International Journal of Information Management Data Insights 2 (2022) 100095

Contents lists available at ScienceDirect

International Journal of Information Management Data Insights

journal homepage: www.elsevier.com/locate/jjimei

How can we manage Offensive Text in Social Media - A Text Classification Approach using LSTM-BOOST☆

Md. Anwar Hussen Wadud[a], Muhammad Mohsin Kabir[a], M.F. Mridha[b,*], M. Ameer Ali[a], Md. Abdul Hamid[c], Muhammad Mostafa Monowar[c]

[a] Department of Computer Science & Engineering, Bangladesh University of Business & Technology, Dhaka, Bangladesh
[b] Department of Computer Science & Engineering, American International University Bangladesh Corresponding Author's Secondary, Dhaka, Bangladesh
[c] Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah-21589, Kingdom of Saudi Arabia

ARTICLE INFO

Keywords:
Natural Language Processing
Long Short Term Memory
Adaptive Boosting
Ensemble Learning
Abusive Text

ABSTRACT

Recently, offensive content has become increasingly popular for harassing and criticizing people on numerous social media platforms. This paper proposes an offensive text classification algorithm named LSTM-BOOST employing Long Short-Term Memory(LSTM) model with ensemble learning to recognize offensive Bengali texts in various social media platforms. The proposed LSTM-BOOST model uses the modified AdaBoost algorithm employing principal component analysis(PCA) along with LSTM networks. In the LSTM-Boost model, the dataset is divided into three categories, and PCA and LSTM networks are applied to each part of the dataset to obtain the most significant variance and reduce the weighted error of the weak hypothesis of the model. Furthermore, different classifiers are used for baseline experiment and the model is evaluated on various word embedding vector methods. Our investigation found that the LSTM-BOOST algorithms outperform most of the baseline architecture, leading F1-score of 92.61% on the Bengali offensive text from Social Platforms(BHSSP) dataset.

1. Introduction

People of this modern world express their views and beliefs through social media platforms, making these a vast source of text data. For example, the current monthly active Facebook users are 2.8 billion (Facebook, 2021), and the daily active users of Twitter are 192 million (Twitter, 2021). Administering abusive text among all user-generated text online is an onerous responsibility. However, most tweets on Twitter or posts, comments, or messages on Facebook are inoffensive and knowledgeable. Still, few can be offensive towards an attacked users, individuals, or minority groups (Santos et al., 2018). Specific contents are meant to be defamatory, embarrassing, or disrespectful and are designated as offensive text (Davidson et al., 2017). Unlike assaulting text, abusive content is generally exposed towards society classes such as religion, gender, disability, ethnic origin, or sexual orientation (Founta et al., 2018).

Abusive languages are such information that contains mean/dirty/spam word/phrase commonly comes from displeasing circumstances such as mental illness, sexual divergence, physical disability, absence of modernization (Colladon and Gloor, 2019) (Batra et al., 2021). On the contrary, offensive language might be an abusive word or a typical information that inadequately offend someone. Using offensive textual information negatively impacts both individuals who use the offensive text and those who are being attacked. An individual who uses offensive textual information in his words might be raised stress, anxiety, depression, and a fewer sense of belonging. Similarly, the mental stability of the affected person is drastically reduced. Hence the social life of both of these two people becomes abnormal.

Another essential element of the evolution of the information system is the growing number of stored offensive and non-offensive data available. The only way to handle this huge amount of offensive and non-offensive data is to use intelligent automation. NLP technologies and linguistic theories are being used to design and implement more user-centric communication and textual information systems. As a result, new approaches and technologies must involve in information management.

Around 245 million people speak Bengali as their native language, presenting it as the 7th most spoken language in the world (Mandal and Sen, 2014). So, billions of multi-media texts are produced every day in the Bengali language through social network platforms. But, scientific investigation on Bengali Language Processing (BLP) is still in its primary stage, and very few works have been done on Bengali offensive

☆ Preprint submitted to International Journal of Information Management Data InsightsMarch 13, 2022

* Corresponding author.
E-mail addresses: mwadud@bubt.edu.bd (Md.A.H. Wadud), mdmkabi@gmail.com (M.M. Kabir), firoz@bubt.edu.bd (M.F. Mridha), dmaa730@gmail.com (M.A. Ali), mabdulhamid1@kau.edu.sa (Md.A. Hamid), mmonowar@kau.edu.sa (M.M. Monowar).

https://doi.org/10.1016/j.jjimei.2022.100095
Received 28 October 2021; Received in revised form 21 June 2022; Accepted 21 June 2022

Table 1
Types of offensive text and examples in the Bengali language.

