

ARTICLE

A Comparative Study of Sentiment Analysis in Movie Reviews Using Machine Learning, Deep Learning, and Transfer Learning Models with Explainable AI

Md. Shymon Islam¹, M. Raihan², S. M. Reazul Islam Rifat², Md. Mahmudul Haque², Kamrul Golder², Riaj Hossen², Sakib Ahmed², Taher Al-Shehari³, Effa-Bassey Ettah⁴, Mohammed Kadrie³ and Anupam Kumar Bairagi^{5,*}

¹Department of Computer Science & Engineering, Shahjalal University of Science and Technology, Sylhet, Bangladesh

²Department of Computer Science and Engineering, North Western University, Khulna, Bangladesh

³Computer Skills, Department Self-Development Skill, Common First Year Deanship, King Saud University, Riyadh, Saudi Arabia

⁴Turing Innovation Catalyst, The University of Manchester, Oxford Road, Manchester, UK

⁵Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh

*Corresponding Author: Anupam Kumar Bairagi. Email: anupam@cse.ku.ac.bd

Received: 03 June 2026; Accepted: 21 September 2026; Published: 23 October 2026

ABSTRACT: Sentiment analysis is essential across all languages, given its widespread use in movie reviews. Regrettably, there is a shortage of benchmark materials for Bangla movie evaluations, including standard datasets and research studies. One of the significant contributions of this work is the development of a novel Bangla movie review dataset compiled by 6 individuals over 4 to 6 months. The dataset comprises movie reviews collected from YouTube using a JavaScript crawler. In this investigation, a novel dataset of 22,289 ratings of different Bangla movies from YouTube videos has been gathered. Next, the evaluations were carefully annotated by six native speakers of Bangla. The main aim of this research is to use multi-models to detect the sentiments of Bangla movie reviews with high accuracy and to use Explainable Artificial Intelligence (AI) to get the reason behind the performance pattern of the particular models. We did an empirical study on 10 deep learning (DL) models and 6 different machine learning (ML) approaches, 4 transfer learning (TL) models, and a stacking ensemble model, whose base classifiers were the 10 DL models. Both train-test splitting and 10-fold cross-validation were applied for evaluation. Under 10-fold cross-validation, our stacking model achieves 94.91% accuracy and an F1-score of 0.95 (94.58% and 0.95, respectively, under the holdout protocol). Statistical significance was assessed using fold-level Friedman tests (10-fold cross-validation accuracies as repeated measures) within each model family, followed by Holm-corrected pairwise Wilcoxon signed-rank post-hoc tests. The omnibus tests were significant for all families ($p < 0.05$), and post-hoc comparisons confirmed that Random Forest, mBERT, and the stacking ensemble significantly outperformed most within-family competitors and the best representative model of each competing family. A supplementary one-sided Wilcoxon signed-rank test across five random seeds provided additional, limited-sample evidence that the stacking ensemble consistently outperforms the runner-up RNN model. This study also analyzes the factors that influence the performance of models at both local and global levels using the Local Interpretable Model-agnostic Explanation (LIME) and SHapley Additive eXplanation (SHAP) provided by explainable AI.

KEYWORDS: Movie review analysis; bangla dataset; machine learning; deep learning; transfer learning; ensemble learning; stacking model; explainable AI

1 Introduction

Sentiment analysis is the method of determining the sentiment or emotional content of a corpus of textual information [1]. It can pick up on the affective connotations and the point of view represented by the specific words that are used to express them. Sentiment analysis (SA) is an important subfield of natural language processing (NLP) and is crucial in extracting insights from textual data [2]. Sentiment Analysis is basically aimed at measuring human opinions, attitudes, and emotional states. The focus is on the detection, categorization, and measurement of the author's positive or negative thoughts, feelings, and attitudes about a given topic. This is done by looking at the explosion of popularity of movie review analysis applications and the lack of resources for Bangla movie review analysis.

1.1 Motivation of the Study

Bangla movie reviews are a good way for reviewers to rate a movie's quality and recommend it to others [2,3]. Movie enthusiasts can decide whether to watch, rent, or purchase the movie without revealing significant plot details or shocks. Although Bangla is one of the predominant languages spoken by nearly 250 million people, of which 160 million are Bangladeshis, Bangla is still in its early stages of development for digital resources and tools compared to other rich languages such as English, Arabic, Turkish, etc. [4]. Most sentiment analysis research and publicly available datasets are limited to English and other languages with abundant resources, such as Arabic, Turkish, Hindi, Gujarati, and many more [4]. Bangla, known for its complex morphology, has a linguistic legacy that has developed over thousands of years, manifested in the diversity of its dialects, and deeply embedded in long-standing cultural traditions. To effectively capture the mood conveyed in Bangla writings, intelligent AI methodologies are required, given the language's complex structure, multiple dialects, and cultural intricacies. SA in Bangla movie reviews has several applications, such as marketers, advertisers, recommendation engines, movie quality evaluation, examination of market trends, market research, economic indicators [5], movie recommendation systems, classifying films [2], understanding public opinion on a movie industry [3], etc. All these trending applications in movie review analysis and the lack of resources in the Bangla language greatly motivated us to work in this domain.

1.2 Contributions of the Study

The main objective of this work is to analyze user sentiment in Bangla movie comments on YouTube and to build an automated system to detect sentiment from Bangla movie reviews. Machine Learning (ML), Deep learning (DL), and Bidirectional Encoder Representation from Transformers (BERT) techniques have been used to predict positive and negative attitudes in the majority of studies on multi-class SA. This work aims to employ learning models from different domains to identify the feelings expressed in Bangla movie reviews, creating a new dataset. The following are the primary contributions that have been discovered in the article:

1. Develops a novel dataset of 22,289 reviews collected from Bangla YouTube movies and annotates them as positive or negative by six native speakers.
2. Introduces the proposed stacking ensemble model consisting of 10 different DL models as base classifiers, which outperforms the others, obtaining an outstanding accuracy of 94.91% under the 10-fold cross-validation protocol.
3. Utilizes LIME and SHAP modules from explainable AI to interpret the reason for a model's performance.

The structure of this article is as follows: [Section 2](#) discusses the recent literature, [Section 3](#) presents the research methodology for Bangla movie review analysis, [Section 4](#) presents the study and discussion of the

experimental results, [Section 5](#) describes the limitation and future relevant works, and [Section 6](#) concludes the work.

2 Related Works

Here are some recent studies on the sentimental analysis of Bangla movie reviews. Each approach has both efficiencies and pitfalls compared to other methods.

Lin et al. [6] proposed a bi-directional flow weighted regression method to reveal the knowledge of social media sentiment scores predicted by large language model (LLM). The technique they use is explainable artificial intelligence and an analysis of a regression type to identify the language elements that are most representative of the classification of sentiment, thus adding transparency and interpretability to the model. Ernie large language model achieved the highest accuracy with an overall accuracy score of 0.921 (92.1%), which is better than ChatGPT 3.5, BERT, RoBERTa, RBTL3 (a 3-layer distilled RoBERTa-www-ext-large variant), ELECTRA, and SnowNLP. The research highlights that explainability methods could be beneficial for gaining semantic understanding of sentiment prediction tasks and improve the certainty of LLM-based sentiment analysis. The main emphasis is on understanding sentiment from social media rather than Bangla, and it does not discuss the creation of an open access Bangla movie review sentiment classification dataset. While it has some advantages, the data for this research is only available at the provincial level, which is less precise than the national level; and the data comes only from a single platform (Weibo), and might be subjected to sentiment bias because of the moderating effect. The proposed study, however, proposes a large-scale Bangla movie review dataset for its study, a comprehensive comparison of both machine learning, deep learning, transfer learning, and ensemble models, as well as sentiment classification using SHAP and LIME for a low-resource language.

Transformer models have proven to be a success for sentiment analysis tasks in low-resource languages in recent years. Chattu et al. [7] created the multi-domain annotated Telugu sentiment classification dataset called Sentikanna and tested the performance of various transformer-based models, such as IndicBERT, RoBERTa, DeBERTa, and XLM-RoBERTa. They found that with other transformer models, XLM-RoBERTa achieved the top accuracy of 79.42% on their recipe review data. The study highlighted the advantages of using multilingual transformer models for acquiring context-specific and semantic knowledge in resource-limited languages. However, the authors pointed out that one of the main challenges with low-resource language sentiment detection is the lack of high-quality annotated language corpora, so there is a need for more research on other under-resourced languages such as Bangla.

Rifa et al. [8] proposed a transformer-based framework (CineXDrama) to analyze movie and drama comments in Bangla from social media (YouTube platform). The study demonstrates the use of Bangla-BERT model and explainable artificial intelligence methods such as LIME to increase the interpretability of the model. The Bangla-BERT model achieved the highest accuracy of 93.30% in sentiment analysis. It was well performed but primarily focused on the comments from YouTube, and not on particular movie review datasets.

Mirza et al. [9] analyzed some of the machine learning and transformer models for sentiment classification in movies reviews. The study evaluated a number of algorithms for English movie review data sets of large scale and demonstrated the effectiveness of the advanced algorithms. The research however was confined to English datasets and didn't consider low-resourced languages like Bangla. The best performance was obtained in their empirical study by using BERT model. The BERT model achieved highest accuracy score of 0.93 in the IMDB 50K dataset, 0.895 in the IMDB 20K dataset, and 0.76 in the Rotten Tomatoes dataset.

The authors in [2] suggested a sentiment recognition approach for Bangla film reviews by using 1000 Bangla reviews collected from Facebook. They found their empirical results to be the most accurate (92.35%) for Unigrams with the Random Forest classifier and 90.03% accuracy for Trigrams with the Random Forest classifier, while the Decision Tree (DT) was the most accurate with Bigrams (89.50%). Another ML-based method is proposed in [10]. To carry out bipolar sentiment analysis of the English movie reviews, they gathered a dataset of user reviews from the IMDB website, which consists of more than 25,000 tagged reviews. Several preprocessing steps were carried out to reduce the dataset's dimensionality, such as stop words removal, stemming, and so on. Three machine learning algorithms, namely Naive Bayes (NB), Logistic Regression (LR), and Support Vector Machine (SVM), were studied in terms of precision, recall, F1-score, and accuracy. They compared the accuracy of the models they used, with SVM being the most successful with 73%, while Logistic Regression (LR) showed 70%, and Naive Bayes (NB) 71%.

In work done by the authors of [11], they suggested a method using user ratings and number of stars given by the IMDB website. For classification, they applied the SVM algorithm, and for studying the emotions in the IMDB reviews, the star rating component was used. In [12] the authors explored the work on the IMDB dataset from Kaggle. From the comparison of the implemented algorithms, Logistic Regression (LR) was found to be the most efficient algorithm, which has an accuracy of 88%, while RF and KNN showed 78%, and 61% accuracy, respectively. Considering the wide use of Bangla language and its importance, Reference [5] developed an approach to analyzing emotions in Bangla written texts. The authors tested a number of machine learning algorithms—Logistic Regression (LR), Stochastic Gradient Descent (SGD), Linear Support Vector Machine (L-SVM), Multinomial Naive Bayes (MNB), Bernoulli Naive Bayes (BNB), SVM, DT, RF, and a custom-made recurrent neural network named Long Short-Term Memory (LSTM)—to determine which one best captured positive and negative sentiments from a dataset of 4000 Bangla movie reviews collected from several online platforms. An accuracy of 84.25% was the best that was obtained. The authors of [3] had discussed the significance of sentiment mining in the developing film industry of Bangladesh. They performed classification on a dataset of 3000 reviews with the help of several ML classifiers on the features derived from the TF-IDF using unigram, bigram, and trigram models. The best accuracy of 86% was obtained by their empirical study with MNB as the classifier and unigram as the features, while for the bigram features the MNB classifier showed 85% accuracy. The authors analyzed movie reviews in [13] using word embeddings and deep learning, explored embedding vector sizes of 150, 250, 350 and 450, and applied six foreign language and Chinese datasets. They got the best accuracy of 86.87% with the SVM model with the Word2Vec feature metric, which was an empirical study.

In [4], the authors proposed a method to automate examining viewer responses in Bangla movies in 2019 using a dataset of over 4000 tagged samples. The authors used the Long Short-Term Memory (LSTM) and SVM algorithms. It achieved 88.90% accuracy with SVM (TF-IDF), 88.38% with MNB, and 82.42% with LSTM. In another attempt, Reference [14] presented a technique for sentiment analysis (SA) of movie reviews, whose sentiment would be positive or negative. The authors applied several preprocessing steps, like tokenization to convert words into word vectors, stemming to extract the word root from the word, and gain ratio for feature selection. They tested eight classifiers on a real-world data set with 42,926 tagged reviews drawn from IMDB database using Naive Bayes (NB), Decision Tree (DT), SVM, Bayes Network (BN), KNN, Ripper Rule Learning (RRL), Stochastic Gradient Descent (SGD), and RF. The dataset was split into 66% for training and 34% for testing, and the classifiers were evaluated according to performance measures like precision, recall, F-measure, accuracy and area under the curve (AUC). Among the tested classifiers, RF gave the best results, achieving the highest precision (0.93), recall (1.00), F-measure (0.96), accuracy (96.01%), and AUC (0.96), while the KNN model achieved a 92.92% accuracy score.

2.1 Findings from Related Works

After studying the existing literature, we have identified several key observations with scope for further research. A summary of recent works on movie review analysis is presented in [Table 1](#). The subsequent sections will demonstrate the key findings.

Table 1: Summary of existing works on sentiment analysis and movie review classification.

Reference	Year	Dataset Used	Model Deployed	Evaluation (Accuracy)	Advantages	Disadvantages
[6]	2026	Social media sentiment dataset (10,860 Weibo posts, Chinese)	ChatGPT 3.5, BERT, RoBERTa, RBTLL3, ELECTRA, SnowNLP, Ernie	92.1% (Ernie LLM)	Provides explainable sentiment analysis with semantic interpretation using XAI	Provincial-scale data; and single-platform (Weibo) sentiment source limit generalizability
[9]	2025	IMDB 50K, IMDB 20K, Rotten Tomatoes Reviews	NB, LR, SVM, DT, RF, BERT	0.93 (BERT-IMDB 50K) 0.895 (BERT-IMDB 20K) 0.76 (BERT-Rotten Tomatoes)	Provides comparative analysis of ML and transformer models	Limited focus on Bangla movie reviews
[7]	2025	3115 Telugu Reviews (Movie, Tourism, Recipe)	IndicBERT, RoBERTa, DeBERTa, XLM-RoBERTa	79.42% (XLM-RoBERTa)	Introduces a multi-domain Telugu dataset and compares transformer models for low-resource sentiment analysis	Limited to Telugu language; binary sentiment analysis; no explainable AI techniques used
[8]	2024	14,000 Bangla movie/drama YouTube comments	Bangla-BERT, Transformer-based models	93.30% (Bangla-BERT)	Uses transformer models and incorporates LIME for interpretation	Focuses on YouTube comments rather than direct movie reviews
[10]	2023	25,000 Reviews (IMDB, English)	LR, SVM, NB	70% (LR), 73% (SVM), 71% (NB)	Offers suggestions for industrial usage	Lacks theoretical depth; dataset bias
[2]	2023	1000 Reviews (Bangla)	DT, RF, MNB, KNN, L-SVM	92.35% (RF-Unigram) 90.03% (RF-Trigram) 89.50% (DT-Bigram)	Enhances sentiment analysis for Bangla reviews	Very small dataset; no explainable AI used
[12]	2023	IMDB (English)	LR, RF, KNN	88% (LR), 78% (RF), 61% (KNN)	Combines ML and NLP in a unified framework	Overfitting not addressed; lacks interpretability
[5]	2021	4000 Reviews (Bangla)	LR, SGD, L-SVM, MNB, BNB, SVM, DT, RF, LSTM	84.25% (LSTM)	Applies both ML and DL to Bangla data	Small dataset; no explainable AI approach used
[3]	2021	3000 Reviews (Bangla)	LR, DT, RF, MNB, KNN, SVM, SGD	86% (MNB-Unigram) 85% (MNB-Bigram)	Focuses on Bangla sentiment classification	Small dataset; uses only basic ML models

(Continued)

Table 1 (continued)

Reference	Year	Dataset Used	Model Deployed	Evaluation (Accuracy)	Advantages	Disadvantages
[13]	2021	6 multilingual datasets (Chinese and Foreign)	Word2Vec + SVM	76.23%–86.87% (Word2Vec + SVM)	Covers diverse languages across datasets	Insufficient investigation of model behavior
[4]	2019	4000 Reviews (Bangla)	SVM, MNB, LSTM	88.90% (SVM-TFIDF) 88.38% (MNB) 82.42% (LSTM)	Advances sentiment analysis in Bangla context	Small dataset used
[14]	2019	42,926 Reviews (IMDB, English)	NB, DT, SVM, BN, KNN, RRL, RF, SGD	96.01% (RF) 92.92% (KNN)	Offers practical insights for film review analysis	Lacks theoretical justification and model interpretation

2.1.1 Lack of Publicly Available Standard Dataset

The data set is the most critical requirement for any classification/regression problem. The success of any technical research primarily depends on the quality of the used dataset. The Bangla NLP research community always faces the difficulty of the unavailability of standard resources, especially datasets and other technical tools. In the aforementioned studies ([2–5]), the researchers used sets of only 1000, 3000, 4000, and 4000 instances for analyzing Bangla movie reviews, which is a primary concern in this study.

2.1.2 Very Few Attempts to Use Advanced Algorithms

In recent years it has been found that in most cases the deep learning algorithms are better than traditional machine learning algorithms. As seen from the summary in Table 1, most traditional machine learning algorithms are used to analyze the movie reviews. However, it is equally important to delve deep into the different well-established deep learning and BERT-based transfer learning and other advanced algorithms in this regard.

This study will consider these issues and make appropriate progress in Bangla movie review analysis.

2.1.3 Lack of Model Interpretability

In this era of artificial intelligence, there is no longer any excuse to treat a model or classifier as a black box; now, it is required to understand a model's behavior for classification. Although some recent research has begun to apply explainable AI to related tasks such as Lin et al. [6] on social media sentiment and Rifa et al. [8] on YouTube movie/drama comments using LIME. Sentiment analysis studies dedicated to Bangla movie reviews using standardized datasets (see Table 1) have not fundamentally addressed this issue, and no study has combined both LIME and SHAP for a comprehensive local and global interpretation.

2.2 Research Gap and How This Study Addresses It

The results obtained from related works are presented in Section 2.1, summarizing the research gaps that were identified in the existing literature. The lack of a publicly available standard and diverse dataset is the major drawback of the Bangla movie review sentiment analysis domain. This study proposes a new dataset: 22,289 reviews collected from Bangla YouTube movies, and six native speakers annotate them as positive or negative with the result and perform the overall agreement analysis among the six annotators (Fleiss' Kappa)

and pairwise agreement analysis (pairwise Cohen's Kappa) and make the developed dataset public. The second gap of the existing literature is the thorough and rigorous experiments using traditional algorithms to advanced algorithms on a standard dataset. The study compares six basic ML algorithms (BNB, SVM, KNN, LR, MNB, RF), six basic DL models (CNN, RNN, LSTM, Bi-LSTM, GRU and Bi-GRU), four basic hybrid DL algorithms (CNN-LSTM, CNN-GRU, CNN-BiLSTM, and CNN-BiGRU), and four transformer algorithms (BERT, mBERT, Bangla-BERT, and DistilBERT). Furthermore, the proposed stacking ensemble framework with ten different DL models as base classifiers is presented that shows that it has outperformed the currently existing state-of-the-art methods with an outstanding accuracy of more than 94% from two different testing methods: holdout and cross-validation strategies. The generalizability of the proposed framework is also proved using six other external sentiment analysis datasets examining with our best models from different domains. Another limitation is the limited applicability of the model's interpretability, especially in the case of Bangla movie review classification. Unlike [6,8] (for adjacent domains, social media and YouTube comments, respectively), this study is directly applied to interpretability in the context of Bangla movie reviews with both LIME and SHAP.

3 Research Methodology

This study aims to develop multiple ML, DL, and BERT-based models and a stacking ensemble model capable of identifying positive and negative sentiments in Bangla movies, as illustrated in Fig. 1, which shows the system's functioning procedure. The subsequent sections will discuss dataset preparation, data preprocessing, feature extraction, and design algorithms.

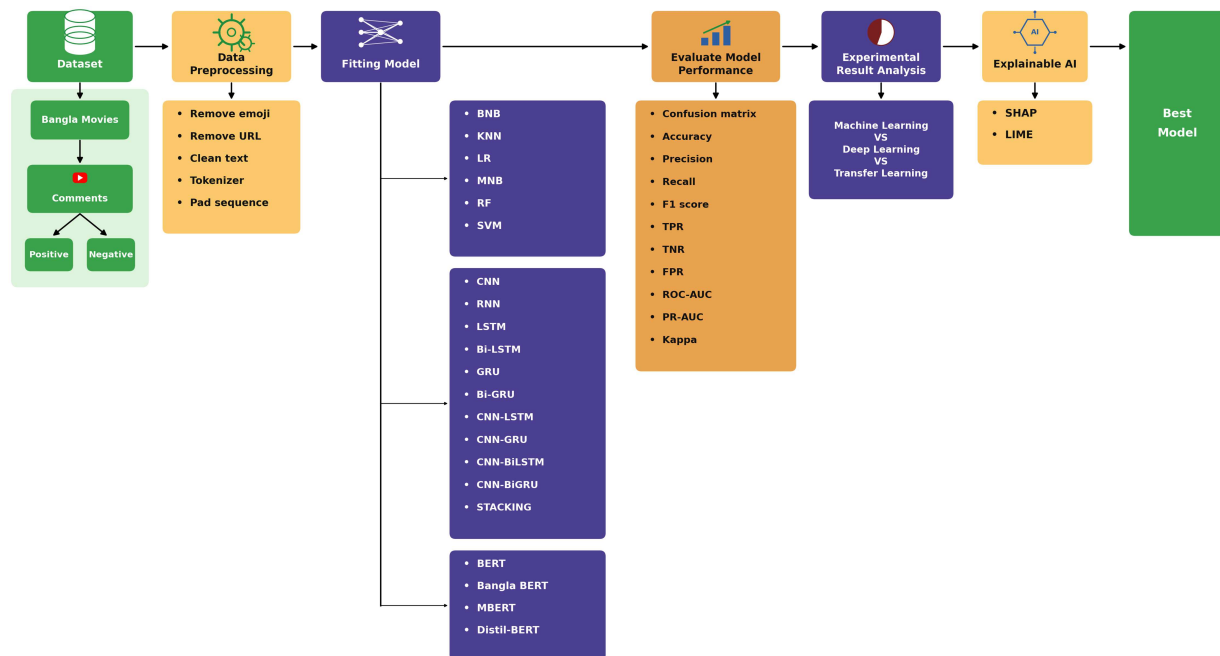


Figure 1: Research methodology for the sentiment analysis on Bangla movie reviews.

3.1 Dataset Preparation

For sentiment classification models to be reliable, a good annotated dataset is crucial [15]. But, there are still limited datasets available in the public domain for Bangla sentiment analysis. This study proposes a new Bangla movie review dataset sourced from YouTube to mitigate this issue and demonstrates its suitability for

search and retrieval tasks. To accomplish this, a custom web crawler was designed, written, and implemented in JavaScript, which helps to collect user comments from 35 Bangla movies, which are spread across the time range 1950 to 2023, from which around half of the movies were released in the last 10 years (2015–2023) from various age groups and from various movies. The final data set of movie reviews contained 22,289 movie reviews, as detailed in the study's documentation file (<https://doi.org/10.5281/zenodo.22106939>). Table 2 shows the movies that were used for data collection.

Table 2: List of movies.

