] Jayanthi, S. M., Nerella, K., Chandu, K. R., & Black, A. W. (2021). CodemixedNLP: An Extensible and Open NLP Toolkit for Code-Mixing. In *arXiv [cs.CL]*.
https://doi.org/10.48550/ARXIV.2106.06004
[16] Dhekane, S. (2025, August). *Code-Mixed Hinglish Hate Speech Detection Dataset*. Kaggle.com. https://www.kaggle.com/datasets/sharduldhekane/code-mixed-hinglish-hate-speech-detection-dataset
[17] Paul, K., Wankhade, M., & Dutta, S. C. (2025). Dynamic multi-attention fusion for joint intent detection and slot filling in code-mixed language understanding. 2025 6th International Conference on Recent Advances in Information Technology (RAIT), 1–6.
[18] Tho, C., Warnars, H. L. H. S., Soewito, B., & Gaol, F. L. (2020). Code-mixed sentiment analysis using machine learning approach – A systematic literature review. 2020 4th International Conference on Informatics and Computational Sciences (ICICoS), 1–6.
[19] Srivastava, V., & Singh, M. (2021). Challenges and considerations with code-mixed NLP for multilingual societies. In arXiv [cs.CL]. http://arxiv.org/abs/2106.07823