Categories	Example of possible targets
Race	কাইল্লা (Black people), ফার্মের মুরিগ (White people)
Behavior	ভোদাই (Fool), বলদ(Fool as cow)
Physical	বাইট্টা (Short people), মোটা (Fat people)
Sexual Orientation	গে (Gay people), লেসবিয়ান (Lesbian)
Class	ফিকিন্ন(Poor people), রিচকিড (Rich people)
Gender	হিজড়া (Eunuch), মেয়েখোর (Play boy)
Ethnicity	রোহিঙ্গা (Rohingyas), চাকমা(Ethnic people)
Disability	লুলা(Autistic), কানা(Blind people)
Religion	মালাউন(Hindu people), ইহুদির বাচ্চা(Jewish people)
Revile	মাগি (Bitch), বেশ্যা (Prostitute)
Addiction	মদখোর (Drunk people), গাঞ্জাখোর (Drug addict)
Others	শয়তানের বাচ্চা(Son of devil), গুটিবাজ (Conspirator)

text classification (Karim, 2013) (Afroz et al., 2021). As millions of people express their feelings through this language, different categories of offensive text content are created daily. The types of offensive text and examples in the Bengali language are presented in Table 1.

Nevertheless, a number of well-known and prominent institutions made some noteworthy attempts on automatic abusive and hateful content classification systems (Razavi et al., 2010). In June 2017, Instagram proclaimed an innovative mechanism toward assaulting posts and comments (Systrom, 2017). Googles Jigsaw incubator has revealed a cutting-edge Application Programming Interface (API) named Perspective1, impersonating text input a percentage of how likely it is to be recognized as hateful. However, these text classification tasks solely depend on text analyzing techniques such as Machine Learning and Natural Language Processing (NLP) algorithms (Pitsilis et al., 2018) (Mridha et al., 2021) (Nobata et al., 2016) (Garg et al., 2021). Several Deep Learning (Sharma et al., 2021) (Maqsood et al., 2020) algorithms are currently applying in this domain to enhance the efficiency of Machine Learning approaches such as BERT: Pre-training of Deep Bidirectional Transformers, Recurrent Neural Network (RNN) (Nasir et al., 2021)(Wadud et al., 2022), Convolutional Neural Network (CNN) (Nasir et al., 2021), Long Short-Term Memory (LSTM), etc. But these text classification approaches have few significant limitations that restrict the efficiency of abusive content recognition tasks. Amongst them, few models and their limitations are presented in Table 2.

Besides, the complexity of automated offensive text classification differs amongst several languages because of certain forms, the number of characters, strokes, and character similarity (Pradhan et al., 2020). The Bengali language contains the following features that are deeply related to the text recognition tasks:

Vowel: Vowels are characters or letters representing the vocal sound produced only with an open vocal tract. In the Bengali language, there have eleven (11) vowels.

Consonant: Consonants are characters or letters that convey a spoken sound that may be pronounced with fully or partially open vocal cords. The Bengali language has 39 consonants.

Diacritic: The signals are formulated over or under a symbol to illustrate a distinct articulation of a similar character. There are two kinds of diacritics in the Bengali language: consonant diacritic and vowel diacritic. The Bengali language has seven consonant diacritics and 11 vowel diacritics.

Consonant Conjuncts: Consonant conjuncts are letters that include two or more consonants connected. In Bengali, there are 118 consonant conjuncts that are essentially practiced.

Grapheme: The shortest portion of any writing style is a grapheme. A grapheme in Bengali must have a grapheme root. A grapheme root is a vowel, consonant, or consonant conjunct character. Diacritics can also be seen in graphemes. An illustration of a Bengali word construction is shown in Fig. 1.

Table 2
Limitations of existing text classification approaches for offensive text classification task.

Model	Architecture	Limitations
Logistic Regression (Genkin et al., 2007)	Bayesian Logistic Regression	Prediction results are based on a set of independent variables
Naive Bayes (Kim et al., 2006)	Multivariate Poisson model for text Classification	Construct a strong assumption about the shape of the data distribution
Support Vector Machine (Lodhi et al., 2002)	String Subsequence Kernel	Lack of transparency in the results
SVM and KNN (Chen et al., 2016)	Inverse Gravity Moment	Fails to capture polysemy and still semantic is not solved
Conditional Random Field (CRF) (Chen et al., 2017)	BiLSTM-CRF	High computational complexity
Deep Learning (Chen et al., 2020)	Deep Neural Networks	Data dependent for designed a model architecture
Deep Learning (Jiang et al., 2018)	Deep Belief Networks	Computationally expensive and model interpretability
Deep Learning (Zhang et al., 2015)	CNN	This model only discover position-invariant features of their inputs
Deep Learning (Kowsari et al., 2018)	Ensemble deep learning algorithm (CNN, DNN and RNN)	Computationally expensive

The Bengali language includes almost 13,000 different grapheme variations involving many vowels, consonants, consonant conjuncts, and diacritics. Numerous sequences of character patterns are also included in these grapheme variants. Bengali writing structures are more sophisticated than English due to the number of characters combinations, making Bengali offensive text classification challenging. Hence, this study comprehensively investigates the current deep learning architecture for Bengali abusive text classification and proposed a more efficient approach using the ensembled LSTM-ADA Boost approach.