SN	Language	Movie	Release Year
1	Bangla	Password (পাসওয়ার্ড)	2019
2	Bangla	Nirsartha Bhalobasha (নিস্বার্থ ভালোবাসা)	2022
3	Bangla	Most Welcome (মোস্ট ওয়েলকাম)	2012
4	Bangla	Hridoy Bhanga Dheu (হৃদয় ভাঙ্গা ঢেউ)	2017
5	Bangla	Molla Barir Bou (মোল্লা বাড়ির বউ)	2005
6	Bangla	Goriber Tarzen (গরীবের টারজান)	2019
7	Bangla	Palta Jobab (পাল্টা জবাব)	2020
8	Bangla	Tokai (টোকাই)	2023
9	Bangla	Hero Alom The Power (হিরো আলম দা পাওয়ার)	2018
10	Bangla	Ammajan (আম্মাজান)	1999
11	Bangla	Monpura (মনপুরা)	2009
12	Bangla	Hajar Bochor Dhore (হাজার বছর ধরে)	2005
13	Bangla	Uttarer Sur (উত্তরের সুর)	2012
14	Bangla	Leader Ami Bangladesh (লিডার আমি বাংলাদেশ)	2023
15	Bangla	Golam (গোলাম)	2016
16	Bangla	Shahosi Hero Alom (সাহসী হিরো আলম)	2020
17	Bangla	Mar Chakka (মার ছক্কা)	2017
18	Bangla	Tumi Acho Tumi Nei (তুমি আছো তুমি নেই)	2021
19	Bangla	Sujon Majhi (সুজন মাঝি)	2023
20	Bangla	Hero Alom The Gangster (হিরো আলম দা গ্যাংস্টার)	2016
21	Bangla	My Name Is Simi	2016
22	Bangla	Paglami (পাগলামী)	2019
23	Bangla	Most Welcome 2 (মোস্ট ওয়েলকাম ২)	2014
24	Bangla	Prem Korbo Tomar Sathe (প্রেম করবো তোমার সাথে)	2017
25	Bangla	Shedin Bristy Chilo (সেদিন বুস্টি ছিল)	2014
26	Bangla	Gharer Bou (ঘরের বউ)	1990
27	Bangla	Boro Bou (বড় বউ)	1988
28	Bangla	Loafer	1997
29	Bangla	Chowdhury Paribar (চৌধুরী পরিবার)	1998
30	Bangla	Vidyasagar (বিদ্যাসাগর)	1950
31	Bangla	Bir (বীর)	2020
32	Bangla	Valobasha Zindabad (ভালোবাসা জিন্দাবাদ)	2013
33	Bangla	Shopner Thikana (শপনের ঠিকানা)	1995
34	Bangla	Chotto Shongshar (ছোট সংসার)	2011
35	Bangla	Norom Gorom (নরম গরম)	1984

3.2 Annotation Workflow and Quality Control

The dataset construction process consisted of several distinct stages such as manual quality screening, removal of sarcastic and irrelevant reviews, independent sentiment annotation, inter-annotator agreement analysis, and tied data removal. These stages served different purposes and were conducted separately.

3.2.1 Manual Quality Screening

After the initial collection of 24,289 YouTube movie reviews, a manual quality screening process was conducted to identify reviews unsuitable for binary sentiment analysis. This stage focused exclusively on data quality rather than sentiment labeling. Reviews containing sarcasm, irony, advertisements, spam, duplicated content, incomplete text, and other non-opinionated or irrelevant information were identified and removed.

The screening procedure involved manual examination and cross-checking among the designated reviewers to improve consistency in identifying unsuitable content. A total of 1500 reviews (see more details in [Section 3.2.2](#)) were excluded during this stage. As a result, 22,789 reviews remained and were subsequently forwarded to the independent sentiment annotation stage.

Importantly, the quality screening decisions were not used to calculate inter-annotator agreement for sentiment classification. The agreement analysis reported in this study was based on the independent positive/negative sentiment annotations described below.

3.2.2 Removal of Sarcastic and Irrelevant Reviews

After collecting the reviews, a manual quality-control procedure was applied to identify content unsuitable for sentiment annotation. A total of 24,289 reviews were initially collected. Four of the six annotators (four graduate students) participated in this screening stage. The reviews were divided between two initial reviewers, with Annotator 1 examining 12,144 reviews and Annotator 2 examining 12,145 reviews. They identified 824 and 676 unsuitable reviews, respectively. The identified reviews included sarcastic or ironic pieces of content, spam, advertisements, duplicate entries, incomplete text, and other irrelevant or non-opinionated content.

The reviews identified during the initial screening were subsequently re-examined by two additional annotators. Annotator 3 reviewed the 824 reviews identified by Annotator 1, while Annotator 4 reviewed the 676 reviews identified by Annotator 2. Both Annotators 3 and 4 agreed to delete all the instances suggested by the initial two quality screen annotators (Annotators 1, and 2). Following this verification, all 1500 reviews were excluded from further annotation. Thus, 22,789 reviews remained for independent sentiment annotation. [Fig. 2](#) illustrates the quality-control process.

Sarcastic reviews were excluded because their intended sentiment may differ from their literal wording, which can introduce ambiguity during sentiment annotation. [Table 3](#) provides examples of such cases. For example, positive expressions such as “অসাধারণ” (amazing) and “দারুণ” (excellent) may be used to express negative opinions, while negative expressions may occur in positively intended statements. Such cases are difficult to interpret using lexical information alone and may introduce noise into sentiment classification.

3.2.3 Independent Sentiment Annotation

The remaining 22,789 reviews were independently annotated by six native Bangla speakers, including four graduate students and two faculty members. Each annotator assigned one of two sentiment labels, *positive* or *negative*, to every review without access to the labels assigned by the other annotators. Thus, each retained review received six independent sentiment judgments.

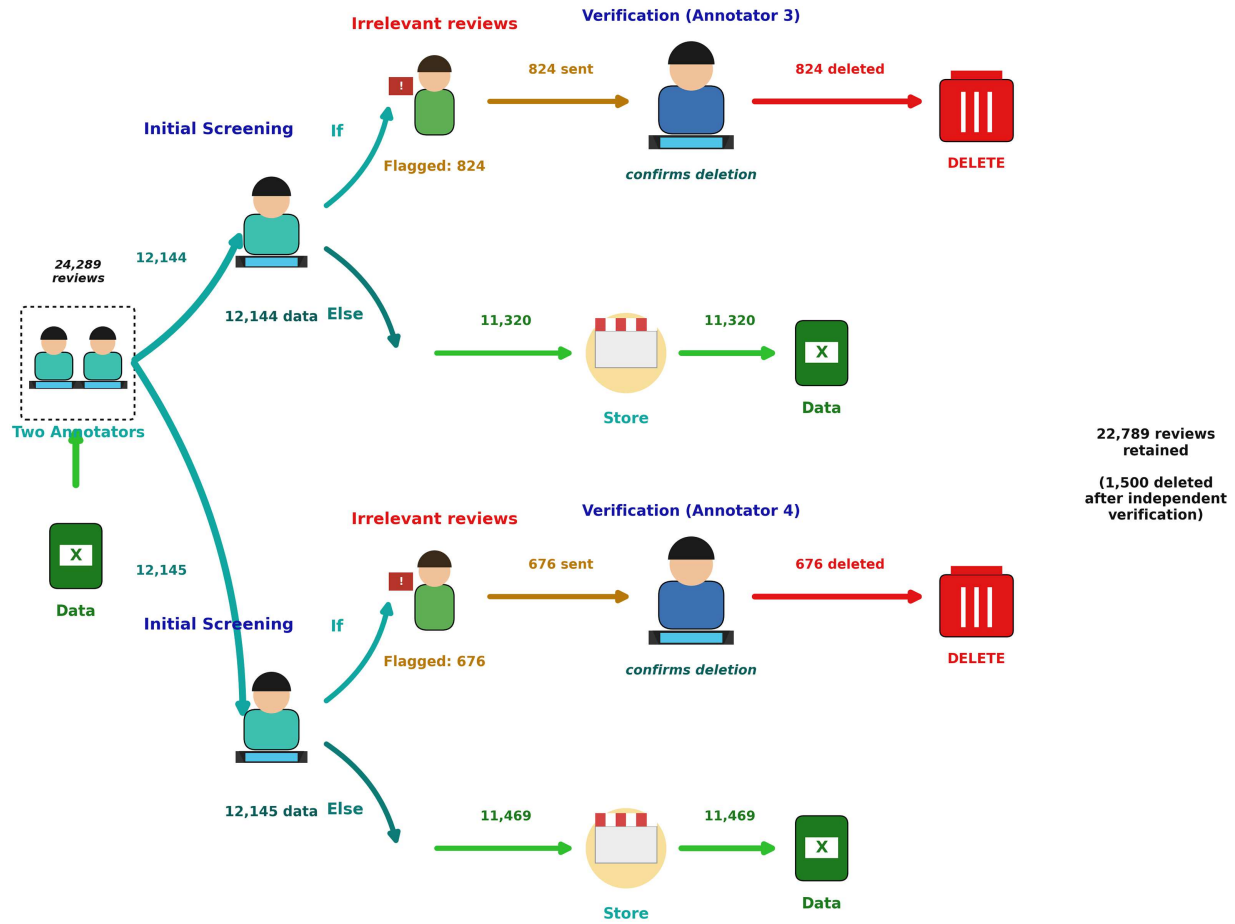


Figure 2: Manual quality-control process for identifying and removing unsuitable reviews before sentiment annotation.

Table 3: Examples of sarcastic bangla movie reviews.

No.	Bangla Review	English Translation	Type	Sentiment
1	বাহ! কী অসাধারণ সিনেমা! পুরো তিন ঘণ্টা ঘুমানোর সুযোগ পেলাম।	Wow! What an amazing movie! It gave me the opportunity to sleep peacefully for three hours.	Positive words → Negative sarcasm	Negative
2	কী দারুণ অভিনয়! এমন অভিনয় দেখে বিরক্ত হওয়ার অভিজ্ঞতা সত্যিই অসাধারণ।	What wonderful acting! The experience of getting annoyed by such acting is truly amazing.	Positive words → Negative sarcasm	Negative
3	সিনেমাটা একেবারে খারাপ না, শুধু গল্প নেই, অভিনয় নেই, আর দেখার মতো কিছুই নেই।	The movie is not bad at all; it just has no story, no acting, and nothing worth watching.	Negative words → Positive sarcasm	Positive
4	ভয়ংকর সিনেমা! এত ভালো লেগেছে যে শেষ হওয়ার পরও বারবার দেখতে ইচ্ছে করছে।	A terrible movie! I liked it so much that I want to watch it again and again after it ended.	Negative words → Positive sarcasm	Positive
5	এত বাজে সিনেমা জীবনে দেখিনি, তাই সবাইকে অবশ্যই দেখার পরামর্শ দেব।	I have never seen such a bad movie in my life, so I would definitely recommend everyone to watch it.	Negative words → Positive sarcasm	Positive

After completing the annotation process, the six independent labels assigned to each review were aggregated using a majority voting procedure. A review was assigned a final sentiment label only when either

the positive or negative category received a clear majority of votes. Reviews receiving an equal number of Positive and Negative votes (3–3) were considered tied or ambiguous because no unique majority label could be determined. A total 500 such reviews were excluded from the final modeling dataset.

Consequently, of the 22,789 independently annotated reviews, a total of 500 tied or ambiguous reviews were removed, resulting in a final dataset of 22,289 reviews used for model development and evaluation.

3.2.4 Inter-Annotator Agreement Analysis

Inter-annotator agreement was evaluated using the independent sentiment labels assigned by the six annotators to the 22,789 reviews before the removal of tied instances. Fleiss' Kappa was used to measure the overall level of agreement among all six annotators, whereas pairwise Cohen's Kappa was used to examine agreement between individual annotator pairs.

The overall agreement of all six annotators and the pairwise agreement matrix are shown in Fig. 3. The overall Fleiss' Kappa score was 0.1628, which is a slight agreement based on the Landis and Koch interpretation guideline. Likewise, the pairwise Cohen's Kappa values were in the range of about 0.15 to 0.175, which correlates to similar agreement among the different annotator pairs. The agreement results suggest significant differences in the scores given by individual annotators, as user-generated movie reviews are subjective and may include informal writing styles, abbreviations, contextual expressions, and ambiguous opinions. For this reason, majority voting was used to label the data when there was a clear majority, and data with no clear majority was not included in the final model dataset. The results of the agreement and the majority voting are taken to be part of the construction of the dataset and not as a reflection of high inter-annotator agreement.

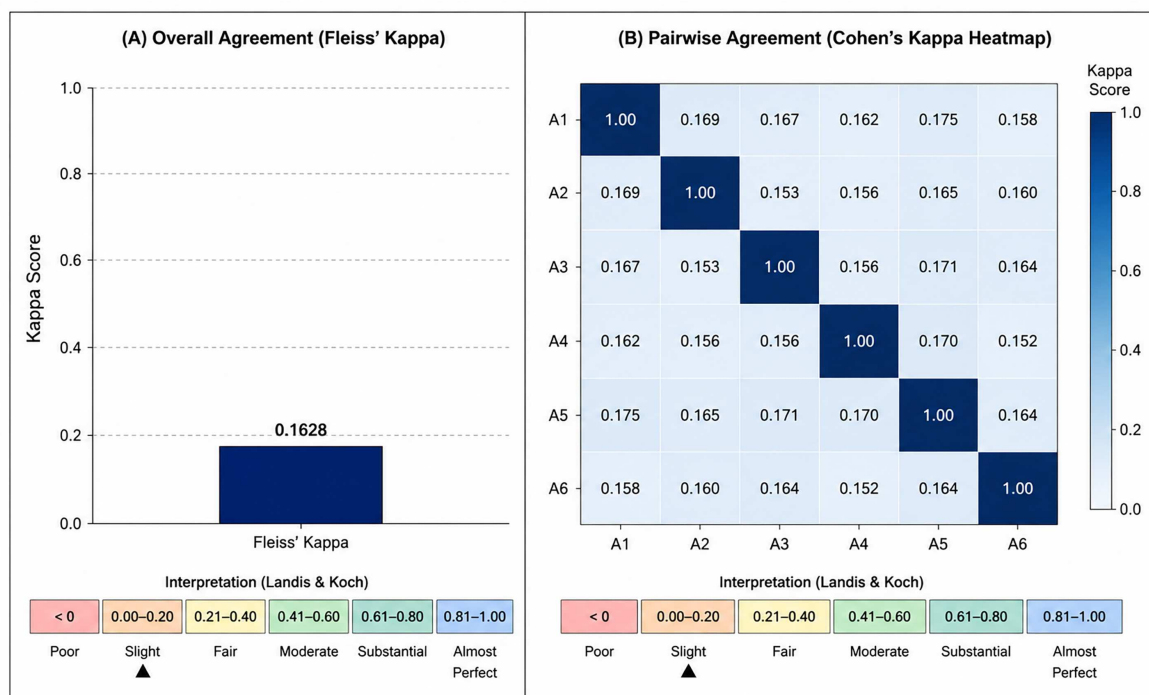


Figure 3: Inter-annotator agreement measured using Fleiss' Kappa and pairwise Cohen's Kappa. (A) Overall agreement across all six annotators, measured by Fleiss' Kappa (score = 0.1628), plotted against the Landis and Koch interpretation bands. (B) Pairwise agreement between each pair of annotators (A1–A6), measured by Cohen's Kappa and displayed as a heatmap (diagonal = 1.00, each annotator's agreement with itself).

Therefore, majority voting was used as a label aggregation mechanism to obtain a single label when a clear majority existed, while 3–3 tied instances were excluded because they lacked a unique consensus label. Given the observed level of inter-annotator agreement, the majority-voting procedure is not interpreted as evidence of high annotation reliability; rather, it provides a transparent rule for aggregating multiple independent annotations and excluding instances for which no majority decision was available.

3.2.5 Tied Data Removal

Inter-annotator agreement was evaluated using Fleiss' Kappa and pairwise Cohen's Kappa based on the six independent sentiment annotations of the 22,789 reviews. The quality-control decisions made before sentiment annotation were not included in this analysis. Following annotation, majority voting was used to assign a final sentiment label when a clear majority was present. Reviews receiving three positive and three negative votes were classified as tied because no unique majority label could be assigned. A total of 500 such reviews were excluded from the modeling dataset.

The exclusion of tied reviews was used as a predefined label-selection criterion rather than as a claim of improved inter-annotator reliability. Consequently, the final dataset contained 22,289 reviews for model development and evaluation. The complete filtering process is shown in Fig. 4, and the final class distribution is reported in Table 4.

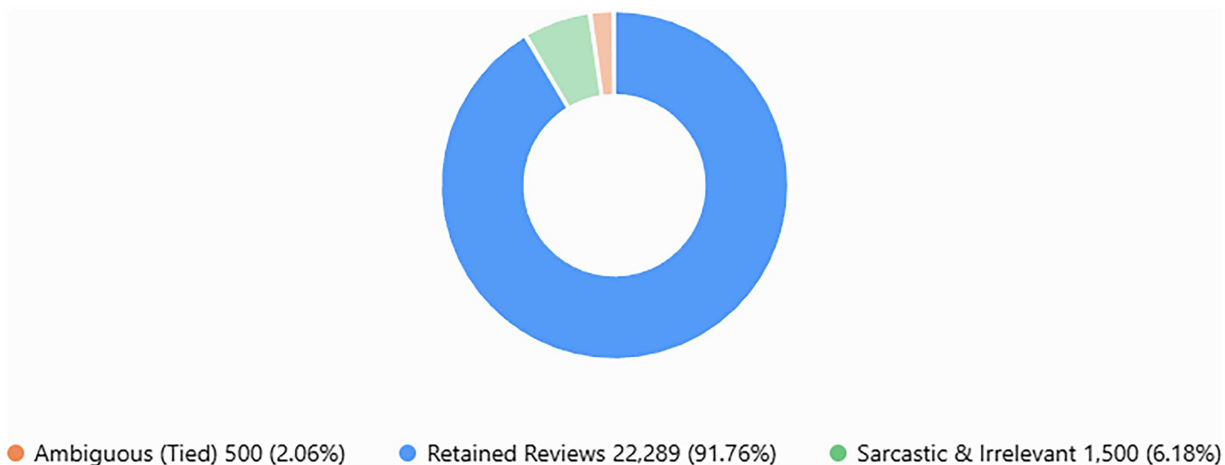


Figure 4: Dataset filtering process. Of the 24,289 initially collected reviews, 1500 were removed during quality screening and a total of 500 tied reviews were subsequently excluded, resulting in 22,289 reviews for model development and evaluation.

Table 4: Dataset construction and final class distribution.

Dataset Stage	Number of Reviews	Description
Initially collected reviews	24,289	YouTube movie reviews collected for the study
Removed during quality screening	1500	Sarcastic/ironic, spam, advertisements, duplicates, incomplete, and other irrelevant reviews
Reviews retained for annotation	22,789	Reviews remaining after manual quality screening

(Continued)

Table 4 (continued)

Dataset Stage	Number of Reviews	Description
Independently annotated reviews	22,789	Each review is independently annotated by six native Bangla speakers
Tied/ambiguous reviews removed	500	Reviews receiving a 3–3 Positive–Negative annotation split
Final modeling dataset	22,289	Reviews retained after tied/ambiguous cases were excluded
Final Class Distribution		
Positive	11,146	Final Positive Reviews
Negative	11,143	Final Negative Reviews
Total	22,289	Final modeling dataset

Table 5 shows five representative tied cases. For example, the first review “এই মুভি টা এমন কেন?” (Why is this movie like this?), received positive labels from A1, A2, and A4 and negative labels from A3, A5, and A6 annotators. Since the six annotations resulted in a 3–3 split, no unique majority label could be assigned, and the review was excluded. The same criterion was applied to all tied cases. A total of 500 reviews were removed for this reason, resulting in the final dataset of 22,289 reviews.

Table 5: Examples of tied annotations removed during dataset preprocessing.

Review	A1	A2	A3	A4	A5	A6	Decision
এই মুভি টা এমন কেন? (English: “Why is this movie like this?”)	P	P	N	P	N	N	Removed (3–3 Tie)
গল্পটা ঠিক ছিল, তবে আরও ভালো হতে পারত (English: “The story was okay, but it could have been better.”)	P	N	P	N	P	N	Removed (3–3 Tie)
অভিনয় ভালো, কিন্তু শেষটা হতাশাজনক (English: “The acting was good, but the ending was disappointing.”)	P	N	P	N	N	P	Removed (3–3 Tie)
দেখতে খারাপ লাগেনি, আবার খুব ভালোও না (English: “It wasn’t bad to watch, but it wasn’t very good either.”)	N	P	N	P	P	N	Removed (3–3 Tie)
একবার দেখা যায়, তবে মনে রাখার মতো নয় (English: “It is watchable once, but not memorable.”)	P	N	N	P	P	N	Removed (3–3 Tie)

3.3 Zipf’s Law

The lexical properties of the final corpus were examined in terms of Zipf’s law, which states that the frequency of a token is inversely proportional to its rank in a corpus [16]. Let t be the frequency of a token and g be its rank. The mathematical expression is given in Eq. (1). The token frequency distribution is shown in Fig. 5. The token frequency distribution follows an approximately Zipfian pattern, consistent with what is

typically observed in natural-language corpora, though with some deviation from a pure power law at the head and tail of the rank distribution.

$$t \propto \frac{1}{g} \quad (1)$$

The final dataset of 22,289 reviews was split into training (80%) and testing (20%) groups using the holdout evaluation approach.

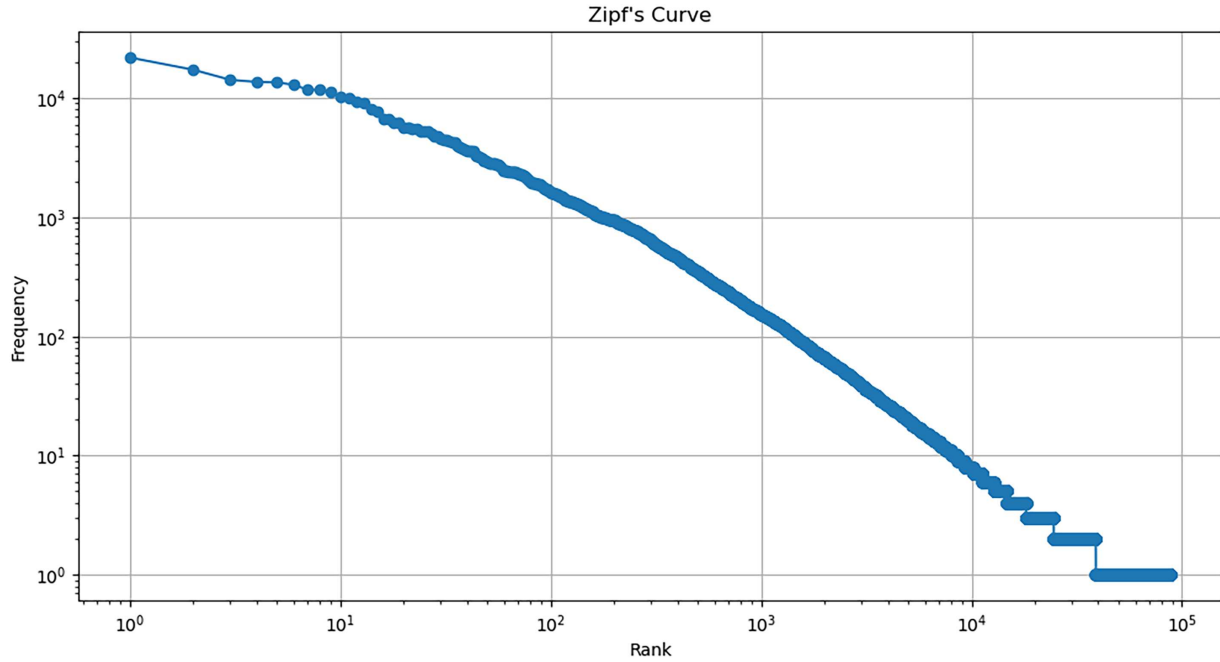


Figure 5: Zipf's curve of the dataset.

3.4 Data Preprocessing

User-generated reviews will introduce irrelevant information and noise into the dataset, hampering the learning models' ability to perform classification properly. So, preprocessing is necessary before applying the classification algorithms. This subsection demonstrates several preprocessing steps, including punctuation removal, emoji and URL removal, tokenization, stopwords removal, and stemming.

3.4.1 Removal of Punctuation, Emojis, and URLs

Frequently used punctuation marks in the Bangla language like “|”, “;”, “?”, “:”, “!” etc. are removed from the dataset initially. Emojis are a common way for users to express themselves on internet platforms; they do not provide the correct information in most cases, which may introduce noise that impairs classification performance. To make the text easier for the model to process, eliminating emojis is a crucial step. This study removed URLs to reduce noise in the corpus. For example, the sample Bangla movie review “এই মুভিটা জবরদস্ত ছিল রে ভাই, মনটা চাইতেছে পড়াশুনা বাদ দিয়ে অভিনয় করি কিন্তু আম্মুকে কিভাবে বুঝাবো !!! @sakib দোস্ত চল অভিনয় করবি” (this movie was awesome brother, my heart wants to leave my studies and act but how will i explain to my mother!!! @sakib dude let's act) after the removal of punctuation, emojis and URLs become “এই মুভিটা জবরদস্ত ছিল রে ভাই মনটা চাইতেছে পড়াশুনা বাদ দিয়ে অভিনয় করি কিন্তু আম্মুকে কিভাবে বুঝাবো দোস্ত চল অভিনয় করবি”.

3.4.2 Tokenization

Tokenization generates tokens from the film reviews collected after eliminating punctuation, emojis, and URLs from the dataset. This approach allows the text to be fragmented into its most fundamental components. For example, the sample Bangla movie review “খুবই জঘন্য ও নিম্নমানের মুভি” (very bad and low-quality movie) after the tokenization produces the tokens as “খুবই”, “জঘন্য”, “ও”, “নিম্নমানের” and “মুভি”.

3.4.3 Stop Words Removal

Stop words are frequently occurring function words in a dataset that provide little or no significant information related to the specific problem domain. The removal of stop words reduces the dimensionality of the text data and enables machine learning models to focus on more meaningful and informative features, which can improve computational efficiency and classification performance. In this study, the 35% most frequent words in the corpus were considered stop words [17] and removed during the preprocessing phase. The study [17] was conducted for Bangla drama and TV series reviews based on sentiment analysis, which is very close to our study movie reviews based on sentiment analysis. So, we have decided to use the top 35% words as the final stop words list. The most frequent stop words identified in our dataset include “এবং” (and), “ও” (and), “তুমি” (you), “যদি” (if), and “হয়ে” (become), among others. These phrases largely serve grammatical purposes and provide little semantic value for sentiment analysis. Therefore, they were removed from the dataset to generate a clean corpus with more relevant and contextually rich language for efficient sentiment analysis.

3.4.4 Stemming

Stemming is the process of converting words into their morphological root forms. It focuses on the specific root forms, excluding suffixes rather than the underlying meaning of words. The Bangla words “মুভিটি”, “মুভিটির”, “মুভিটা”, “মুভিগুলো”, “মুভিগুলো” all contain the same root form “মুভি” (movie), so converting all of these five words to the root form will make it much easier for the learning models to work and process further. We have used the *bangla-stemmer* package (version 1.0) [18], developed by Protick and distributed via the Python Package Index (PyPI), to perform stemming in the proposed dataset. Stemming was applied at the word level using *BanglaStemmer().stem(word)* from the *bangla_stemmer.stemmer.stemmer* module. The stemmer implements a rule-based algorithm originally developed by Kamal [19], with modifications introduced by the package author to improve compatibility across different natural language processing pipelines.

3.5 Feature Extraction

Directly dealing with text data is not straightforward for the learning models and requires the conversion of the textual data to meaningful numerical features that capture the meaning and context of the textual data; methods that can solve this problem are called feature extraction. The essential features were extracted by two methods: the Term Frequency-Inverse Document Frequency (TF-IDF) and word embedding (Word2Vec).

3.5.1 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is one of the most popular feature extraction methods in text mining and natural language processing (NLP). It assigns a weight for each term in a document depending on the frequency of the term in the document and the rarity of the term in all the documents.

The term frequency of a term t in a movie review d is calculated as:

$$TF(t, d) = \frac{\text{Number of occurrences of term } t \text{ in review } d}{\text{Total number of tokens in review } d} \quad (2)$$

The inverse document frequency is calculated as:

$$IDF(t) = \log_e \left(\frac{N}{DF(t)} \right) \quad (3)$$

where N denotes the total number of reviews in the corpus and $DF(t)$ represents the number of reviews containing the term t .

Finally, the TF-IDF weight of term t in review d is computed as the product of term frequency and inverse document frequency:

$$TF\text{-}IDF(t, d) = TF(t, d) \times IDF(t) \quad (4)$$

This study fixed the parameters of the TF-IDF model at $ngram_range = (1, 3)$ and $max_features = 1000$, values chosen based on common practice in prior Bangla sentiment analysis literature rather than through a systematic hyperparameter search. Therefore, unigram, bigram, and trigram features were extracted, and the top 1000 features were retained according to the TF-IDF vectorization procedure for the proposed dataset.

3.5.2 Word Embedding

Word embeddings are a technique in natural language processing (NLP) that involves encoding words as vectors [20,21]. Lexical mapping refers to the technique of representing words in a multidimensional space. This mapping is then converted to a vectorial representation of indices. Keras is an embedding package that we utilized during our evaluation to include word embeddings within the models designed by the neural network. This is a very prominent feature of this package, as it offers a pre-made word embedding layer. We used the *Keras* tokenizer to preprocess the data and to convert our text data to integer sequences. The embedding layer was then used to map the integer indices to dense vectors of real numbers. Each of the *Tokenizer*, *pad_sequences*, and *Embedding* methods are examined in turn to obtain vectors of actual values for the textual inputs, and the best parameters for each of these methods are listed in Table 6.

Table 6: Parameters for the methods of word embedding.

Method	Parameter	Measurement/Value
Tokenizer	num_words	5000
	oov_token	UNK
pad_sequences	padding	post
	maxlen	1000
Embedding	input_dim	vocab_size
	output_dim	50
	input_length	1000

3.6 Model Training and Evaluation Strategy

Two complementary validation methods were used to provide a reliable and unbiased evaluation of the proposed models. The information was first split into an 80/20 holdout test set, where 80% was used to train the model and 20% of the information was used for autonomous testing. This separation permits a somewhat fair evaluation of the model's ability to generalize to new data. The tenfold cross-validation technique was additionally used to increase the validity of the evaluation. The training data was divided into 10 equal parts; in each iteration, 9 were used for training and 1 for validation. This was done 10 times to make sure that all samples were used for training and validation. The final results were obtained from the average performance of all folds, which eliminates any potential bias and fluctuation.

The dataset is well-balanced, yet there is some class imbalance within each training fold after splitting the data. The partitioning for the holdout train-test split and 10-fold cross-validation was done at the review level using scikit-learn's `train_test_split` and `StratifiedKFold` functions with fixed random seeds, rather than by movie. So it is possible to have assessments of the same movie in the training and test (or validation) sets, with no limits on grouping movies at the level of the partitioning.

3.7 Design Algorithms

The most critical—if not essential—component of a classification system is the design of the algorithms themselves [22]. Proper algorithm selection, optimization, and design are the key to the success of a classifier. In this paper, we used some machine learning (ML) algorithms, deep learning (DL) algorithms, BERT-based transfer learning (TF) algorithms, and a stacking ensemble model. For model training, and out-of-fold (OOF) predictions were created through 10-fold cross-validation. These out-of-fold predictions, expressed as class probabilities rather than hard labels (see [Section 3.7.4](#)), constitute the rows of the meta-feature matrix used to train the meta-classifier. The base models were trained with 9 folds of the training set and tested on the 10th fold of the training set, which was not used to train the models. The predictions generated from all the 10 held out folds were then aggregated to form the meta-feature matrix, which was fed to train the meta-classifier (Multi-Layer Perceptron). The same fold assignments were used for all the 10 base learners to ensure a valid stacking architecture.

In order to avoid overfitting, early stopping was used for all deep learning models, and the proposed stacking model by monitoring the validation loss was used. The training was stopped after validation loss did not decrease for 5 epochs (*patience* = 5, *monitor* = 'val_loss'). We noticed from preliminary experiments that most deep learning models performed best within 500 epochs, and that performance started to decline beyond this point. Thus, the maximum number of epochs was fixed at 500 for all deep learning models, based on these preliminary observations rather than a formal hyperparameter tuning search.

All experiments were repeated five times with different random seeds (42, 52, 62, 72 and 82) to assess the robustness and stability of the proposed framework. For every random seed, the entire training and evaluation process, such as data partitioning, model and initialization, and optimization, was repeated with the same hyperparameter configurations. The performance metrics reported in this paper are the average over five of the experimental runs, minimizing the effect of random initialization and data partitioning on the experimental results. Finally, the stacking ensemble model was compared against the second-best individual model using a one-sided Wilcoxon signed-rank test was performed across five random seeds, and the model families (ML, DL, BERT-based TL) were compared using fold-level Friedman tests with Holm-corrected pairwise post-hoc tests; results are reported in [Section 4.2](#).

3.7.1 Machine Learning Algorithms

The sentiment analysis of the Bangla movie reviews has been conducted using the ML models: BNB, SVM, KNN, LR, MNB, and RF. BNB applies Bayes' theorem [observe (5)] to compute the probability of a feature appearing in each class [23].

$$\text{Bayes Theorem, } P(R_i|S) = \frac{P(S|R_i)P(R_i)}{\sum_{i=1}^m P(R_i)P(S|R_i)} \quad (5)$$

In (5), $P(R|S)$ states the probability of appearance of R given S has occurred, $P(R)$ denotes the prior probability, $P(S)$ demonstrates the probability of appearance of action S , $P(S|R)$ is the probability of appearance of S given R has already occurred. The SVM is well recognized for its efficient performance in pattern recognition and other classification problems. It functions by determining the optimal hyperplane of features that maximizes the margin of the class mean [2]. The assignment of a new data point by KNN is contingent upon the class of its k -nearest neighbors [12] and this study uses the Euclidean distance [observe (6)] to find the nearest neighbors.

$$d(r, s) = \sqrt{\sum_{i=1}^p (s_i r_i)^2} \quad (6)$$

Using the Bayes theorem [observe (5)], MNB is a probabilistic classifier that optimizes its effectiveness by producing target labels with the greatest likelihood [4]. The Random Forest algorithm improves accuracy and knowledge by generating multiple decision trees during training and aggregating the class mode of each tree to obtain the final classification prediction. Table 7 briefly describes the fixed parameters of the implemented ML algorithms.

Table 7: Parameters of the implemented ML algorithms.

Algorithm	Parameter	Measurement/Value
BNB	alpha	1.0
SVM	kernel	linear
	C (regularization parameter)	1.0
	gamma	scale
	degree	3
	random_state	42
KNN	n_neighbors	5
	weights	uniform
	leaf_size	30
	p (Power parameter for the Minkowski metric)	2
	algorithm	auto
LR	C (Regularization parameter)	1.0
	random_state	42
MNB	alpha	1.0

(Continued)

Table 7 (continued)

Algorithm	Parameter	Measurement/Value
RF	n_estimators	100
	criterion	gini
	min_samples_split	2
	min_samples_leaf	1
	max_features	auto
	random_state	42

3.7.2 Deep Learning Algorithms

Deep learning (DL) models have been shown to be effective compared to the traditional machine learning (ML) algorithms recently and have been slower to train. For this work the authors have employed the Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), Gated Recurrent Unit (GRU), Bidirectional Gated Recurrent Unit (Bi-GRU), and the hybrid models CNN-LSTM, CNN-GRU, CNN-BiLSTM, and CNN-BiGRU. CNNs, which involve convolutional and pooling layers, have been found to achieve a very high accuracy in document categorization tasks [24]. An RNN is an algorithm that takes in the data one piece at a time, and keeps a context or state to find and determine the temporal dependencies.

$$RNN : h_t^l, h_{t_1}^l \rightarrow h_t^l \quad (7)$$

The RNN variant LSTM uses memory cells and gates to keep long-term dependencies and sequentially process input sequences to effectively mitigate the vanishing gradient problem. It is widely used in natural language processing [25].

$$LSTM : h_t^l, h_{t_1}^l, c_{t_1}^l \rightarrow h_t^l, c_t^l \quad (8)$$

The Bi-LSTM architecture extends the basic LSTM by including the past and future context information, enabling the analysis of input sequences in both the forward and backward directions [26]. The GRU is based on the LSTM architecture with gating mechanisms to control information flow, which improves gradient propagation and training efficiency [27].

$$GRU : h_t^l, h_{t_1}^l, r_t^l, z_t^l, z_{t_1}^l \rightarrow h_t^l, c_t^l \quad (9)$$

The Bi-GRU algorithm enhances GRU by analyzing input sequences in both forward and backward orientations, incorporating context information from future and previous states [28]. These deep learning models consist of an embedding layer that transforms input data into a more rigorous format, followed by layers that extract information become a processing layer, synthesize features for predictions, and a fully connected output layer. This framework helps the model make sense of past data and predict the future. The parameters of the implemented deep learning algorithms are shown in Tables 8–10, respectively.

Table 8: Parameters of the DL algorithms (part 1).

Algorithm	Parameter	Measurement/Value
CNN	max_features	5000
	embedding_dim	50
	conv1d_filters	128
	kernel_size	5
	pool_size	2
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'
RNN	max_features	5000
	embedding_dim	50
	rnn_units (SimpleRNN)	128
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'
LSTM	max_features	5000
	embedding_dim	50
	lstm_units	128
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'

(Continued)

Table 8 (continued)

Algorithm	Parameter	Measurement/Value
Bi-LSTM	max_features	5000
	embedding_dim	50
	lstm_units (per direction)	64
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'

Table 9: Parameters of the DL algorithms (part 2).

Algorithm	Parameter	Measurement/Value
GRU	max_features	5000
	embedding_dim	50
	gru_units	64
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch	500
	early stopping	patience = 5, monitor = 'val_loss'
Bi-GRU	max_features	5000
	embedding_dim	50
	gru_units (per direction)	64
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam

(Continued)

Table 9 (continued)

Algorithm	Parameter	Measurement/Value
CNN-LSTM	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'
	max_features	5000
	embedding_dim	50
	conv1d_1_filters	128
	conv1d_2_filters	64
	kernel_size (both Conv1D)	5
	pool_size	2 (after Conv1D_1 only)
	lstm_units	64
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'

Table 10: Parameters of the DL algorithms (part 3).

Algorithm	Parameter	Measurement/Value
CNN-GRU	max_features	5000
	embedding_dim	50
	conv1d_1_filters	32
	conv1d_2_filters	64
	kernel_size (both Conv1D)	5
	pool_size	2 (after Conv1D_1 only)
	gru_units	64
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'

(Continued)

Table 10 (continued)

Algorithm	Parameter	Measurement/Value
CNN-BiLSTM	max_features	5000
	embedding_dim	50
	conv1d_1_filters	128
	conv1d_2_filters	32
	kernel_size (both Conv1D)	5
	pool_size	2 (after Conv1D_1 only)
	lstm_units (per direction)	32
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
CNN-BiGRU	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'
	max_features	5000
	embedding_dim	50
	conv1d_1_filters	64
	conv1d_2_filters	64
	kernel_size (both Conv1D)	5
	pool_size	2 (after Conv1D_1 only)
	gru_units (per direction)	64
	dropout	0.5
	dense_units	64
	dense_activation	ReLU
	output_units	2
	output_activation	Softmax
	losses	CategoricalCrossentropy
	optimizer	Adam
	batch_size	32
	epoch (max)	500
	early stopping	patience = 5, monitor = 'val_loss'

While [Tables 8–10](#) report the hyperparameter values for each deep learning model, they do not by themselves specify the order in which layers are connected or how filter and unit counts map onto specific layers. To address this, [Table 11](#) presents the complete layer-by-layer architecture of each of the ten deep learning models, listing every layer in the exact sequence in which it is applied, from the input embedding layer to the final classification layer. For all ten models, the input is first mapped through an Embedding layer (vocabulary size 5000, embedding dimension 50, input length 1000 tokens). For the convolutional and

hybrid models (CNN, CNN-LSTM, CNN-GRU, CNN-BiLSTM, CNN-BiGRU), this is followed by one or two Conv1D layers with ReLU activation, each paired with a MaxPooling1D layer (pool size 2) applied only after the first convolutional block in the two-convolution hybrids. The recurrent component—SimpleRNN, LSTM, Bidirectional LSTM, GRU, or Bidirectional GRU—is then applied with the unit counts specified in [Tables 8–10](#) (halved per direction for the bidirectional variants to preserve a comparable total parameter budget). A Dropout layer (rate 0.5) follows the recurrent or pooling stage to mitigate overfitting, after which a Dense layer (64 units, ReLU activation) performs non-linear feature synthesis, followed by a second Dropout layer (rate 0.5). Every model terminates in the same output layer: a Dense layer with 2 units and Softmax activation, producing class probabilities for the Positive and Negative sentiment categories. This uniform output-layer design, combined with the shared embedding configuration, ensures that performance differences across models in [Section 4](#) are attributable to the intermediate feature-extraction layers rather than to inconsistencies in input representation or classification head.

Table 11: Layer-by-layer architectures of the implemented deep learning models.

Model	Layer Sequence
CNN	Embedding(5000, 50, input_length=1000) → Conv1D(filters=128, kernel_size=5, activation=ReLU) → MaxPooling1D(pool_size=2) → Dropout(0.5) → GlobalMaxPooling1D() → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
RNN	Embedding(5000, 50, input_length=1000) → SimpleRNN(128) → Dropout(0.5) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
LSTM	Embedding(5000, 50, input_length=1000) → LSTM(128) → Dropout(0.5) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
Bi-LSTM	Embedding(5000, 50, input_length=1000) → Bidirectional(LSTM(64)) → Dropout(0.5) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
GRU	Embedding(5000, 50, input_length=1000) → GRU(64) → Dropout(0.5) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
Bi-GRU	Embedding(5000, 50, input_length=1000) → Bidirectional(GRU(64)) → Dropout(0.5) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
CNN-LSTM	Embedding(5000, 50, input_length=1000) → Conv1D(128, 5, ReLU) → MaxPooling1D(2) → Conv1D(64, 5, ReLU) → Dropout(0.5) → LSTM(64) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
CNN-GRU	Embedding(5000, 50, input_length=1000) → Conv1D(32, 5, ReLU) → MaxPooling1D(2) → Conv1D(64, 5, ReLU) → Dropout(0.5) → GRU(64) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
CNN-BiLSTM	Embedding(5000, 50, input_length=1000) → Conv1D(128, 5, ReLU) → MaxPooling1D(2) → Conv1D(32, 5, ReLU) → Dropout(0.5) → Bidirectional(LSTM(32)) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)
CNN-BiGRU	Embedding(5000, 50, input_length=1000) → Conv1D(64, 5, ReLU) → MaxPooling1D(2) → Conv1D(64, 5, ReLU) → Dropout(0.5) → Bidirectional(GRU(64)) → Dense(64, ReLU) → Dropout(0.5) → Dense(2, Softmax)

3.7.3 BERT-Cased Transfer Learning Algorithms

Prior to adapting to natural language understanding tasks, the BERT model is trained on large amounts of text data to create proper word embeddings [29,30]. We have already implemented some BERT-based TL algorithms like BERT, Bangla-BERT, mBERT, Distil-BERT etc. The biggest contribution of BERT is the use of a deep bidirectional transformer encoder pre-trained using a masked language model (MLM) objective, as opposed to previous shallowly bidirectional or unidirectional language models, which allows conditioning on both the left and right contexts. It was trained using a large text corpus from the English Wikipedia articles.

In contrast, Bangla-BERT is a monolingual BERT model trained with the BanglaLM corpus that is specially designed for the Bangla language [31]. In the mBERT model, the traditional BERT model is enhanced by training many corpora of texts in 104 languages. At the same time, Distil-BERT is a simplified and efficient version of BERT designed for smaller and faster production [32]. Table 12 demonstrates the fixed hyperparameter configuration used for the implemented BERT-based transfer learning algorithms.

Table 12: Parameters and implementation details of the BERT-based transfer learning algorithms.

Algorithm	Parameter	Measurement/Value
BERT	Checkpoint	bert-base-uncased
	Tokenizer	Matching WordPiece tokenizer (uncased, English vocabulary)
	Architecture	12-layer, 768-hidden, 12-head Transformer encoder
	Classification head	Linear layer on pooled [CLS] representation, num_labels = 2
	Output activation	Softmax (applied post-hoc for probability outputs)
	Loss function	Cross-entropy
	Truncation	True
	Max Length	256
	Padding	Dynamic, per-batch (DataCollatorWithPadding)
	Optimizer	AdamW
	Learning Rate (lr)	0.00001
	Weight Decay	0.01
	LR Scheduler	Linear decay, no warmup
	Mixed Precision	fp16 (enabled if GPU available)
	Batch Size (Train)	4
	Batch Size (Test)	4
	Epochs	10
	Checkpoint Selection	Lowest eval_loss per epoch (load_best_model_at_end=True)
	Cross-Validation	Stratified 10-fold (StratifiedKFold, shuffle = True)
	Random Seeds	42, 52, 62, 72, 82
mBERT	Checkpoint	bert-base-multilingual-cased
	Tokenizer	Matching WordPiece tokenizer (cased, 104-language vocabulary)
	Architecture	12-layer, 768-hidden, 12-head Transformer encoder
	Classification head	Linear layer on pooled [CLS] representation, num_labels = 2
	Output activation	Softmax (applied post-hoc for probability outputs)
	Loss function	Cross-entropy
	Truncation	True
	Max Length	256
	Padding	Dynamic, per-batch (DataCollatorWithPadding)
	Optimizer	AdamW
	Learning Rate (lr)	0.00001
	Weight Decay	0.01

(Continued)

Table 12 (continued)

Algorithm	Parameter	Measurement/Value
mBERT	LR Scheduler	Linear decay, no warmup
	Mixed Precision	fp16 (enabled if GPU available)
	Batch Size (Train)	4
	Batch Size (Test)	4
	Epochs	10
	Checkpoint Selection	Lowest <code>eval_loss</code> per epoch (<code>load_best_model_at_end=True</code>)
	Cross-Validation	Stratified 10-fold (<code>StratifiedKFold</code> , <code>shuffle = True</code>)
	Random Seeds	42, 52, 62, 72, 82
Bangla-BERT	Checkpoint	<code>sagorsarker/bangla-bert-base</code>
	Tokenizer	Matching WordPiece tokenizer trained on the BanglaLM corpus
	Architecture	12-layer, 768-hidden, 12-head Transformer encoder (BERT-base config)
	Classification head	Linear layer on pooled [CLS] representation, <code>num_labels = 2</code>
	Output activation	Softmax (applied post-hoc for probability outputs)
	Loss function	Cross-entropy
	Truncation	True
	Max Length	256
	Padding	Dynamic, per-batch (<code>DataCollatorWithPadding</code>)
	Optimizer	AdamW
	Learning Rate (lr)	0.00001
	Weight Decay	0.01
	LR Scheduler	Linear decay, no warmup
	Mixed Precision	fp16 (enabled if GPU available)
	Batch Size (Train)	4
	Batch Size (Test)	4
	Epochs	10
	Checkpoint Selection	Lowest <code>eval_loss</code> per epoch (<code>load_best_model_at_end=True</code>)
	Cross-Validation	Stratified 10-fold (<code>StratifiedKFold</code> , <code>shuffle = True</code>)
	Random Seeds	42, 52, 62, 72, 82
Distil-BERT	Checkpoint	<code>distilbert-base-uncased</code>
	Tokenizer	Matching WordPiece tokenizer (uncased, English vocabulary; shared with BERT-base)
	Architecture	6-layer, 768-hidden, 12-head Transformer encoder
	Classification head	Linear layer on first-token hidden state (no pooler layer), <code>num_labels = 2</code>
	Output activation	Softmax (applied post-hoc for probability outputs)
	Loss function	Cross-entropy
	Truncation	True
	Max Length	256
	Padding	Dynamic, per-batch (<code>DataCollatorWithPadding</code>)

(Continued)

Table 12 (continued)

Algorithm	Parameter	Measurement/Value
Distil-BERT	Optimizer	AdamW
	Learning Rate (lr)	0.00001
	Weight Decay	0.01
	LR Scheduler	Linear decay, no warmup
	Mixed Precision	fp16 (enabled if GPU available)
	Batch Size (Train)	4
	Batch Size (Test)	4
	Epochs	10
	Checkpoint Selection	Lowest <code>eval_loss</code> per epoch (<code>load_best_model_at_end=True</code>)
	Cross-Validation	Stratified 10-fold (<code>StratifiedKFold</code> , <code>shuffle = True</code>)
	Random Seeds	42, 52, 62, 72, 82

Table 12 reports the complete implementation and configuration for each of the four BERT-based models, organized per algorithm to allow direct comparison. For reproducibility, the table specifies the exact Hugging Face model checkpoint and its matching tokenizer used for each model: *bert-base-uncased* for BERT, *bert-base-multilingual-cased* for mBERT, *sagorsarker/bangla-bert-base* for Bangla-BERT, and *distilbert-base-uncased* for Distil-BERT.

It is worth noting that BERT and Distil-BERT use English-vocabulary checkpoints, while only mBERT and Bangla-BERT were pretrained on Bangla or multilingual corpora that natively cover the Bangla script; this distinction affects how each tokenizer segments Bangla input and is discussed further in Section 5. Each model follows the standard Hugging Face *AutoModelForSequenceClassification* design: a linear classification head with two output units is attached on top of the pooled [CLS] representation (or, for Distil-BERT, the first-token hidden state, since DistilBERT has no pooler layer), and the model is trained end-to-end using a cross-entropy loss with class probabilities obtained by applying Softmax to the output logits at evaluation time. All architectural hyperparameters not explicitly overridden (e.g., hidden-layer dropout of 0.1) retain their pretrained default values. Inputs were tokenized with truncation enabled and a maximum sequence length of 256 tokens, while padding was applied dynamically per batch via a *DataCollatorWithPadding*, rather than to a fixed length, to minimize unnecessary computation. All four models were fine-tuned using the *AdamW* optimizer with a learning rate of 0.00001, a weight decay of 0.01, and a linear learning-rate decay schedule with no warmup steps. Training used a batch size of 4 for both the training and evaluation splits, and used mixed-precision (fp16) computation when a GPU was available. After each epoch, the checkpoint with the lowest validation loss was retained (`load_best_model_at_end = True`), rather than always using the final epoch's weights. Model evaluation followed the same stratified 10-fold cross-validation scheme used elsewhere in this study, repeated across five random seeds (42, 52, 62, 72, and 82) to assess robustness, consistent with the reproducibility protocol summarized in Section 4.2.

3.7.4 Stacking Ensemble Model

Stacking is a combination of multiple classifiers in order to create a generalizable classifier: base classifiers are trained on the properties of the dataset, and then their predictions are used to train a meta-classifier, which uses the predictions from the base classifiers to generate predictions [33,34]. Specifically, for each of the ten base learners, the meta-feature contribution per instance consists of the two softmax output probabilities, P(positive) and P(negative), rather than a hard class label; concatenating these across all ten

base learners yields a 20-dimensional meta-feature vector ($10 \text{ learners} \times 2 \text{ class probabilities}$) that serves as the input to the MLP meta-classifier. The architecture of the proposed stacking ensemble model is given in Fig. 6. The architecture illustrates this two-stage process in detail, including the generation of out-of-fold predictions used to construct the meta-feature matrix. It demonstrates the end-to-end procedure used to generate the meta-feature matrix and train the meta-classifier of the proposed stacking ensemble. For a given base learner k , the training data is partitioned into 10 folds; the learner is trained on 9 folds and used to generate predictions on the held-out 10th fold, which was not seen during that learner's training. Repeating this process across all 10 folds yields out-of-fold (OOF) predictions for every training instance. Rather than recording a hard class label, each OOF prediction is stored as the pair of softmax output probabilities, $P(\text{positive})$ and $P(\text{negative})$, produced by the base learner. This entire procedure is repeated independently for each of the 10 base learners (CNN, RNN, LSTM, Bi-LSTM, GRU, Bi-GRU, CNN-LSTM, CNN-GRU, CNN-BiLSTM, and CNN-BiGRU), using the same fold assignment throughout to keep the OOF predictions aligned across learners. The two-probability OOF outputs from all 10 base learners are then concatenated for each training instance, producing a 20-dimensional meta-feature vector ($10 \text{ base learners} \times 2 \text{ class probabilities}$). Stacking these vectors across all N training instances forms an $N \times 20$ meta-feature matrix, which is used to train the Multi-Layer Perceptron (MLP) meta-classifier. The trained meta-classifier then maps this 20-dimensional probability representation to the final sentiment prediction (positive or negative). This design ensures that the meta-classifier learns from calibrated probability estimates of each base learner rather than from potentially information-losing hard labels, and that no base learner's predictions on a given instance were produced by a model that had already seen that instance during training, thereby avoiding information leakage into the meta-feature matrix. The parameters of the stacking ensemble method are given in Table 13.

3.8 Hyperparameter Configuration

Rather than performing an exhaustive, cross-validated hyperparameter search, this study used fixed hyperparameter settings for all implemented models, selected on the basis of values commonly reported in prior sentiment analysis literature and confirmed through informal preliminary trials on the training data. The complete configuration used for each model family is reported in Table 7 (ML), Tables 8–10 (DL), Table 12 (BERT-based transfer learning), and Table 13 (proposed stacking ensemble). The same fixed configuration was applied consistently across all 10-fold cross-validation partitions and across all five random seeds (42, 52, 62, 72, 82), so that any observed performance differences between models reflect architectural differences rather than inconsistencies in hyperparameter selection. Table 13 summarizes the parameters of the proposed stacking ensemble model, organized into two levels. The Level-1 section lists the ten deep learning base learners (CNN, RNN, LSTM, Bi-LSTM, GRU, Bi-GRU, CNN-LSTM, CNN-GRU, CNN-BiLSTM, and CNN-BiGRU), whose individual hyperparameters are detailed in Tables 8–10; the same 10-fold assignment was maintained across all base learners to generate out-of-fold (OOF) predictions. The Level-2 section reports the architecture and training configuration of the Multi-Layer Perceptron (MLP) meta-classifier, which takes the base learner's OOF predictions as input and produces the final sentiment prediction. This includes the network structure (input dimension, hidden layers, activation functions), optimization settings (loss function, optimizer, learning rate), regularization (L2), and training procedure (batch size, maximum epochs, early stopping, and random seeds), ensuring full reproducibility of the meta-classifier.

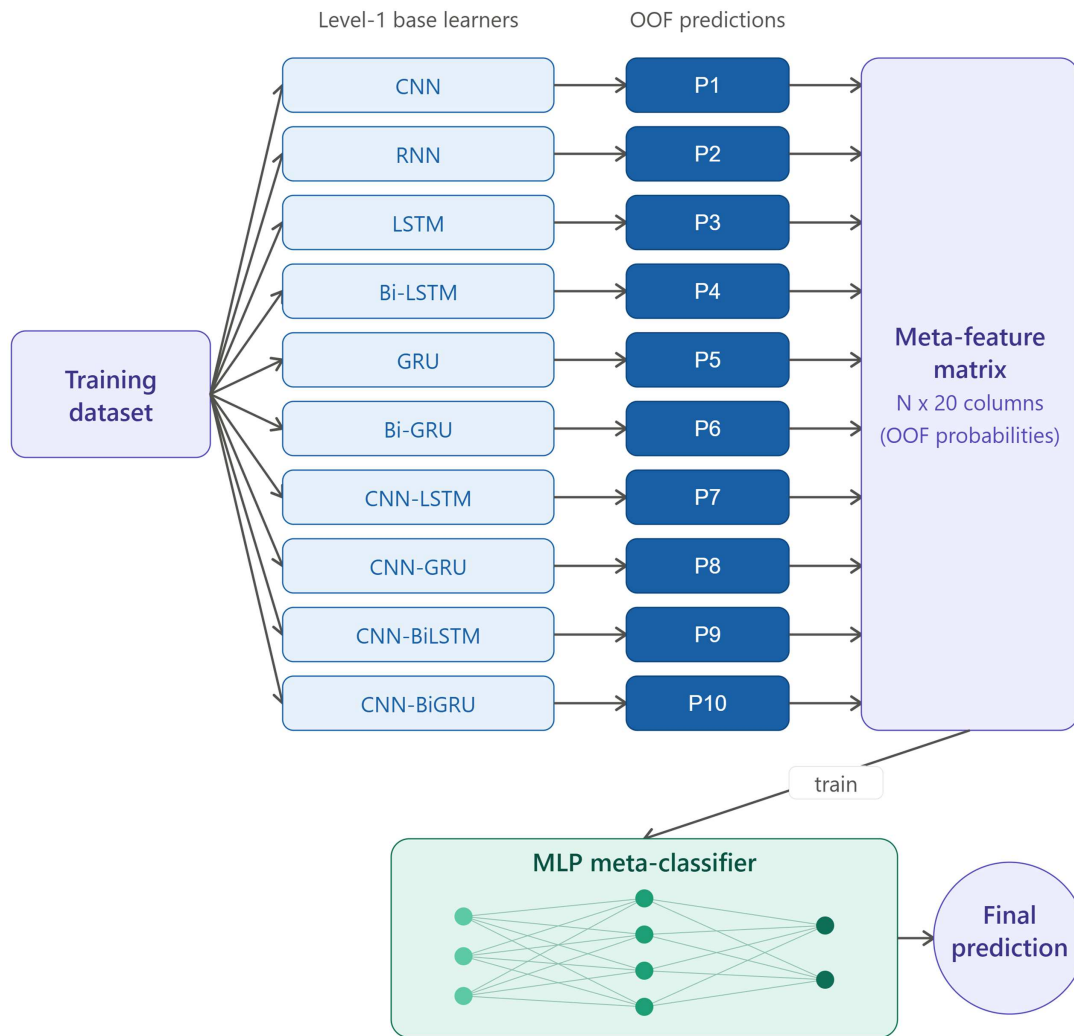
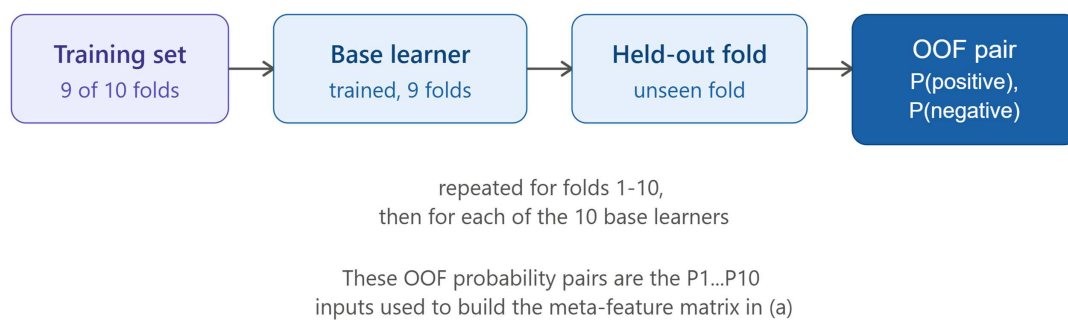
(a) Stacking ensemble architecture**(b) Out-of-fold (OOF) prediction generation, per base learner**

Figure 6: Architecture of the proposed stacking ensemble model. **(a)** Each of the ten base learners is trained via 10-fold cross-validation, and its out-of-fold (OOF) class probabilities P (positive) and P (negative)—not hard labels—are concatenated across all ten learners into a 20-dimensional meta-feature matrix used to train the MLP meta-classifier, which produces the final prediction. **(b)** Detail of the OOF generation process, repeated for every fold and for each of the ten base learners using identical fold assignments.

Table 13: Parameters of the proposed stacking ensemble model (base learners and meta-classifier).

Component	Parameter	Measurement/Value
<i>Level-1: Base Learners</i>		
CNN	Hyperparameters	See Table 8
RNN	Hyperparameters	See Table 8
LSTM	Hyperparameters	See Table 8
Bi-LSTM	Hyperparameters	See Table 8
GRU	Hyperparameters	See Table 9
Bi-GRU	Hyperparameters	See Table 9
CNN-LSTM	Hyperparameters	See Table 9
CNN-GRU	Hyperparameters	See Table 10
CNN-BiLSTM	Hyperparameters	See Table 10
CNN-BiGRU	Hyperparameters	See Table 10
Fold assignment	—	Same 10-fold split maintained across all base learners (OOF predictions)
<i>Level-2: MLP Meta-Classifier (scikit-learn MLPClassifier)</i>		
	Input dimension	20 (P (negative), P (positive) per base learner $\times$ 10 base learners)
	Hidden layers	(64,)—single hidden layer
	Hidden layer activation	ReLU
	Output units	2 (Positive, Negative)
	Output layer activation	Softmax
	Loss function	Log-loss (cross-entropy, internal to MLPClassifier; not user-configurable)
	Optimizer	Adam
	Learning rate	0.001
	Regularization	L2 (alpha = $1e-4$)
	Batch size	auto (scikit-learn default)
	Maximum epochs	500
	Early stopping	early_stopping = True, n_iter_no_change = 5
	Random seeds	42, 52, 62, 72, 82

We acknowledge that an exhaustive search (e.g., grid search or random search) conducted independently within each outer training fold could potentially yield further performance improvements and would provide a stronger guarantee against overfitting than fixed settings alone. Overfitting in this study was instead mitigated through early stopping (monitoring validation loss with patience = 5), dropout regularization in the deep learning models, L2 regularization in the meta-classifier, and evaluation on a held-out test set in addition to 10-fold cross-validation.

4 Experimental Results Analysis and Discussion

The proposed task employs various models to precisely identify sentiments in Bangla movie reviews and explain their reasoning.

4.1 Performance Metrics

The model performance is assessed based on various criteria such as precision, recall, F1-score, accuracy, ROC-AUC, PR-AUC, and kappa, among others, in recent research. The precision and recall are measured using the Eqs. (10) and (11). The recall is actually the true positive rate (TPR), which is defined in (14). It is indicative of how well the model can classify positive instances. In Eqs. (10)–(13), the terms TP , TN , FP , and FN are the number of true positive, true negative, false positive, and false negative predictions, respectively. TP refers to the total number of test instances in which the true class is positive, and the model recognizes it

as also positive; *TN* means the total number of occurrences where the true class is negative, and the model classifies it as negative. The same, *FP* shows the number of such instances that it was determined that the real class was negative, and the model was positive, *FN* shows the number of such instances that it was determined that the real class was positive, but the model was negative. A balanced measure of precision and recall is the F1-score, whose definition is given in (12). The ratio of the number of negative examples that are misclassified is called false positive rate (FPR), defined in (15). Accuracy is defined as the percentage of correctly classified examples of all examples as per (13), which stands for the ratio of correctly classified examples to all examples. The receiver operating characteristics (ROC) curve (16) is only a plot of TPR vs. FPR over all possible thresholds, and the area under the curve (ROC-AUC) is simply the area under all points of this ROC curve. The precision-recall (PR) curve, as specified in (17), is merely a graphical presentation of precision and recall values for all possible thresholds, and the area under the curve (PR-AUC) is the entire area under the PR curve.

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (12)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (13)$$

$$TPR = \frac{TP}{TP + FN} \quad (14)$$

$$FPR = \frac{FP}{TN + FP} \quad (15)$$

$$ROC = TPR \text{ vs. } FPR \quad (16)$$

$$PR = Precision \text{ vs. } Recall \quad (17)$$

4.2 Obtained Results Analysis

As is crucial for every classification problem to obtain reliable and satisfactory results, efficient data preprocessing and a well-considered hyperparameter configuration are important steps in the classification process [35]. Following several common preprocessing methods, model hyperparameters were fixed prior to training, based on values commonly reported in related literature and confirmed through preliminary pilot runs, in order to provide a fair and consistent evaluation of all models (see Tables 7–10, 12 and 13).

The performance of the machine learning algorithms that were actually implemented, with the holdout protocol, is shown in Table 14. Evaluation measures such as precision, recall, F1-score, accuracy, ROC-AUC, PR-AUC, and Kappa were used to evaluate the results. The precision, recall and F1-score were computed for each sentiment class, both positive and negative. BNB and RF had the highest value of 0.86 for the positive class. Regarding the negative class, the precision value was also the highest for RF with a value of 0.86. In recall, RF achieved the highest score of 0.86 in the positive class, while BNB obtained the highest score of 0.89 in the negative class. RF had the highest score for both sentiment classes with the F1-score. Moreover, the overall accuracy, ROC-AUC, PR-AUC, and Kappa value were 86.00%, 0.93, 0.91 and 0.26, respectively, for RF. In general, these results suggest that RF is a better-performing machine learning model among the non-cross-validated models. In tabular form (Table 15), the performance of the machine learning algorithms is presented for 10-fold cross-validation. As anticipated, a clear drop in performance is seen in all the models, compared to the non-cross-validated results, given the more stringent and general evaluation method. The

machine learning models performed best with RF, giving the highest accuracy of 64.11%, and the highest ROC-AUC value of 0.70. From these findings, it can be concluded that the models based on RF are more stable and better generalize than other machine learning models when tested on the cross-validation.

Table 14: Results of the implemented machine learning algorithms using the holdout method.

Model	Sentiment	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
BNB	Positive	0.86	0.69	0.77	79.06	0.88	0.82	0.10
	Negative	0.74	0.89	0.81				
SVM	Positive	0.85	0.77	0.81	81.63	0.92	0.86	0.13
	Negative	0.79	0.87	0.83				
KNN	Positive	0.82	0.70	0.76	77.14	0.84	0.78	0.16
	Negative	0.74	0.84	0.79				
LR	Positive	0.84	0.83	0.83	83.55	0.90	0.85	0.13
	Negative	0.83	0.84	0.83				
MNB	Positive	0.82	0.80	0.81	81.08	0.89	0.84	0.10
	Negative	0.81	0.82	0.81				
RF	Positive	0.86	0.86	0.86	86.00	0.93	0.91	0.26
	Negative	0.86	0.86	0.86				

Table 15: Results of the implemented machine learning algorithms using 10-fold cross-validation.

Model	Sentiment	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
BNB	Negative	0.55	0.52	0.53	55.00	0.57	0.56	0.09
	Positive	0.55	0.58	0.56				
SVM	Negative	0.58	0.50	0.54	58.03	0.62	0.61	0.16
	Positive	0.56	0.64	0.60				
KNN	Negative	0.59	0.58	0.58	59.42	0.63	0.62	0.19
	Positive	0.59	0.59	0.59				
LR	Negative	0.56	0.45	0.50	57.86	0.61	0.60	0.16
	Positive	0.54	0.65	0.59				
MNB	Negative	0.55	0.54	0.54	55.17	0.59	0.58	0.10
	Positive	0.55	0.55	0.55				
RF	Negative	0.61	0.68	0.65	64.11	0.70	0.69	0.28
	Positive	0.64	0.56	0.60				

The machine learning models (Table 16) showed that the highest average accuracy (64.11%) with the lowest standard deviation (0.25) was achieved by the Random Forest (RF) classifier, which presented more consistent performance than the rest of the machine learning models. The average accuracy and standard deviation of KNN, SVM, Logistic Regression (LR), Bernoulli Naïve Bayes (BNB), and Multinomial Naïve Bayes (MNB) were lower, but their performance increased across folds to a certain degree.

Table 16: 10-fold cross-validation accuracy of the implemented machine learning models. Mean and standard deviation (SD) are calculated from the ten fold-level accuracy values.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	Mean	SD
BNB	54.42	55.33	55.12	54.88	55.48	55.16	54.91	55.55	55.04	54.11	55.00	0.45
SVM	57.61	58.42	57.98	58.25	58.31	57.74	58.46	57.89	58.15	57.49	58.03	0.34
KNN	59.11	59.68	59.37	59.82	59.54	59.18	59.75	59.33	59.49	58.93	59.42	0.29
LR	57.42	58.13	57.69	58.27	57.91	57.45	58.34	57.88	58.02	57.49	57.86	0.34
MNB	54.65	55.44	55.22	55.61	55.08	54.83	55.37	55.14	55.52	54.84	55.17	0.32
RF	63.72	64.18	64.35	63.96	64.47	64.02	64.25	63.84	64.39	63.92	64.11	0.25

The deep learning models implemented with holdout protocol are shown in Table 17. As for the standalone deep learning architectures, CNN out-performed the others with an ROC-AUC of 0.95 and an accuracy of 90.31%. CNN-BiLSTM and CNN-BiGRU were competitive in both classes of sentiment analysis in terms of precision, recall and F1-score. Nevertheless, the stacking ensemble model exhibited the best performance among all the deep learning models with accuracy of 94.58%, F1-score of 0.95 under the holdout protocol. This finding is consistent with the performance of ensemble learning systems that increase the classification performance by using multiple architectures of deep learning.

Table 17: Results of the implemented deep learning algorithms using the holdout method.

Model	Sentiment	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
CNN	Positive	0.89	0.88	0.88	90.31	0.95	0.94	0.80
	Negative	0.88	0.89	0.88				
RNN	Positive	0.86	0.92	0.89	89.38	0.94	0.93	0.78
	Negative	0.91	0.85	0.88				
LSTM	Positive	0.88	0.89	0.88	88.02	0.92	0.91	0.76
	Negative	0.89	0.87	0.88				
Bi-LSTM	Positive	0.88	0.90	0.89	88.78	0.94	0.93	0.77
	Negative	0.90	0.87	0.88				
GRU	Positive	0.89	0.88	0.88	88.39	0.94	0.93	0.76
	Negative	0.88	0.89	0.88				
Bi-GRU	Positive	0.87	0.90	0.88	88.74	0.94	0.93	0.77
	Negative	0.90	0.87	0.88				
CNN-LSTM	Positive	0.66	0.58	0.62	60.35	0.71	0.68	0.20
	Negative	0.60	0.61	0.60				
CNN-GRU	Positive	0.88	0.88	0.88	89.78	0.91	0.90	0.79
	Negative	0.89	0.89	0.89				
CNN-BiLSTM	Positive	0.90	0.89	0.89	88.38	0.94	0.93	0.76
	Negative	0.89	0.90	0.89				
CNN-BiGRU	Positive	0.88	0.90	0.89	89.19	0.93	0.92	0.78
	Negative	0.90	0.87	0.88				
Stacking	Positive	0.94	0.96	0.95	94.58	0.96	0.96	0.89
	Negative	0.96	0.94	0.95				

Table 18 shows the test results from deep learning models that are created by using 10-fold cross-validation. Like machine learning models, most deep learning networks suffer from a drop in performance when they are cross-validated. Nevertheless, the generalization ability of the stacking ensemble model was good. The overall best performance was obtained for the proposed stacking ensemble model with an accuracy of 94.91%, and an ROC-AUC value of 0.99 under the 10-fold cross-validation protocol. These results show the stacking model has very high robustness and is not sensitive to the partitioning of the data.

Table 18: Results of the implemented deep learning algorithms using 10-fold cross-validation.

Model	Sentiment	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
CNN	Positive	0.89	0.91	0.90	89.86	0.96	0.95	0.79
	Negative	0.91	0.88	0.89				
RNN	Positive	0.96	0.92	0.94	93.81	0.98	0.97	0.87
	Negative	0.93	0.97	0.95				
LSTM	Positive	0.51	0.60	0.55	51.00	0.50	0.50	0.02
	Negative	0.51	0.41	0.45				
Bi-LSTM	Positive	0.64	0.59	0.61	63.12	0.66	0.65	0.26
	Negative	0.62	0.67	0.64				
GRU	Positive	0.90	0.90	0.90	90.09	0.96	0.95	0.80
	Negative	0.90	0.90	0.90				
Bi-GRU	Positive	0.92	0.94	0.93	93.06	0.98	0.97	0.86
	Negative	0.95	0.91	0.93				
CNN-LSTM	Positive	0.62	0.63	0.62	62.45	0.69	0.67	0.24
	Negative	0.58	0.62	0.60				
CNN-GRU	Positive	0.95	0.81	0.87	88.92	0.95	0.94	0.77
	Negative	0.83	0.94	0.88				
CNN-BiLSTM	Positive	0.67	0.51	0.58	63.02	0.67	0.66	0.26
	Negative	0.61	0.76	0.68				
CNN-BiGRU	Positive	0.62	0.72	0.67	63.43	0.67	0.66	0.26
	Negative	0.66	0.56	0.61				
Stacking	Positive	0.97	0.93	0.95	94.91	0.99	0.98	0.89
	Negative	0.93	0.97	0.95				

The results of the 10-fold cross-validation of the deep learning models are summarized in Table 19. The proposed stacking model achieved the best mean accuracy of 94.91% with a standard deviation of 0.22, which shows an excellent stability of the proposed model and good predictive performance across the validation folds. The individual deep learning models also showed high and consistent performance with low standard deviations where RNN (93.81%) and Bi-GRU (93.06%) demonstrated good performance. GRU and CNN showed competitive results, while LSTM, Bi-LSTM, CNN-LSTM, CNN-BiLSTM, and CNN-BiGRU showed comparatively lower values of accuracy, but that did not affect their performance in the individual folds.

Table 19: 10-fold cross-validation accuracy of the implemented deep learning models. Mean and standard deviation (SD) are calculated from the ten fold-level accuracy values.

Model	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
CNN	89.31	89.74	90.12	89.95	89.63	90.08	89.81	89.58	90.01	90.37	89.86	0.31
RNN	93.34	93.76	94.08	93.92	93.58	94.12	93.86	93.47	94.05	93.92	93.81	0.27
LSTM	50.41	50.96	51.54	51.02	50.67	51.23	50.88	51.39	50.92	50.98	51.00	0.33
Bi-LSTM	62.84	63.11	63.47	63.32	62.96	63.55	63.08	63.24	63.36	62.27	63.12	0.37
GRU	89.63	89.92	90.31	90.18	89.85	90.46	89.94	90.07	90.29	90.25	90.09	0.25
Bi-GRU	92.61	92.94	93.37	93.22	92.88	93.44	93.15	92.96	93.28	92.75	93.06	0.27
CNN-LSTM	61.94	62.36	62.77	62.58	62.19	62.91	62.47	62.66	62.84	61.78	62.45	0.38
CNN-GRU	88.47	88.83	89.21	89.04	88.69	89.34	88.95	89.08	89.27	88.32	88.92	0.34
CNN-BiLSTM	62.58	62.91	63.37	63.18	62.76	63.46	63.05	63.24	63.41	62.24	63.02	0.40
CNN-BiGRU	62.94	63.27	63.71	63.56	63.08	63.79	63.35	63.61	63.83	63.16	63.43	0.31
Stacking	94.52	94.83	95.14	94.96	94.67	95.23	94.88	94.75	95.08	95.05	94.91	0.22

The results of the models implemented using BERT for sentiment classification using without cross-validation is shown in [Table 20](#). The results show that both positive and negative sentiment classes can be achieved with good performance of all models. The overall accuracy of the standard BERT is 90.60%, and the ROC-AUC is 0.91. BERT's recall and F1-score for the positive class is 0.90 and 0.86, respectively, and for the negative class is 0.89 and 0.86, respectively, thus showing good class-wise prediction capacity.

Table 20: Results of implemented BERT-based transfer learning algorithms using the holdout method.

Model	Sentiment	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
BERT	Positive	0.82	0.90	0.86	90.60	0.91	0.89	0.81
	Negative	0.89	0.91	0.90				
mBERT	Positive	0.94	0.90	0.92	92.30	0.95	0.94	0.84
	Negative	0.91	0.94	0.92				
Bangla-BERT	Positive	0.94	0.88	0.91	91.25	0.91	0.90	0.82
	Negative	0.89	0.94	0.91				
Distil-BERT	Positive	0.90	0.90	0.90	89.66	0.75	0.73	0.79
	Negative	0.88	0.92	0.90				

Overall, mBERT outperforms the other BERT models; the highest accuracy of 92.30% is achieved with the mBERT model ([Table 20](#)). It gets F1 for both positive sentiment (0.92) and negative sentiment (0.92), indicating that all positive and negative sentiment scores are correctly identified. The Bangla-BERT also achieves a competitive accuracy of 91.25% and an AUC value of 0.91. However, the efficient Distil-BERT model has comparatively low accuracy (89.66%), and ROC-AUC (0.75) values. Notably, the two lowest-performing models on this holdout evaluation, BERT and Distil-BERT, are also the two checkpoints fine-tuned from English-only vocabularies rather than Bangla or multilingual pretraining corpora, suggesting that part of this gap reflects tokenizer coverage rather than architecture alone (see [Section 5](#)).

[Table 21](#) shows the results achieved with the 10-fold cross-validation approach. Not surprisingly, there is a modest decline in performance on each of the models, owing to the more stringent cross-validation

procedure. Standard BERT model performs well on the folds with an accuracy of 89.40%, a Kappa of 0.78, and an AUC of 0.90. The class-wise F1-scores are unbiased, both the positive sentiment (0.85) and the negative sentiment (0.89).

Table 21: Results of implemented BERT-based transfer learning algorithms using 10-fold cross-validation.

Model	Sentiment	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
BERT	Positive	0.81	0.89	0.85	89.40	0.90	0.88	0.78
	Negative	0.88	0.90	0.89				
mBERT	Positive	0.93	0.89	0.91	91.70	0.89	0.88	0.83
	Negative	0.90	0.93	0.91				
Bangla-BERT	Positive	0.93	0.87	0.90	90.80	0.90	0.89	0.81
	Negative	0.88	0.93	0.90				
Distil-BERT	Positive	0.89	0.89	0.89	88.30	0.74	0.72	0.76
	Negative	0.87	0.91	0.89				

Consistent with the holdout results, mBERT still performs better than the other models when evaluated on cross-validation. It shows the maximum accuracy of 91.70% and a Kappa coefficient of 0.83 indicating the good agreement between the predicted and actual labels. The model has an F1 value of 0.91 on both sentiments, and has a relatively low error value. Moreover, Bangla-BERT is found to be stable and robust under the cross-validation setting with an accuracy of 90.80%. Distil-BERT is also the least-performing, with an accuracy of 88.30%, and a lower ROC-AUC of 0.74. This same ranking persists under cross-validation, reinforcing that the accuracy gap between BERT/Distil-BERT and mBERT/Bangla-BERT is a consistent pattern across both evaluation protocols rather than an artifact of a single train-test split.

The fold-wise evaluation of the BERT-based transfer learning models is given in Table 22. Out of these models, mBERT performed best in terms of average accuracy (91.70%) and has the least standard deviation (0.24) which shows a strong and consistent performance in the cross-validation process. Bangla-BERT also had good classification performance with an average accuracy of 90.80%, followed by BERT (89.40%) and Distil-BERT (88.30%). All transformer-based models have small standard deviations, which indicates good and consistent performance on various data partitions.

Table 22: 10-fold cross-validation accuracy of the implemented BERT-based transfer learning models. Mean and standard deviation (SD) are calculated from the ten fold-level accuracy values.

Model	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
BERT	88.96	89.22	89.58	89.43	89.11	89.75	89.37	89.52	89.68	89.38	89.40	0.25
mBERT	91.26	91.54	91.83	91.72	91.43	92.04	91.66	91.81	91.97	91.74	91.70	0.24
Bangla-BERT	90.35	90.62	90.96	90.83	90.51	91.12	90.74	90.91	91.03	90.93	90.80	0.24
Distil-BERT	87.82	88.11	88.46	88.33	88.04	88.67	88.29	88.42	88.58	88.28	88.30	0.26

The accuracy of the machine learning model, deep learning model, and BERT-based transfer learning model, respectively, from the 10-fold cross-validation experiments is shown in Tables 16, 19 and 22 using a fold-wise basis. The results clearly show that the performance of the models is similar for each fold, highlighting the stability of the evaluation procedure and that the models are not affected by the train-test

split. All of the models have relatively small standard deviation values, indicating that there is low variance and a good ability to generalize.

Tables 23 and 24 show the champion model per family under the holdout and 10-fold cross-validation protocols, respectively. The two protocols are reported separately because holdout and cross-validation estimates are not statistically interchangeable. The precision, recall and F1-scores are evaluated based on the average of positive and negative class values of different algorithms as mentioned in Tables 23 and 24.

Table 23: Champion models per domain under the holdout evaluation protocol.

Domain	Best Model	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
ML	RF	0.86	0.86	0.86	86.00	0.93	0.91	0.26
DL	CNN	0.89	0.89	0.88	90.31	0.95	0.94	0.80
BERT-based TL	mBERT	0.92	0.92	0.92	92.30	0.95	0.94	0.84
Ensemble	Stacking	0.95	0.95	0.95	94.58	0.96	0.96	0.89

Table 24: Champion models per domain under the 10-fold cross-validation protocol.

Domain	Best Model	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC	PR-AUC	Kappa
ML	RF	0.63	0.62	0.63	64.11	0.70	0.69	0.28
DL	RNN	0.94	0.94	0.94	93.81	0.98	0.97	0.87
BERT-based TL	mBERT	0.92	0.91	0.91	91.70	0.89	0.88	0.83
Ensemble	Stacking	0.95	0.95	0.95	94.91	0.99	0.98	0.89

The point estimates in Table 24 suggest that, under 10-fold cross-validation, the DL family outperformed the ML family and the stacking ensemble achieved the highest point-estimate accuracy among all families. Under the holdout protocol (Table 23), the same ranking (ML < DL < BERT-TL < Ensemble) holds, though absolute values and the DL-family champion (CNN rather than RNN) differ. These descriptive comparisons are formally tested below using fold-level Friedman and post-hoc pairwise tests.

To properly test whether individual models differ in classification ability, a non-parametric Friedman test [36] was conducted using fold-level accuracy as the repeated-measures blocks, rather than performance metrics. For each model family, the 10 fold-wise accuracy values reported in Table 16 (ML), Table 19 (DL), and Table 22 (BERT-based TL) were treated as $n = 10$ paired blocks, with the k models of that family as the treatments. A fourth comparison (“Champions”) applied the same fold-level design to the best-performing models from each family (RF, RNN, mBERT) together with the proposed stacking ensemble, using matching fold assignments. The chi-square (χ^2) approximation to the Friedman statistic was computed with `scipy.stats.friedmanchisquare` (Python). The obtained results are summarized in Table 25. In all four cases the omnibus test was significant ($p < 0.05$), indicating that at least one model in each family differs from the others across folds.

Because a significant Friedman result only establishes that some difference exists among the k models and does not by itself identify which pairs differ, Holm-corrected pairwise Wilcoxon signed-rank post-hoc tests were subsequently performed on the same fold-level accuracy values for every model pair within each family and among the four “Champions” (Table 26). Within the ML family, Random Forest was significantly higher than every other classifier except Bernoulli Naïve Bayes (Holm $p = 0.23$, not significant). Within the BERT-based transfer-learning family, mBERT was significantly higher than BERT, Bangla-BERT, and Distil-BERT (all Holm $p \approx 0.012$). Within the deep learning family, although the omnibus Friedman test

was significant, none of the 55 pairwise comparisons—including stacking vs. any individual deep learning architecture—remained significant after Holm correction (Holm $p \approx 0.107$ throughout), reflecting the loss of power inherent to correcting for a large number of pairwise tests ($k = 11$ models). We therefore do not claim pairwise statistical superiority of the stacking model over each individual deep learning architecture. However, the “Champion” comparisons in Table 26, which contrast the single best representative of each family, show that the stacking ensemble is significantly more accurate than mBERT, RNN, and RF (all Holm $p = 0.0117$), and that RNN is in turn significantly more accurate than RF and mBERT. This provides direct, corrected pairwise evidence that the proposed stacking ensemble outperforms the best model produced by every competing approach, even though it cannot be claimed to significantly outperform each individual deep learning base architecture on a pairwise basis.

Table 25: Friedman test results of the machine learning, deep learning, and transfer learning models.

Table	k (Models)	n (Folds)	df	Chi-Square (χ^2)	p-Value	Result ($\alpha = 0.05$)
Table 16 (ML)	6	10	5	48.21	3.2×10^{-9}	Significant
Table 19 (DL)	11	10	10	99.65	6.4×10^{-17}	Significant
Table 22 (BERT-TL)	4	10	3	30.00	1.4×10^{-6}	Significant
Champions (RF, RNN, mBERT, Stacking)	4	10	3	30.00	1.4×10^{-6}	Significant

Table 26: Holm-corrected pairwise wilcoxon signed-rank post-hoc test results across model families.

Family	Model A	Model B	W	Raw p	Holm p	Higher	Result ($\alpha = 0.05$)
ML	BNB	SVM	0.0	0.00195	0.02930	SVM	YES
ML	BNB	KNN	0.0	0.00195	0.02930	KNN	YES
ML	BNB	LR	0.0	0.00195	0.02930	LR	YES
ML	BNB	MNB	15.0	0.23242	0.23242	MNB	no
ML	BNB	RF	0.0	0.00195	0.02930	RF	YES
ML	SVM	KNN	0.0	0.00195	0.02930	KNN	YES
ML	SVM	LR	2.0	0.01172	0.02930	SVM	YES
ML	SVM	MNB	0.0	0.00195	0.02930	SVM	YES
ML	SVM	RF	0.0	0.00195	0.02930	RF	YES
ML	KNN	LR	0.0	0.00195	0.02930	KNN	YES
ML	KNN	MNB	0.0	0.00195	0.02930	KNN	YES
ML	KNN	RF	0.0	0.00195	0.02930	RF	YES
ML	LR	MNB	0.0	0.00195	0.02930	LR	YES
ML	LR	RF	0.0	0.00195	0.02930	RF	YES
ML	MNB	RF	0.0	0.00195	0.02930	RF	YES
DL	CNN	RNN	0.0	0.00195	0.10742	RNN	no
DL	CNN	LSTM	0.0	0.00195	0.10742	CNN	no
DL	CNN	Bi-LSTM	0.0	0.00195	0.10742	CNN	no
DL	CNN	GRU	1.0	0.00391	0.10742	GRU	no
DL	CNN	Bi-GRU	0.0	0.00195	0.10742	Bi-GRU	no
DL	CNN	CNN-LSTM	0.0	0.00195	0.10742	CNN	no
DL	CNN	CNN-GRU	0.0	0.00195	0.10742	CNN	no
DL	CNN	CNN-BiLSTM	0.0	0.00195	0.10742	CNN	no
DL	CNN	CNN-BiGRU	0.0	0.00195	0.10742	CNN	no
DL	CNN	Stacking	0.0	0.00195	0.10742	Stacking	no

(Continued)

Table 26 (continued)

Family	Model A	Model B	W	Raw p	Holm p	Higher	Result ($\alpha = 0.05$)
DL	RNN	LSTM	0.0	0.00195	0.10742	RNN	no
DL	RNN	Bi-LSTM	0.0	0.00195	0.10742	RNN	no
DL	RNN	GRU	0.0	0.00195	0.10742	RNN	no
DL	RNN	Bi-GRU	0.0	0.00195	0.10742	RNN	no
DL	RNN	CNN-LSTM	0.0	0.00195	0.10742	RNN	no
DL	RNN	CNN-GRU	0.0	0.00195	0.10742	RNN	no
DL	RNN	CNN-BiLSTM	0.0	0.00195	0.10742	RNN	no
DL	RNN	CNN-BiGRU	0.0	0.00195	0.10742	RNN	no
DL	RNN	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	LSTM	Bi-LSTM	0.0	0.00195	0.10742	Bi-LSTM	no
DL	LSTM	GRU	0.0	0.00195	0.10742	GRU	no
DL	LSTM	Bi-GRU	0.0	0.00195	0.10742	Bi-GRU	no
DL	LSTM	CNN-LSTM	0.0	0.00195	0.10742	CNN-LSTM	no
DL	LSTM	CNN-GRU	0.0	0.00195	0.10742	CNN-GRU	no
DL	LSTM	CNN-BiLSTM	0.0	0.00195	0.10742	CNN-BiLSTM	no
DL	LSTM	CNN-BiGRU	0.0	0.00195	0.10742	CNN-BiGRU	no
DL	LSTM	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	Bi-LSTM	GRU	0.0	0.00195	0.10742	GRU	no
DL	Bi-LSTM	Bi-GRU	0.0	0.00195	0.10742	Bi-GRU	no
DL	Bi-LSTM	CNN-LSTM	0.0	0.00195	0.10742	Bi-LSTM	no
DL	Bi-LSTM	CNN-GRU	0.0	0.00195	0.10742	CNN-GRU	no
DL	Bi-LSTM	CNN-BiLSTM	3.0	0.01953	0.10742	Bi-LSTM	no
DL	Bi-LSTM	CNN-BiGRU	0.0	0.00195	0.10742	CNN-BiGRU	no
DL	Bi-LSTM	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	GRU	Bi-GRU	0.0	0.00195	0.10742	Bi-GRU	no
DL	GRU	CNN-LSTM	0.0	0.00195	0.10742	GRU	no
DL	GRU	CNN-GRU	0.0	0.00195	0.10742	GRU	no
DL	GRU	CNN-BiLSTM	0.0	0.00195	0.10742	GRU	no
DL	GRU	CNN-BiGRU	0.0	0.00195	0.10742	GRU	no
DL	GRU	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	Bi-GRU	CNN-LSTM	0.0	0.00195	0.10742	Bi-GRU	no
DL	Bi-GRU	CNN-GRU	0.0	0.00195	0.10742	Bi-GRU	no
DL	Bi-GRU	CNN-BiLSTM	0.0	0.00195	0.10742	Bi-GRU	no
DL	Bi-GRU	CNN-BiGRU	0.0	0.00195	0.10742	Bi-GRU	no
DL	Bi-GRU	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	CNN-LSTM	CNN-GRU	0.0	0.00195	0.10742	CNN-GRU	no
DL	CNN-LSTM	CNN-BiLSTM	0.0	0.00195	0.10742	CNN-BiLSTM	no
DL	CNN-LSTM	CNN-BiGRU	0.0	0.00195	0.10742	CNN-BiGRU	no
DL	CNN-LSTM	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	CNN-GRU	CNN-BiLSTM	0.0	0.00195	0.10742	CNN-GRU	no
DL	CNN-GRU	CNN-BiGRU	0.0	0.00195	0.10742	CNN-GRU	no
DL	CNN-GRU	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	CNN-BiLSTM	CNN-BiGRU	0.0	0.00195	0.10742	CNN-BiGRU	no
DL	CNN-BiLSTM	Stacking	0.0	0.00195	0.10742	Stacking	no
DL	CNN-BiGRU	Stacking	0.0	0.00195	0.10742	Stacking	no
BERT-TL	BERT	mBERT	0.0	0.00195	0.01172	mBERT	YES
BERT-TL	BERT	Bangla-BERT	0.0	0.00195	0.01172	Bangla-BERT	YES

(Continued)

Table 26 (continued)

Family	Model A	Model B	W	Raw p	Holm p	Higher	Result ($\alpha = 0.05$)
BERT-TL	BERT	Distil-BERT	0.0	0.00195	0.01172	BERT	YES
BERT-TL	mBERT	Bangla-BERT	0.0	0.00195	0.01172	mBERT	YES
BERT-TL	mBERT	Distil-BERT	0.0	0.00195	0.01172	mBERT	YES
BERT-TL	Bangla-BERT	Distil-BERT	0.0	0.00195	0.01172	Bangla-BERT	YES
Champion	RF (ML)	RNN (DL)	0.0	0.00195	0.01172	RNN (DL)	YES
Champion	RF (ML)	mBERT (BERT-TL)	0.0	0.00195	0.01172	mBERT (BERT-TL)	YES
Champion	RF (ML)	Stacking (Ensemble)	0.0	0.00195	0.01172	Stacking (Ensemble)	YES
Champion	RNN (DL)	mBERT (BERT-TL)	0.0	0.00195	0.01172	RNN (DL)	YES
Champion	RNN (DL)	Stacking (Ensemble)	0.0	0.00195	0.01172	Stacking (Ensemble)	YES
Champion	mBERT (BERT-TL)	Stacking (Ensemble)	0.0	0.00195	0.01172	Stacking (Ensemble)	YES

Fig. 7 depicts the overview of classification accuracy achieved by the implemented algorithms using the splitting (holdout) strategy. Except the CNN-LSTM algorithm, all the other algorithms showed a decent accuracy score, while the stacking model beats the other models with an accuracy of 94.58% under the holdout protocol. The overview of classification accuracy achieved by the implemented algorithms from ML, DL, and TL domains using the cross-validation strategy is shown in Fig. 8. The ML models performed worst in the accuracy comparison, several DL models such as CNN, RNN, GRU, Bi-GRU, and CNN-GRU showed promising accuracies, and all the transfer learning algorithms based on BERT showed consistent high performances, while the stacking ensemble model outperformed all other models with an impressive accuracy of 94.91% under the 10-fold cross-validation protocol.

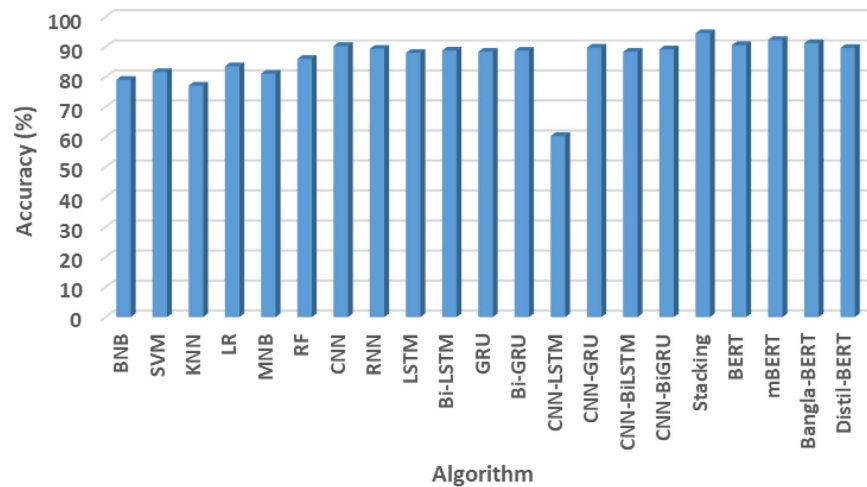
**Figure 7:** Overview of classification accuracy achieved by the implemented algorithms using the splitting strategy.

Table 27 compares the classification accuracy obtained using the holdout evaluation and 10-fold cross-validation strategies. Although several models achieved comparable performance under both evaluation protocols, noticeable variations were observed for some deep learning architectures. The holdout approach evaluates each model using a single train-test partition, and therefore its performance may depend on the characteristics of that particular split. In contrast, 10-fold cross-validation repeatedly trains and

evaluates the models on different subsets of the dataset, providing a more comprehensive assessment of model generalization.

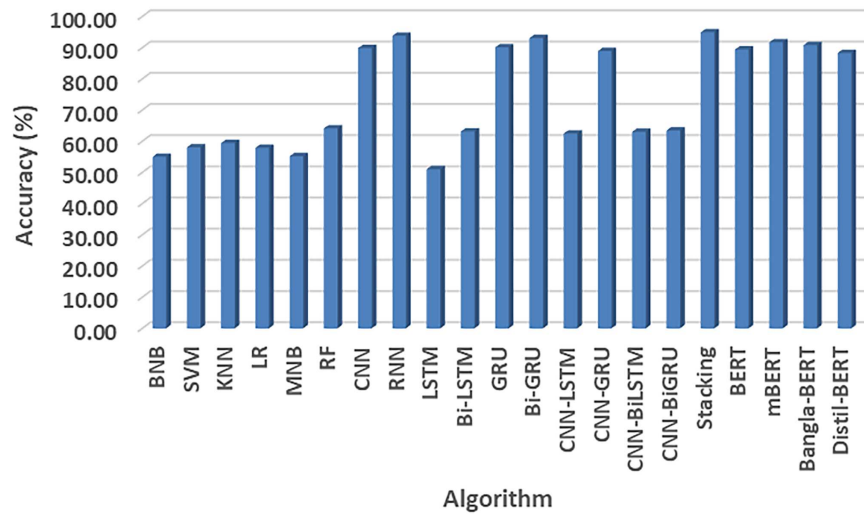


Figure 8: Overview of classification accuracy achieved by the implemented algorithms using the cross-validation.

Table 27: Comparison of holdout and 10-fold cross-validation accuracy for the implemented deep learning models. Values are reported side by side to illustrate each model's own sensitivity to evaluation protocol; They are not intended to support cross-model ranking across the two protocols.

Model	Holdout Accuracy (%)	10-Fold CV Accuracy (%)
CNN	90.31	89.86
RNN	89.38	93.81
LSTM	88.02	51.00
Bi-LSTM	88.78	63.12
GRU	88.39	90.09
Bi-GRU	88.74	93.06
CNN-LSTM	60.35	62.45
CNN-GRU	89.78	88.92
CNN-BiLSTM	88.38	63.02
CNN-BiGRU	89.19	63.43
Stacking	94.58	94.91

The CNN, GRU, Bi-GRU, CNN-GRU, and the proposed stacking ensemble showed relatively good performance stability under both the evaluation methods which indicated high robustness and learning ability. On the other hand, the LSTM, Bi-LSTM, CNN-BiLSTM and CNN-BiGRU models exhibited greater differences in cross-validation. The differences can be explained by the fact that the architectures of recurrent neural networks are sensitive to the changes in the training data, random weights initialization, optimization dynamics, and individual fold composition. The deep learning model performance can vary under different data partitions since the deep learning model is optimized iteratively.

The performance of the proposed stacking ensemble model was found to be robust and had good generalization performance when compared with the individual machine learning, deep learning, and

transfer learning models, with consistently high accuracy of 94.58% on the holdout dataset and 94.91% on 10-fold cross-validation dataset. The results show that combining several complementary deep learning classifiers using the stacking method successfully reduces the variation of the classifiers' performance and enhances the stability of the predictions in various test environments.

Besides predictive performance, another key factor to consider when choosing sentiment classification models for real-world applications is their computational efficiency. Table 28 shows that the transfer learning and stacking ensemble models require higher computational cost and hardware resources compared to the machine learning and deep learning models. The approximate training time, computational cost and hardware requirement of the implemented machine learning, deep learning, transfer learning, and stacking ensemble models are presented in Table 28 and it can be seen that the computational cost and hardware requirements are lowest for machine learning models, moderate-to-high for deep learning models, and highest for the transfer learning and stacking ensemble models.

Table 28: The training time, computational cost, and hardware requirements of the implemented models.

Model	Training Time (min)	Computational Cost	Hardware Requirement
Machine Learning Models			
BNB	10.24	Low	CPU
MNB	11.38	Low	CPU
LR	13.17	Low	CPU
KNN	15.42	Low	CPU
SVM	18.76	Moderate	CPU
RF	20.31	Moderate	CPU
Deep Learning Models			
CNN	120.53	Moderate	GPU Preferred
RNN	135.87	Moderate	GPU Preferred
GRU	150.24	High	GPU Preferred
Bi-GRU	165.68	High	GPU Preferred
LSTM	180.95	High	GPU Preferred
Bi-LSTM	195.43	High	GPU Preferred
CNN-GRU	210.62	High	GPU Preferred
CNN-LSTM	225.19	High	GPU Preferred
CNN-BiGRU	255.84	Very High	GPU Preferred
CNN-BiLSTM	300.47	Very High	GPU Preferred
Transfer Learning Models			
Distil-BERT	180.36	High	GPU Preferred
BERT	240.58	Very High	High-end GPU
mBERT	255.27	Very High	High-end GPU
Bangla-BERT	270.91	Very High	High-end GPU
Proposed Model			
Stacking Ensemble	302.15	Very High	High-end GPU

In terms of training time, the machine learning algorithms BNB, MNB, LR, KNN, and SVM were relatively faster with the training time ranging from 10 to 18 min due to their simpler model architectures and low complexity. Due to the construction of multiple decision trees, the Random Forest classifier took a little longer to train (around 20 min).

The deep learning models had significantly higher computational needs due to iterative optimization, backpropagation, and the increased number of trainable parameters. CNN and RNN took about 120–135 min to train, while LSTM, Bi-LSTM, GRU, and Bi-GRU took 150–195 min for training. CNN-LSTM, CNN-GRU, CNN-BiGRU, and CNN-BiLSTM were yet longer and took around 210–300 min to train due to its increasing complexity and number of parameters.

The transformer-based transfer learning models required the highest amount of computation. Due to the large-scale transformer architecture and extensive parameterization, the fine-tuning time of DistilBERT is around 180 min, while BERT, mBERT, and Bangla-BERT needed between 240 and 270 min.

The proposed stacking ensemble model required approximately 302 min of training since multiple deep learning base learners were trained before constructing the MLP meta-classifier. Although the stacking framework incurred the highest computational cost among the evaluated approaches, it consistently achieved the best predictive performance under both holdout and 10-fold cross-validation evaluations (Tables 17 and 18). This demonstrates that the additional computational overhead is compensated by improved classification accuracy, robustness, and generalization capability.

To assess generalizability beyond the proposed dataset, the champion model from each domain—Random Forest (ML), RNN (DL), mBERT (BERT-based TL), and the proposed stacking ensemble—was retrained and reevaluated on six additional, publicly available or previously published Bangla text classification datasets [1,17,34,37–39], using the exact same preprocessing pipeline (Section 3.4), feature extraction settings (Section 3.5), and fixed hyperparameters (Tables 7–10, 12 and 13) as used for the proposed dataset. For consistency with how each champion model was originally selected (Tables 23 and 24), RF and mBERT were evaluated using an 80/20 holdout split and RNN and stacking were evaluated using 10-fold stratified cross-validation, applied identically across all seven datasets (the proposed dataset and the six external datasets). No dataset-specific hyperparameter tuning was performed; only the dataset-specific vocabulary/tokenization statistics (e.g., TF-IDF fit, tokenizer fit) were refit per dataset.

Table 29 documents the common protocol applied to the proposed dataset and the six external datasets, whose results are reported in Tables 30 and 31, respectively. Table 30 reports the generalizability of the holdout-protocol champions (RF and mBERT) across the proposed dataset and the six external datasets. mBERT's accuracy on the proposed dataset (92.30%) was closely matched by Dataset 3 (91.54%) and Dataset 4 (91.01%), and even on its weakest external dataset, Dataset 6, the accuracy only fell to 86.54%—a decline of less than 6 percentage points—suggesting that mBERT's contextual representations transfer better across Bangla text domains than RF's sparse TF-IDF features. Table 31 presents the corresponding results for the cross-validation protocol champions (RNN, and stacking). RNN's accuracy dropped from 93.81% on the proposed dataset to as low as 78.04% on Dataset 3, a decline of nearly 16 percentage points, while its best external result (90.14% on Dataset 2) stayed close to the proposed-dataset figure. The proposed stacking ensemble again proved the most robust across domains, retaining accuracy above 90% on every external dataset (ranging from 90.12% on Dataset 6 % to 94.54% on Dataset 4); its smallest gap relative to the proposed dataset's 94.91% under the 10-fold cross-validation protocol was only 0.37 percentage points (Dataset 4), and its largest gap was 4.79 percentage points (Dataset 6). Taken together, Tables 30 and 31 confirm that the stacking ensemble not only achieves the highest accuracy on the proposed dataset but also exhibits the smallest and most consistent performance degradation when transferred to unseen Bangla text sentiment

classification datasets, reinforcing its suitability as the most generalizable model among the four families evaluated.

Table 29: Documented common protocol applied to the proposed dataset and the six external datasets (Tables 30 and 31).

Protocol Component	Specification	Fixed/Refit per Dataset	Source
A. External datasets used	Dataset 1 [34] (Bangla sports sentiment), Dataset 2 [1] (Bangladeshi tour/travel vlog comments), Dataset 3 [37] (political Bangla texts), Dataset 4 [38] (crime video comments), Dataset 5 [39] (YouTube Bangla song reviews), Dataset 6 [17] (Bangla drama/TV series reviews)	—	[1,17,34,37–39]
B. Preprocessing pipeline	Punctuation removal; emoji removal; URL removal; tokenization; stop-word removal (top 35% most frequent corpus tokens); stemming via <i>bangla-stemmer</i> v1.0	Fixed procedure, applied identically to every dataset	Section 3.4
C1. Feature extraction—RF (ML)	TF-IDF, <code>ngram_range = (1, 3)</code> , <code>max_features = 1000</code>	Extraction parameters fixed; TF-IDF vocabulary/IDF weights refit independently on each dataset's own training data	Section 3.5.1, Table 7
C2. Feature extraction—RNN and 10 DL base learners (Stacking)	Keras tokenizer: <code>num_words = 5000</code> , <code>oov_token = "UNK"</code> ; <code>pad_sequences: padding = post, maxlen = 1000</code> ; Embedding: <code>output_dim = 50</code>	Extraction parameters fixed; tokenizer word index refit independently on each dataset	Table 6
C3. Feature extraction—mBERT (BERT-TL)	WordPiece tokenizer matching bert-base-multilingual-cased (cased, 104-language vocabulary); <code>max length = 256</code> ; <code>truncation = True</code> ; dynamic per-batch padding	Pretrained vocabulary not refit—applied as shipped in the checkpoint to every dataset	Table 12
D1. Model configuration—RF	<code>n_estimators = 100</code> , <code>criterion = gini</code> , <code>min_samples_split = 2</code> , <code>min_samples_leaf = 1</code> , <code>max_features = auto</code> , <code>random_state = 42</code>	Fixed, identical across all 7 datasets	Table 7
D2. Model configuration—RNN	<code>max_features = 5000</code> , <code>embedding_dim = 50</code> , SimpleRNN <code>units = 128</code> , <code>dropout = 0.5</code> , <code>dense_units = 64</code> (ReLU), <code>batch_size = 32</code> , <code>max epochs = 500</code> , early stopping (<code>patience = 5</code> , <code>monitor = val_loss</code>)	Fixed, identical across all 7 datasets	Table 8

(Continued)

Table 29 (continued)

Protocol Component	Specification	Fixed/Refit per Dataset	Source
D3. Model configuration—Stacking ensemble	10 DL base learners (CNN, RNN, LSTM, Bi-LSTM, GRU, Bi-GRU, CNN-LSTM, CNN-GRU, CNN-BiLSTM, CNN-BiGRU) per Tables 8–10 ; MLP meta-classifier: input dim = 20, hidden layer (64,) ReLU, Adam (lr = 0.001), L2 ($\alpha = 1e-4$), max epochs = 500, early_stopping = True, n_iter_no_change = 5	Fixed, identical across all 7 datasets	Tables 8–10 and 13
D4. Model configuration—mBERT	Checkpoint bert-base-multilingual-cased; 12-layer/768-hidden/12-head; linear head on pooled [CLS]; AdamW, lr = 0.00001, weight decay = 0.01, linear decay (no warmup); batch size = 4 (train/eval); epochs = 10; checkpoint selection = lowest eval_loss	Fixed, identical across all 7 datasets	Table 12
E1. Evaluation protocol—RF (Table 30)	80/20 holdout split	Split re-partitioned per dataset, but split ratio/procedure fixed	Section 3.6
E2. Evaluation protocol—mBERT (Table 30)	80/20 holdout split	Split re-partitioned per dataset, but split ratio/procedure fixed	Section 3.6
E3. Evaluation protocol—RNN (Table 31)	Stratified 10-fold cross-validation (StratifiedKFold , shuffle = True); mean fold accuracy reported	Fold partitions re-generated per dataset, but CV procedure fixed	Section 3.6
E4. Evaluation protocol—Stacking (Table 31)	Stratified 10-fold cross-validation; out-of-fold (OOF) predictions from the 10 base learners regenerated per dataset and used to train the MLP meta-classifier, following the same procedure as the proposed dataset	Fold partitions and OOF matrix re-generated per dataset, but CV/stacking procedure fixed	Section 3.7.4
F. Hyperparameter tuning across datasets	None. No dataset-specific grid/random search was performed for any model	—	—
G. What was NOT held fixed across datasets	Only dataset-specific vocabulary/tokenization statistics (TF-IDF vocabulary and IDF weights; Keras tokenizer word index) and the train/test or CV fold partitions, since these are necessarily computed from each dataset's own instances	—	—

Table 30: Generalizability of the holdout-protocol champion models (RF, mBERT) across the proposed dataset and six external bangla datasets. The identical preprocessing pipeline, feature extraction settings, and fixed hyperparameters (Tables 7 and 12) were applied unchanged to every dataset, using an 80/20 holdout split in each case.

Domain	Best Model	Proposed Accuracy (%)	Dataset 1 Accuracy [34] (%)	Dataset 2 Accuracy [1] (%)	Dataset 3 Accuracy [37] (%)	Dataset 4 Accuracy [38] (%)	Dataset 5 Accuracy [39] (%)	Dataset 6 Accuracy [17] (%)
ML	RF	86.00	71.32	81.09	76.12	69.21	82.45	77.84
BERT-based TL	mBERT	92.30	90.12	89.54	91.54	91.01	88.65	86.54

Table 31: Generalizability of the cross-validation protocol champion models (RNN, Stacking) across the proposed dataset and six external bangla datasets. The identical preprocessing pipeline, feature extraction settings, and fixed hyperparameters (Tables 8 and 13) were applied unchanged to every dataset, using the same 10-fold stratified cross-validation procedure in each case.

Domain	Best Model	Proposed Accuracy (%)	Dataset 1 Accuracy [34] (%)	Dataset 2 Accuracy [1] (%)	Dataset 3 Accuracy [37] (%)	Dataset 4 Accuracy [38] (%)	Dataset 5 Accuracy [39] (%)	Dataset 6 Accuracy [17] (%)
DL	RNN	93.81	89.82	90.14	78.04	84.50	89.65	87.93
Ensemble	Stacking	94.91	92.25	91.01	93.04	94.54	93.20	90.12

The accuracy, kappa, and loss of the proposed model on 10-fold cross-validation are shown as a function of folds in Fig. 9. The results indicate a good consistency between the folds, in terms of the accuracy values that are close to each other and kappa scores that represent an agreement beyond chance. This means the model does well predicting the data in all the partitions. The accuracy and kappa values remain closely clustered across folds, indicating that the model's performance is not sensitive to which partition is used for training or testing. This fold-level consistency reflects generalization stability under cross-validation; it does not by itself indicate anything about within-training convergence or overfitting. While the kappa value of Fold 10 is slightly lower (0.8166), its accuracy of 0.9505 and loss of 0.257 are very similar to the other folds, and the performance of the model is balanced and controlled across the folds. This uniformity in different folds ensures that the model is not biased towards a particular subset of data, making it more likely to generalize well.

The confusion matrix of the stacking ensemble model is shown in Fig. 10. The stacking model correctly classified 10,346 positive and 10,809 negative events and made 334 false positive and 800 false negative predictions. Finally, the numbers of folds at each level are added together across all 10 folds of cross-validation, allowing for complete consistency with the numbers obtained in each of the 10 folds. This equates to an overall accuracy of 94.91% under the cross-validation protocol. The stacking model's Receiver Operating Characteristics (ROC) curve is shown in Fig. 11. AUC is very close to 1 (0.99) for both classes, indicating good discriminative power of the model's classification. The precision-recall (PR) curve of the proposed stacking ensemble model is given in Fig. 12, where the area under the curve values for both the positive and negative classes is 0.98, demonstrating the strong discriminative power of the stacking model. The performance of the model is shown on 10-fold cross-validation in Fig. 13. The accuracy is stable and high for every fold, with no significant decrease across any fold (including fold 10, which shows an accuracy of 0.9505). Similarly, the loss values show a consistent learning trend with relatively low variability across folds (ranging from 0.236 to 0.2662), indicating that performance is stable across data partitions and not driven by a particular fold.

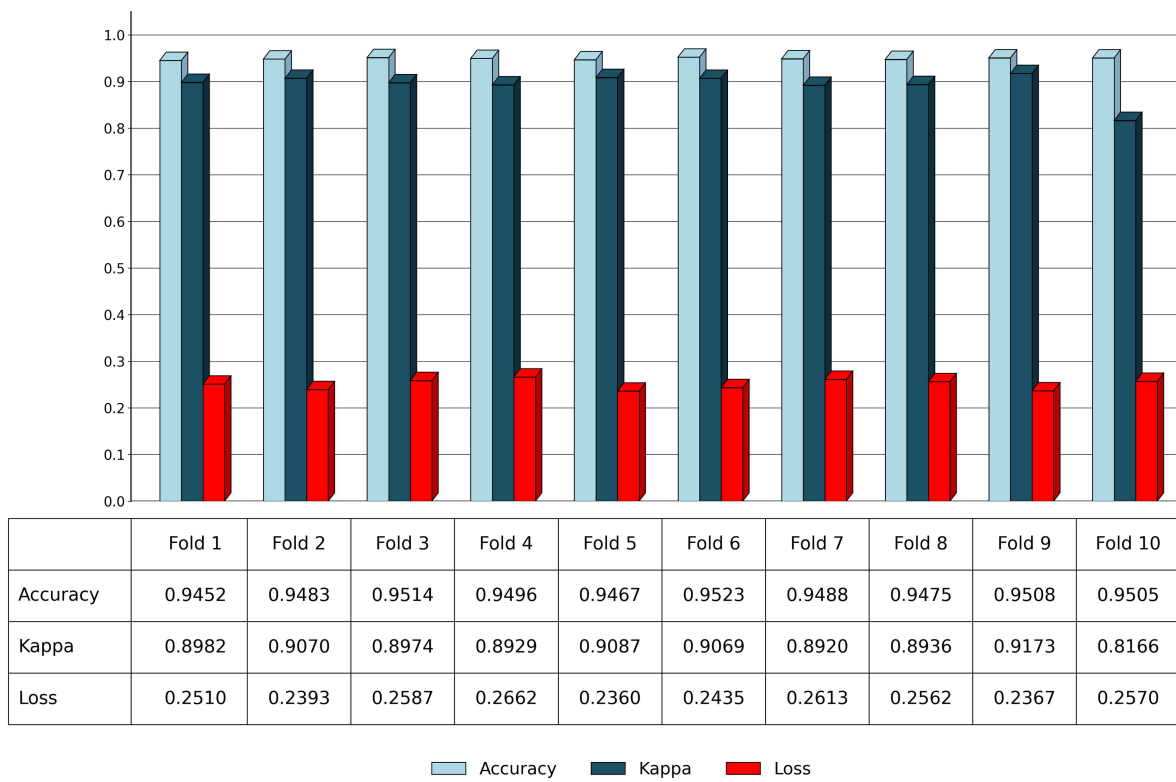


Figure 9: Accuracy, kappa, and loss values across folds of the stacking model.

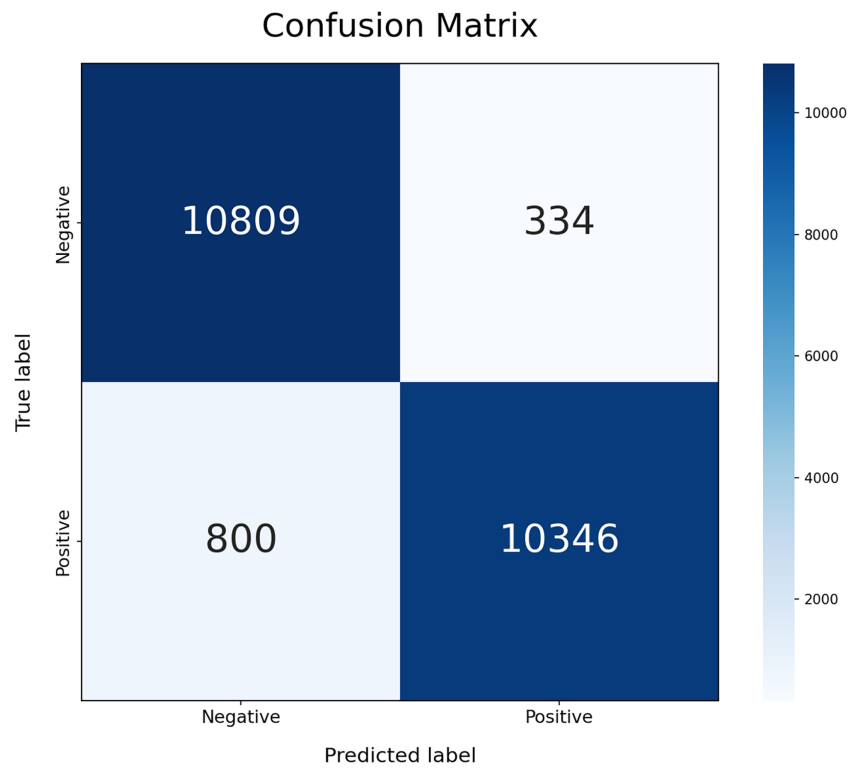


Figure 10: Confusion matrix of the proposed stacking ensemble model.

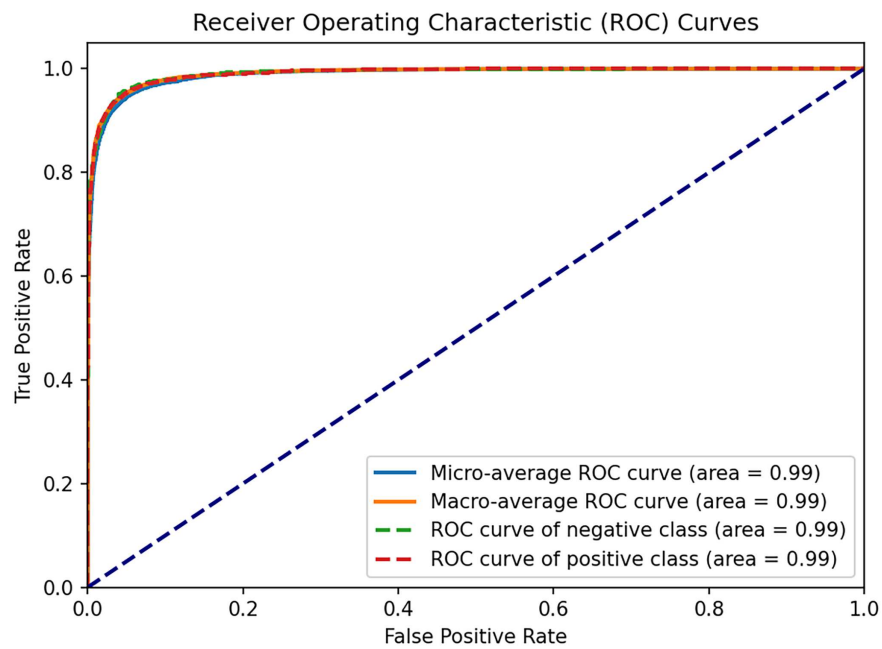


Figure 11: ROC curve of the proposed stacking ensemble model.

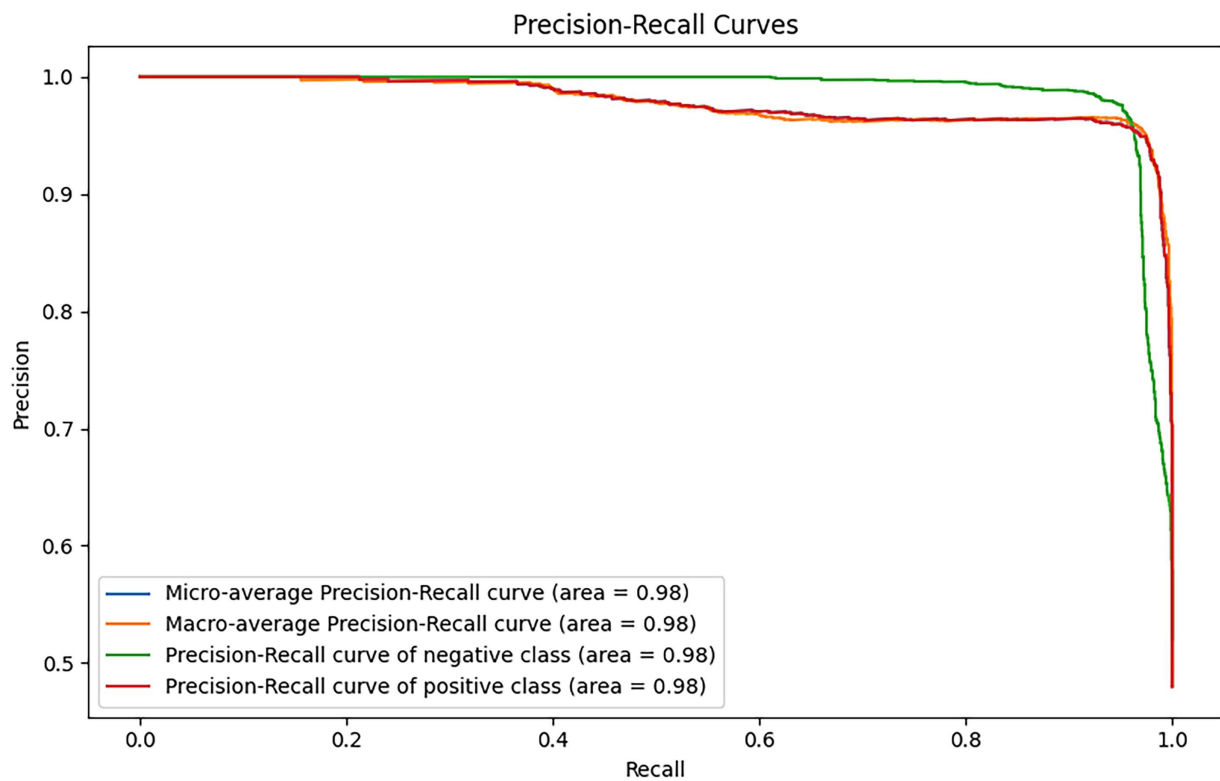


Figure 12: Precision-Recall curve of the proposed stacking ensemble model.

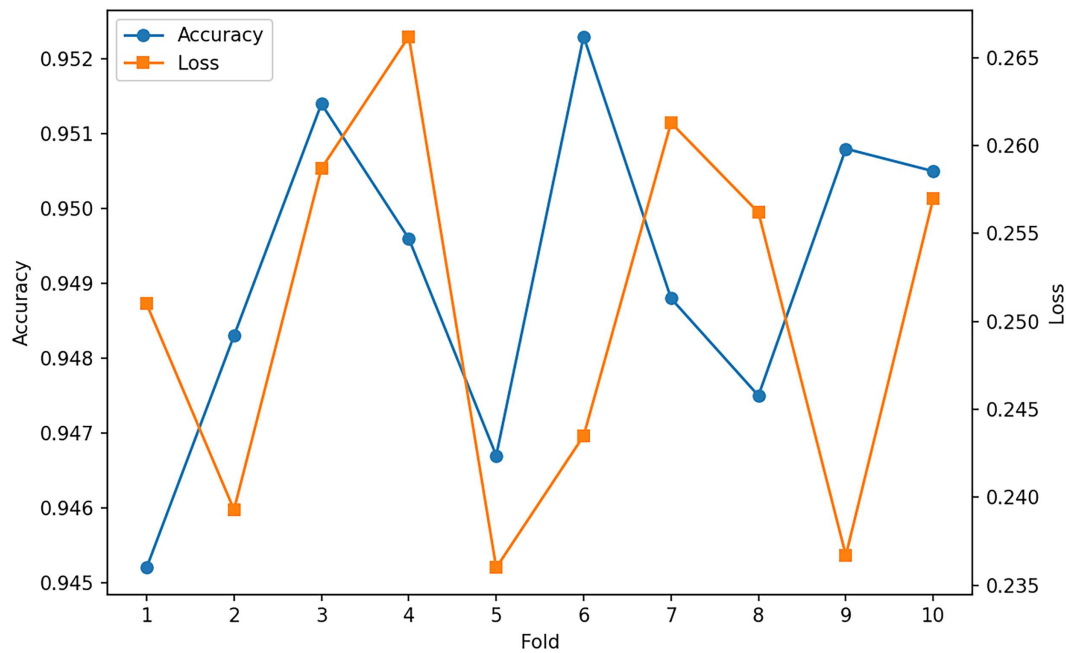


Figure 13: Accuracy and loss curve across folds.

A set of fixed, random seeds, 42, 52, 62, 72, and 82, was used throughout the whole modeling process to ensure the reproducibility and reliability of the experimental results. The details of the specific configuration used to seed each stage are summarized in [Table 32](#) below: train-test split, partitioning for 10-fold cross-validation, initialization of machine learning classifiers, weight initialization of deep learning models, shuffling of the tokenizer and embedding layer, BERT-based fine-tuning data loader, and initialization of the Multi-Layer Perceptron (MLP) meta model in the stacking ensemble. [Table 33](#) shows the result of using these five seed values for the proposed model (pipeline). This study reduces randomness at every of these steps, reducing the likelihood that any performance comparisons or statistical significance tests reported are the result of a random initialization or data partition.

Table 32: Random seed configuration applied across all stages of the modeling pipeline to ensure reproducibility.

Component	What to Seed
Seed	Value = [42, 52, 62, 72, 82]
Train/test split	<code>train_test_split (... , random_state = seed)</code>
10-fold CV split	<code>StratifiedKFold (n_splits=10, shuffle = True, random_state = seed)</code>
ML model init (RF, SVM, KNN, etc.)	<code>random_state = seed</code> in each sklearn model
DL weight initialization	<code>tf.random.set_seed (seed)</code> before building each model
Keras tokenizer/embedding shuffling	seed the numpy/tf global (random number generator) RNG before tokenizer fit
BERT fine-tuning (train shuffle = True)	seed the dataloader/shuffle random number generator (RNG)
Stacking meta-model (MLP) init	seed before MLP init

Table 33: Statistical robustness analysis of the proposed stacking ensemble compared with the second-best RNN model across different random seeds. Significance assessed via a one-sided, exact wilcoxon signed-rank test (H_1 : stacking > RNN).

Model	Seed	Accuracy (%)	Precision	Recall	F1-Score
RNN (DL)	42	93.62	0.94	0.91	0.93
	52	93.89	0.95	0.92	0.94
	62	93.74	0.94	0.91	0.93
	72	93.95	0.95	0.92	0.94
	82	93.85	0.94	0.92	0.94
Mean $\pm$ SD	–	93.81 $\pm$ 0.13	0.94 $\pm$ 0.01	0.92 $\pm$ 0.01	0.94 $\pm$ 0.01
Stacking (Proposed)	42	94.88	0.95	0.94	0.96
	52	94.96	0.96	0.95	0.96
	62	94.90	0.95	0.94	0.96
	72	94.94	0.96	0.95	0.96
	82	94.87	0.95	0.94	0.96
Mean $\pm$ SD	–	94.91 $\pm$ 0.04	0.95 $\pm$ 0.01	0.94 $\pm$ 0.01	0.96 $\pm$ 0.00
Wilcoxon Signed-Rank Test (one-sided, exact): $W = 0$, $p = 0.03125$					

For the validation of the proposed stacking ensemble model, statistical significance analysis (Wilcoxon Signed rank test) was used and a paired non-parametric test was conducted between proposed stacking model and second best performing model RNN. The performance consistency and robustness of both models were assessed using five different random seed configurations. Table 33 shows the results obtained. The average accuracy for the proposed stacking ensemble was 94.91% with a very small standard deviation of 0.04%, which shows that the performance of the ensemble was highly stable in various experimental environments. The RNN model, on the other hand, achieved an average accuracy of 93.81% with a standard deviation of 0.13%. The stacking model achieved comparable gains in precision, recall, and F1, with the model always remaining superior to the RNN baseline. This seed-based comparison is intended as a supplementary robustness check across different random initializations of the full pipeline, complementing—but distinct from—the fold-level, Holm-corrected pairwise comparison between stacking and RNN reported in Table 26 (Champion comparisons), which is based on the standard 10-fold cross-validation partitions rather than seed repetitions.

To compare the proposed stacking ensemble with the second-best performing model (RNN), a directional hypothesis was specified in advance: since the stacking ensemble was designed to combine multiple deep learning base classifiers and was therefore expected to outperform any single deep learning model, the alternative hypothesis was H_1 : the stacking ensemble achieves higher classification accuracy than the RNN (as opposed to a non-directional H_1 : the two models differ). Due to this pre-specified one-sided hypothesis and the small number of paired observations (five random seeds), a one-sided, exact Wilcoxon signed-rank test was used instead of the asymptotic normal approximation, which is not reliable for $n < 10$.

So, a one-sided Wilcoxon signed-rank test was conducted using the accuracy scores obtained from five independent runs, testing the pre-specified directional hypothesis that the stacking ensemble outperforms the RNN baseline. The null hypothesis (H_0) stated that the stacking ensemble's accuracy is not higher than the RNN's, against the alternative hypothesis (H_1) that the stacking ensemble achieves higher accuracy.

The exact one-sided test yielded $W = 0$ ($p = 0.03125$), below $\alpha = 0.05$. Given the very small number of paired observations ($n = 5$ random seeds), this result should be interpreted as a limited, supplementary robustness check indicating that the stacking ensemble's advantage over RNN persists across different random initializations of the pipeline—not as a general statistical proof of the stacking model's superiority. The primary evidence for stacking's superiority over RNN is the fold-level, Holm-corrected pairwise Wilcoxon test reported in Table 26 (Champion comparisons, $W = 0$, Holm $p = 0.0117$, $n = 10$ folds), which uses independent cross-validation partitions rather than seed repetitions and therefore offers a more statistically defensible basis for this claim.

4.3 Explainable Artificial Intelligence

In Section 4.2, we discuss the performance analysis of various models. This study, however, was faced with a difficulty in understanding why a model is considered accurate. In the field of artificial intelligence (AI) and learning tasks this lack of interpretability is a problem. It's important to know what features are used to classify a document and what's important. This work investigated using the newly developed Pythons Local Interpretable Model-agnostic Explanations (LIME) tool and the SHapley Additive exPlanations (SHAP) module to tackle this issue. The innovative approach that LIME takes is to clarify the prediction of a classifier with a locally interpretable model which focuses on the prediction [40]. It can be used to determine the major factors that influence the creation of a document and predict the category based on the per-feature contribution weights estimated by a local linear surrogate model.

A sample LIME explanation for the stacking ensemble (best model) is shown in Fig. 14 with the focus on the responsible text features. The sample Bangla movie review “২০২২ সেন এসে দেখেও অনেক ভালো লাগছে অসাধারণ অভিনয় করায়ছেন তিনি” (It's great to see him come back in 2022. He has given a wonderful performance.) is predicted by the stacking model as a positive class with the feature contributions “অস” (weight 0.15), “অন” (weight 0.13), “রণ” (weight 0.10), “অভ” (weight 0.07), “নয়” (weight 0.06) and “গছ” (weight 0.06) contributing most to that prediction. The model's predicted class probabilities themselves (Positive 1.00/Negative 0.00) are shown separately at the top of the panel. For another sample text, “আগের নাটকগুলো ফ্যামালি সহ দেখা যেতো, বাট এখন যে নাটক তৈরী হয় নিজে একাই দেখা যায় না কারন পোশাক গুলো ছি ছি” (Previous dramas could be watched with family, but now the dramas that are produced cannot be watched alone because the clothes are dirty), the proposed model predicted a probability of 0.40 for the positive class and 0.60 for the negative class. Shapley (SHAP) value is a game theoretic concept that puts forward a technique of computing the relative contribution of each feature to the output produced by the model [41].

Fig. 15 presents the class-wise SHAP value usage for the stacking ensemble's meta-classifier, a multilayer perceptron (MLP) trained on the outputs of the ten constituent base learners rather than on the raw textual features. The horizontal axis is labelled “Mean Positive SHAP Value (Average Contribution Toward Predicted Class),” explicitly indicating that the quantity plotted is a derived summary statistic rather than a raw Shapley value, while the vertical axis, labelled “Predicted Class,” enumerates the two possible ensemble outputs, each identified unambiguously by both its numeric encoding and its semantic label—0 (Negative) and 1 (Positive)—with the corresponding value annotated at the terminus of each bar. The features underlying this analysis are the twenty meta-features supplied to the MLP, namely the predicted probability of the negative class and the predicted probability of the positive class generated by each of the ten base classifiers for a given review; these values were computed and aggregated over a stratified, class-balanced sample of 300 held-out test instances (150 negative, 150 positive) that were excluded from the training of both the base learners and the meta-classifier. Given the intractability of exact Shapley-value enumeration over a twenty-dimensional feature space, SHAP values were estimated using a permutation-sampling approximation, in which sixty randomly ordered feature-inclusion sequences were generated per instance, features were incrementally

reintroduced from a baseline defined as the mean meta-feature vector of the training set, and the resulting marginal changes in predicted class probability were averaged to yield each feature's estimated contribution. Because the SHAP values corresponding to the negative and positive classes are, by construction, exact negatives of one another in a binary setting, the mean absolute SHAP value would be identical across both classes and therefore uninformative; the reported statistic instead retains only the positive component of each class's SHAP values (i.e., $\max(\text{SHAP}, 0)$), which permits the two class-wise magnitudes to diverge and reveals that, on average, the base learners' collective output exerts a marginally stronger influence toward the positive class (0.0072) than toward the negative class (0.0066).

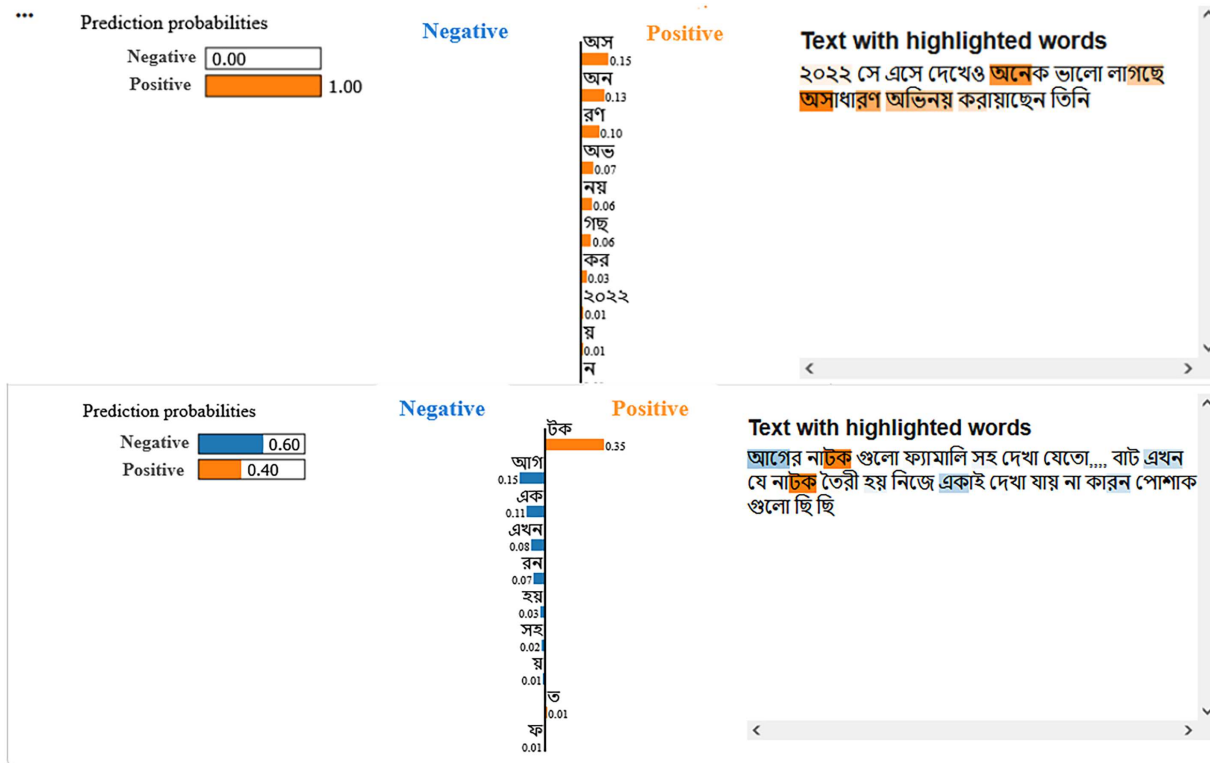


Figure 14: LIME explanation of a Bangla movie review.

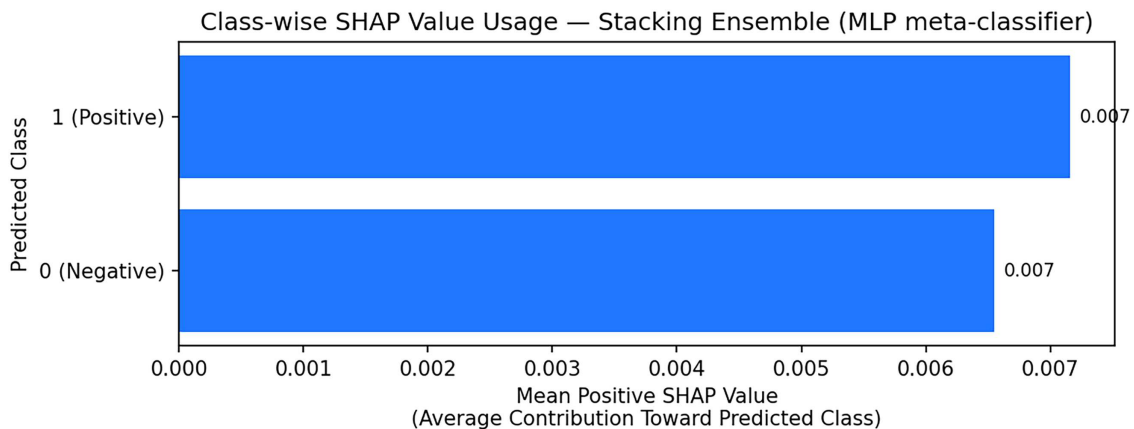


Figure 15: Class-wise SHAP value usage (Stacking ensemble, MLP meta-classifier).

Fig. 16 depicts the full, unaggregated distribution of these SHAP values in the form of a class-wise beeswarm grid, again derived from the stacking ensemble's MLP meta-classifier. The figure is organized as a two-by-two matrix in which the column headings ("Negative (0)," "Positive (1)," under the shared label "True sample class") denote the ground-truth label of the samples contained within each panel, and the row headings (under the shared label "SHAP output class") denote which class's SHAP values are being visualized; each panel is thus uniquely defined by the intersection of the explained class and the true class of the underlying samples. A shared color bar, labeled "Feature contribution (SHAP value)," encodes the sign and magnitude of each point using a diverging blue-to-red scale, and a vertical reference line at zero demarcates the boundary between negative and positive contributions within every panel. As in the preceding figure, the underlying features are the twenty base-learner-derived meta-features rather than lexical features, and each panel aggregates every (sample, meta-feature) SHAP value pair drawn from the same 300-instance held-out test set, subject to the panel's column-defined class restriction, yielding up to 3000 plotted points per panel (150 samples $\times$ 20 meta-features). Unlike the usage plot, no averaging is applied here: each individual value produced by the permutation-sampling procedure described above is rendered as a discrete, jittered point superimposed on a kernel-density violin outline, thereby preserving the shape, spread, and skewness of the underlying distribution rather than collapsing it to a single summary statistic.

Class-wise SHAP Value Distribution — Stacking Ensemble (MLP meta-classifier)

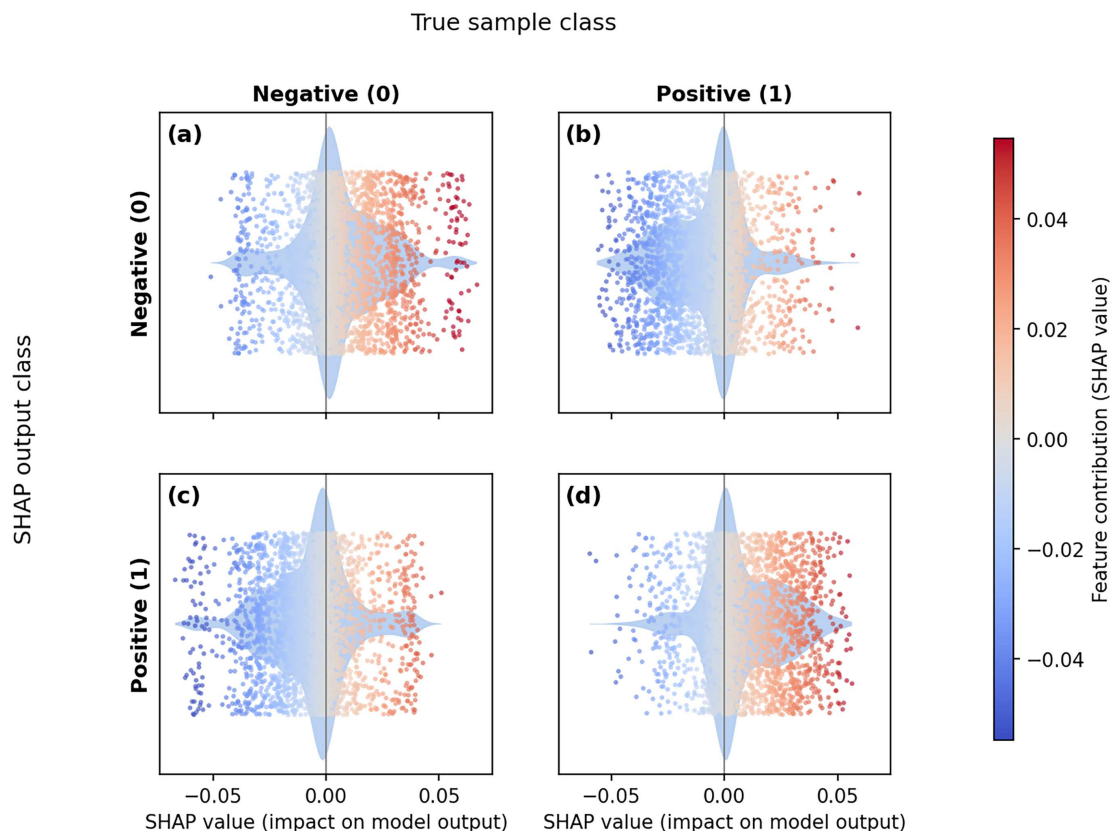


Figure 16: Class-wise SHAP Beeswarm grid (Stacking ensemble, MLP meta-classifier). (a) SHAP values for the Negative output class computed on samples whose true label is Negative (concordant/diagonal). (b) SHAP values for the Negative output class computed on samples whose true label is Positive (discordant/off-diagonal). (c) SHAP values for the Positive output class computed on samples whose true label is Negative (discordant/off-diagonal). (d) SHAP values for the Positive output class computed on samples whose true label is Positive (concordant/diagonal). The two diagonal panels (a,d) show a marked skew toward positive SHAP values, consistent with a well-calibrated meta-classifier.

The two diagonal panels—Negative-output paired with negative-truth, and Positive-output paired with positive-truth—exhibit a marked skew toward positive SHAP values, providing a visual confirmation that the meta-classifier’s explanations are consistent with, and reinforce, the ground-truth labels of the samples, whereas the off-diagonal panels display the inverse pattern, as would be expected under a well-calibrated model.

Unlike Figs. 15 and 16, which analyze the internal behavior of the MLP meta-classifier in terms of its 20 base-learner-derived meta-features, Fig. 17 explains a character n-gram TF-IDF logistic-regression surrogate distilled to approximate the full stacking ensemble pipeline’s behavior on the same review shown in Fig. 14, using SHAP’s LinearExplainer with a 500-review background sample. The large number of aggregated features (296,454) reflects the size of this character n-gram vocabulary rather than the 20-dimensional meta-feature space used in Figs. 15 and 16. The surrogate’s $f(x) = 1.117$ and $E[f(X)] = 0.567$ are its raw decision-function output (log-odds), not the stacking ensemble’s own predicted probability.

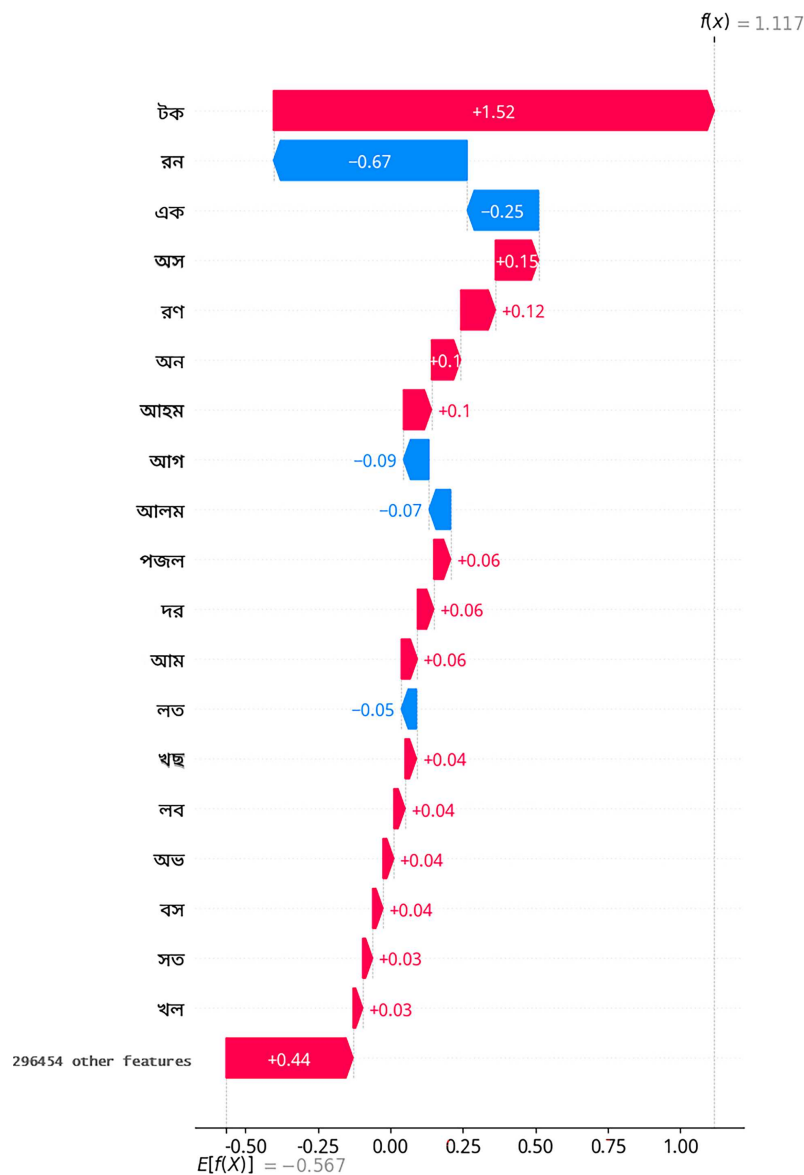


Figure 17: Waterfall plot of an instance for the stacking model.

Positive SHAP values (shown in red) increase the predicted output, whereas negative SHAP values (shown in blue) decrease it. Among all features, “টেক” exhibits the highest positive contribution (+1.52), indicating its dominant role in driving the prediction. In contrast, “রন” (−0.67) and “এক” (−) (−0.25) have the strongest negative influence on the model output. Most features alone provide small SHAP values, but the combined effect of the 296,454 other features (+0.44) is large. This demonstrates that the proposed model successfully incorporates a few important features and many minor ones, ensuring both interpretability and robustness of the prediction.

4.4 Results Comparison

It is essential to have a thorough knowledge of the performance of a model in previous studies and trends. The comparison of existing works and proposed works is shown in Table 34 based on several parameters such as year, best model, accuracy (%), statistical test, and explainable AI. We have applied 22,289 movie reviews from Bangla movies in our proposed model. The data set used is of equal or greater size/scope as that used in earlier research. Having such a set of data significantly increases the model’s ability to make generalizations and see patterns. The proposed stacking model achieves 94.91% accuracy (F1-score 0.95) under 10-fold cross-validation and 94.58% accuracy under an independent 80/20 holdout split. The Friedman test entry refers to the fold-level omnibus test with Holm-corrected pairwise post-hoc comparisons (Tables 25 and 26); the Wilcoxon entry refers to both the fold-level Champion comparison (Table 26) and a supplementary five-seed robustness check (Table 33). The results obtained are at least as good as, if not better than, those reported in previous work on Bangla sentiment analysis.

Table 34: Comparison between existing literature and the proposed methodology for sentiment analysis of bangla film reviews. Accuracies for existing works are as reported in their original studies, while the proposed method’s accuracy is reported under both holdout and 10-fold cross-validation protocols.

Reference	Year	Dataset	Best Model	Accuracy (%)	Statistical Test	Explainable AI
[4]	2019	4000	SVM	88.90	No	No
[5]	2021	4000	LSTM	84.25	No	No
[3]	2021	3000	MNB	86.00	No	No
[2]	2023	1000	RF	92.35	No	No
Our Proposed	2026	22,289	Stacking	94.91 (10-fold CV) 94.58 (holdout)	Yes (Friedman Test & Wilcoxon Signed Rank Test)	Yes (LIME & SHAP)

5 Limitations and Future Works

This study includes only binary sentiment analysis of movie reviews, but there are several types of sentiments that are not included. Other domains can be used to collect movie reviews for this work, but only YouTube has been used for this one, so that the data collected is not as diverse and robust as it could be. This research primarily centers on explicit sentiments expressed within the Bangla movie reviews and fails to address the subtleties of sentiment that can be expressed in Bangla, including sarcasm, irony, or mixed sentiments, which may involve more in-depth analysis and interpretation of context and culture. For some models, notably LSTM, the performance on cross-validation is much worse than with holdout methods.

Because splitting was performed at the review level rather than grouped by movie and because the dataset originates from only 35 source movies, individual folds may still contain uneven proportions of movie-specific vocabulary, style, and review difficulty, which could contribute to this performance difference across folds. In future work, movie-level (group-wise) splitting should be adopted to provide a stricter generalization test to entirely unseen movies.

Additionally, because the train-test and cross-validation splits were performed at the review level rather than by movie, reviews from the same movie could appear in both training and evaluation sets. This may allow models to partly learn movie-specific vocabulary or naming patterns, which could modestly inflate the reported accuracy relative to a strictly movie-disjoint evaluation.

A related limitation concerns the vocabulary coverage of the BERT-based checkpoints. BERT and Distil-BERT were fine-tuned from English-vocabulary checkpoints, whereas only mBERT and Bangla-BERT were pretrained on corpora that natively include Bangla script, causing the English tokenizers to fragment Bangla words into more subword pieces or out-of-vocabulary fallbacks. Consequently, the accuracy differences reported in [Tables 20–22](#) partly reflect this vocabulary mismatch rather than architecture alone, and comparisons between BERT and mBERT/Bangla-BERT should not be read as evidence that BERT is inherently less suited to Bangla sentiment classification. Future work should use Bangla-adapted or vocabulary-extended versions of BERT and Distil-BERT, or report subword fragmentation statistics, to isolate the effect of tokenizer vocabulary from architecture.

A further limitation is that this study used fixed hyperparameter settings ([Tables 7–13](#)), based on values common in prior literature and preliminary trials, rather than a systematic hyperparameter search. Overfitting was mitigated through early stopping, dropout, L2 regularization, and evaluation on both a held-out test set and 10-fold cross-validation, but the reported settings are not guaranteed to be optimal for each model—particularly the deep learning architectures that showed high variance across folds (e.g., LSTM, Bi-LSTM, CNN-BiLSTM, CNN-BiGRU). Future work should apply a systematic search (e.g., grid or random search) conducted within each outer training fold (nested cross-validation), which would give each model its own optimal configuration and remove any risk of tuning-evaluation leakage.

A further limitation concerns the statistical testing procedure itself. While the fold-level Friedman test ([Table 25](#)) is an appropriate omnibus test for detecting overall differences among models, the Holm correction applied to the large number of pairwise comparisons within the deep learning family (55 pairs for $k = 11$ models) substantially reduces statistical power, so that no individual pairwise comparison—including stacking vs. any single deep learning architecture—remained significant after correction, despite a significant omnibus result. This is a known trade-off of strict family-wise error control and does not necessarily indicate the absence of a true difference. Future work could pre-register a smaller set of comparisons of primary interest, or use a less conservative multiple-comparison procedure (e.g., Benjamini–Hochberg false discovery rate control), to increase power while still adjusting for multiplicity. Additionally, the five-seed one-sided Wilcoxon comparison between the stacking ensemble and RNN ([Table 33](#)) is based on only five paired observations and should be regarded as a limited robustness check rather than a definitive or generalizable claim of superiority; a larger number of independent seeds or repeated nested cross-validation runs would provide stronger evidence.

The six external-dataset experiments ([Tables 30 and 31](#)) used the same fixed hyperparameters as the proposed dataset, with no per-dataset tuning. This isolates how well a fixed configuration transfers across domains, so the results should be read as evidence of relative robustness under a controlled, documented protocol—not as an absolute claim of state-of-the-art transfer performance. Future work should repeat this analysis with per-dataset hyperparameter tuning.

Explainable AI techniques were applied only to representative examples rather than systematically across the full datasets. The model performance may differ when it is used for other domain of Bangla text. This work is examined only in Bangla language, but mixing of language in sentiment analysis can be conducted as it is a common practice nowadays.

These limitations will be addressed in future studies by adding sarcasm-aware sentiment model, adding aspect-level sentiment model, quantitative evaluation of explainability, human evaluation of explainability and applying more cross-domain testing with different Bangla datasets.

6 Conclusion

Numerous companies aim to market movies, and analyzing the sentiments expressed in movie reviews is crucial to their marketing goals. The analysis of movie reviews is still in its infancy in the context of Bangladesh because data are not readily available. This article describes the creation of a novel, large-scale dataset of 22,289 reviews from the YouTube platforms, in the context of Bangla films. The intent is to partly resolve the issue of scarcity of resources for the Bangla language. The main aim of this study is to find out whether the attitudes (positive or negative) are present in comments regarding Bangla films or not. This study's empirical analysis used six different machine learning algorithms, ten deep learning models, a stacking ensemble model built from the ten deep learning models as base classifiers, and BERT-based transfer learning. For transfer learning, four models were used, namely BERT, mBERT, Bangla-BERT, and Distil-BERT. The performance of the stacking ensemble model was outstanding, achieving an accuracy of 94.91% under 10-fold cross-validation, and 94.58% under the holdout split, outperforming the rest of the models deployed under each respective protocol. Fold-level Friedman tests confirmed significant overall differences among models within each family, and Holm-corrected pairwise Wilcoxon post-hoc tests showed that Random Forest and mBERT significantly outperformed most individual competitors within their respective families, and that the stacking ensemble significantly outperformed the best representative model of every competing family (RF, RNN, and mBERT). Pairwise superiority of stacking over each individual deep learning architecture did not survive Holm correction, a consequence of the large number of comparisons within that family rather than evidence against the ensemble's benefit. A supplementary one-sided Wilcoxon signed-rank test across five random seeds provided additional, though statistically limited, support that the stacking ensemble consistently outperformed the RNN model under varying initializations. We combined Local Interpretable Model-agnostic Explanations (LIME) and Shapley Additive eXplanations (SHAP) techniques, both rooted in explainable AI, to increase the explanatory power of a model. In the future, we plan to explore several hybrid feature extraction methods and advanced algorithms to analyze bilingual movie reviews.

Acknowledgement: The authors extend their thanks to the Ongoing Research Funding Program (ORF-2026-1886), King Saud University, Riyadh, Saudi Arabia.

Funding Statement: This work was supported by the Ongoing Research Funding Program (ORF-2026-1886), King Saud University, Riyadh, Saudi Arabia.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Md. Shymon Islam, M. Raihan and Anupam Kumar Bairagi; methodology, Md. Shymon Islam, S. M. Reazul Islam Rifat, Md. Mahmudul Haque, Kamrul Golder, Riaj Hossen, Sakib Ahmed and Effa-Bassey Ettah; software, Md. Shymon Islam, Kamrul Golder, Taher Al-Shehari, Effa-Bassey Ettah and Mohammed Kadrie; validation, Md. Shymon Islam, M. Raihan, Kamrul Golder and Anupam Kumar Bairagi; formal analysis, Md. Shymon Islam, S. M. Reazul Islam Rifat, Riaj Hossen and Mohammed Kadrie; investigation, Md. Shymon Islam, M. Raihan, Md. Mahmudul Haque, Kamrul Golder, Taher Al-Shehari and Anupam Kumar Bairagi; resources, Md. Shymon Islam, M. Raihan, Taher Al-Shehari and Anupam

Kumar Bairagi; data curation, Md. Shymon Islam, M. Raihan, S. M. Reazul Islam Rifat, Kamrul Golder, Riaj Hossen and Sakib Ahmed; writing—original draft preparation, S. M. Reazul Islam Rifat, Md. Mahmudul Haque, Kamrul Golder, Riaj Hossen and Sakib Ahmed; writing—review and editing, Md. Shymon Islam, M. Raihan, Md. Mahmudul Haque, Kamrul Golder and Anupam Kumar Bairagi; visualization, Md. Shymon Islam, S. M. Reazul Islam Rifat and Kamrul Golder; supervision, M. Raihan, Mohammed Kadrie and Anupam Kumar Bairagi; project administration, Md. Shymon Islam, M. Raihan and Anupam Kumar Bairagi; funding acquisition, Taher Al-Shehari, Effa-Bassey Ettah, Mohammed Kadrie and Anupam Kumar Bairagi. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: All relevant data generated during this study are reported in the manuscript, and the complete dataset is publicly available in the Zenodo repository <https://doi.org/10.5281/zenodo.22106939>.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Islam MS, Aun SI, Mou MR, Nitu SM, Habibullah M. Analyzing public sentiments from Bangladeshi tour and travel Vlog video comments using machine learning. In: Proceedings of the 2024 International Conference on Innovations in Science, Engineering and Technology (ICISSET); 2024 Oct 26–27; Chittagong, Bangladesh. p. 1–6.
2. Jahan H, Arman MS, Hasan A, Bristy SN. A machine learning-based approach for sentiment analysis of movie reviews on a Bangladeshi OTT platform. In: International Conference on Intelligent Computing & Optimization (ICO), LNNS. Cham, Switzerland: Springer; 2023. p. 339–49.
3. Rahman R, Masud MA, Mimi RJ, Dina MNS. Sentiment analysis on Bengali movie reviews using Multinomial Naïve bayes. In: Proceedings of the 2021 24th International Conference on Computer and Information Technology (ICCIT); 2021 Dec 18–20; Dhaka, Bangladesh. p. 1–6.
4. Chowdhury RR, Hossain MS, Hossain S, Andersson K. Analyzing sentiment of movie reviews in Bangla by applying machine learning techniques. In: Proceedings of the 2019 International Conference on Bangla Speech and Language Processing (ICBSLP); 2019 Sep 27–28; Sylhet, Bangladesh. p. 1–6.
5. Saha P, Sultana N. Sentiment analysis from Bangla text review using feedback recurrent neural network model. In: Communication and Intelligent Systems: Proceedings of ICCIS. Singapore: Springer; 2021. p. 423–34.
6. Lin A, Liu H, Wu H. A bi-directional flow weighted regression for interpreting social media sentiment identified by large language models. *Int J Geogr Inf Sci*. 2026;40(4):1189–221. doi:10.1080/13658816.2025.2555582.
7. Chattu K, Reddy KAN, Veesam S, Chirumamilla PS, Babu DV, Prakash K, et al. Sentiment classification for telugu using transformed based approaches on a multi-domain dataset. *Sci Rep*. 2025;15(1):22185. doi:10.1038/s41598-025-05703-9.
8. Rifa UA, Debnath P, Rafa BK, Hridi SS, Rahman MA. CineXDrama: relevance detection and sentiment analysis of Bangla Youtube comments on movie-drama using transformers: insights from interpretability tool. *arXiv:2411.06548*. 2024.
9. Mirza JF, Shafi MMS, Moheb SH. Comparison of machine learning for Sentiment analysis in movie reviews. *J FST*. 2025;3(1):147–70. doi:10.64494/jfst/v3i1/jm/2025/07/147-170.
10. Sharma H, Pangaonkar S, Gunjan R, Rokade P. Sentimental analysis of movie reviews using machine learning. *ITM Web Conf*. 2023;53(1–2):02006. doi:10.1051/itmconf/20235302006.
11. Ramadhan NG, Ramadhan TI. Analysis Sentiment based on IMDB aspects from movie reviews using SVM. *Sinkron*. 2022;7(1):39–45. doi:10.33395/sinkron.v7i1.11204.
12. Zumma MT, Tahmiduzzaman KBM, Khan R, Roni NS, Rahat-Bin-Rafique AAM, Akash AH. Sentimental analysis of movie review using machine learning approach. In: Proceedings of the 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET); 2022 Dec 23–24; Bhopal, India. p. 1–5.
13. Zhang F, Zeng Q, Lu L, Li Y. Sentiment analysis of movie reviews based on deep learning. *J Phys Conf Ser*. 2021;1754(1):012234. doi:10.1117/12.2675234.

14. Yaseen M, Tedmori S. Movies reviews Sentiment analysis and classification. In: Proceedings of the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT); 2019 Apr 9–11; Amman, Jordan. p. 860–5.
15. Gong Y, Liu G, Xue Y, Li R, Meng L. A survey on dataset quality in machine learning. *Inf Softw Technol.* 2023;162(2):107268. doi:10.1016/j.infsof.2023.107268.
16. Zipf GK. Human behavior and the principle of least effort. Reading, MA, USA: Addison-Wesley Press; 1949.
17. Islam MS, Imon E, Raihan M, Noor MM, Rasa TJ, Ema IAH. Analyzing Bangla drama and TV series reviews using stacking method. In: Proceedings of the 3rd International Conference on Computing Advancements; 2024 Oct 27–28; Dhaka, Bangladesh. p. 55–62.
18. Protick MI. Bangla-stemmer. Python Package Index (PyPI); 2020. Version 1.0. [cited 2026 Aug 26]. Available from: <https://pypi.org/project/bangla-stemmer/>.
19. Kamal R. Bangla-stemmer. 2018 [cited 2026 Aug 26]. Available from: <https://github.com/rafi-kamal/Bangla-Stemmer>.
20. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781. 2013.
21. Lebrecht RP. Word embeddings for natural language processing [dissertation]. Lausanne, Switzerland: EPFL; 2016.
22. Dewi C, Tsai BJ, Chen RC. Shapley additive explanations for text classification and Sentiment analysis of internet movie database. In: Asian Conference on Intelligent Information and Database Systems. Singapore: Springer; 2022. p. 69–80.
23. Singh G, Kumar B, Gaur L, Tyagi A. Comparison between multinomial and Bernoulli Naïve bayes for text classification. In: Proceedings of the 2019 International Conference on Automation, Computational and Technology Management (ICACTM); 2019 Apr 24–26; London, UK. p. 593–6.
24. Li Z, Liu F, Yang W, Peng S, Zhou J. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Trans Neural Netw Learn Syst.* 2022;33(12):6999–7019. doi:10.1109/tnnls.2021.3084827.
25. Graves A. Long short-term memory. In: Supervised sequence labelling with recurrent neural networks. Berlin/Heidelberg, Germany: Springer; 2012. p. 37–45.
26. Long F, Zhou K, Ou W. Sentiment analysis of text based on bidirectional LSTM with multi-head attention. *IEEE Access.* 2019;7:141960–9. doi:10.1109/access.2019.2942614.
27. Du J, Vong CM, Chen CP. Novel efficient RNN and LSTM-like architectures: recurrent and gated broad learning systems and their applications for text classification. *IEEE Trans Cybern.* 2021;51(3):1586–97. doi:10.1109/tcyb.2020.2969705.
28. Zhang D, Tian L, Hong M, Han F, Ren Y, Chen Y. Combining convolution neural network and bidirectional gated recurrent unit for sentence semantic classification. *IEEE Access.* 2018;6:73750–9. doi:10.1109/access.2018.2882878.
29. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional Transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers); 2019 Jun 2–7; Minneapolis, MN, USA. p. 4171–86.
30. Alaparthi S, Mishra M. Bidirectional encoder representations from transformers (BERT): a sentiment analysis odyssey. arXiv:2007.01127. 2020.
31. Kowsher M, Sami AA, Prottasha NJ, Arefin MS, Dhar PK, Koshiba T. Bangla-BERT: transformer-based efficient model for transfer learning and language understanding. *IEEE Access.* 2022;10(8):91855–70. doi:10.1109/access.2022.3197662.
32. Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108. 2019.
33. Wolpert DH. Stacked generalization. *Neural Netw.* 1992;5(2):241–59. doi:10.1016/s0893-6080(05)80023-1.
34. Golder K, Islam MS, Tarun FF, Biswas P, Akter S. An empirical study on developing stacking ensemble model for Bangla sports sentiment analysis. In: Proceedings of the 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT); 2024 Jun 24–28; Kamand, India. p. 1–7.

35. Van Rijn JN, Hutter F. Hyperparameter importance across datasets. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018 Aug 19–23; London, UK. p. 2367–76.
36. Zimmerman DW, Zumbo BD. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. *J Exp Educ.* 1993;62(1):75–86. doi:10.1080/00220973.1993.9943832.
37. Islam MS, Fariha SAS, Snaha SS, Raihan M, Mostafa MS, Abrar SF, et al. An interpretable sentiment recognition method from political Bangla texts using stacking ensemble model. In: Proceedings of the 2025 International Conference on Electrical, Computer and Communication Engineering (ECCE); 2025 Feb 13–15; Chittagong, Bangladesh. p. 1–6.
38. Mostafa MS, Islam MS. Recognizing emotions from crime video comments based on pretrained language models and Explainable AI. In: Proceedings of the 2024 27th International Conference on Computer and Information Technology (ICCIT); 2024 Dec 20–22; Cox's Bazar, Bangladesh. p. 2309–14.
39. Islam MS, Khan AN, Dev DD, Raihan M, Al Mubid S, Mostafa MS. Stacked-KBR: an ensemble method for analyzing YouTube Bangla song reviews using a novel Dataset. In: Proceedings of the 2024 27th International Conference on Computer and Information Technology (ICCIT); 2024 Dec 20–22; Cox's Bazar, Bangladesh. p. 1522–7.
40. Ribeiro MT, Singh S, Guestrin C. “Why should i trust you?”: explaining the predictions of any Classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA. p. 1135–44.
41. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. p. 4768–77.