However, this paper focuses on analyzing human-generated information on social platforms using a computerized deep learning model named LSTM-BOOST. Text from different social platforms is extracted, pre-processed, and classified into different types of offensive labels. This article provides an LSTM-ADA Boost ensembled architecture as a Bengali abusive text recognition scheme to identify abusive text on social network platforms. The proposed system is called Ensembled Long term memory- Adaptive Boost (LSTM-BOOST). It is trained with a Bengali

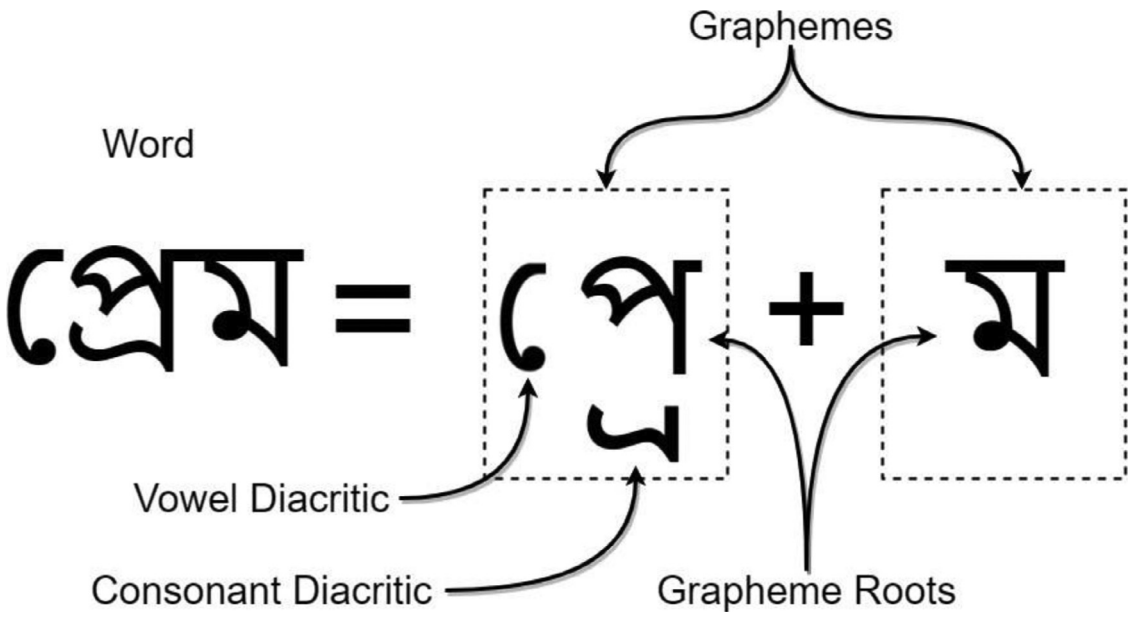


Figure 1. A Bengali word-formation illustration.

dataset containing 20,000 posts, memes, and comments from several Bengali blogs, websites, and social media platforms.

The overall contribution of the article includes:

- We develop a corpus containing 20,000 posts, memes, and comments from several Bengali blogs, websites, and social media platforms named Bengali offensive text from Social Platforms (BHSSP) dataset.
- We modify the LSTM model with PCA for text classification.
- We introduce an Ensembled LSTM-ADA Boost (LSTM-BOOST) system to classify abusive content on social media platforms using modified Adaptive Boost with LSTM and PCA.
- We compare the proposed LSTM-BOOST architecture with several existing text classification approaches.

The rest of the article is organized as follows. Section 2 introduces contemporary architectures of abusive text classification. Section 3 gives a comprehensive explanation of the methodology of the proposed model and Section 4 discuss the detail architecture of the proposed LSTM-BOOST model. Section 5 shows the performance evaluation of the proposed system. Lastly, Section 6 contains the discussion and Section 7 ends the paper.

2. Related Works

Text classification in deep learning is an automated approach to recognize inappropriate content from a vast dataset. Text classification approaches can be divided into three divisions in terms of classifier usage: unsupervised, semi-supervised, and supervised.

- **Supervised learning:** The supervised learning method is domain-dependent as it relies on the hand-operated labeling of the vast dataset. Nevertheless, most of the strategies practiced for abusive text classi- fication tasks are supervised approaches. For example, Burnap and Williams (Burnap and Williams, 2015) have applied various supervised classifiers to distinguish offensive text on Twitter.
- **Semi-supervised learning:** In semi-supervised learning, both labeled and unlabeled data are used to classify text. Hua et al. (Hua et al., 2013) suggested that labeled data combined with unlabeled data can dramatically im- prove text classification. The authors explained that unsupervised learning could not efficiently handle small case events, and supervised learning can do this but need manual labeled data. Hence, the semi- supervised method was required.
- **Unsupervised learning:** Unsupervised learning is a domain-independent method and can handle different content while advancing scalability

(Pandarachalil et al., 2015). It does not rely on human effort to label a high-volume training set; rather, it dynamically extracts domain-related vital features.

Initially, in the abusive text classification domain, feature-engineering- based methods practiced contextual features (Yin et al., 2009). Also, word/character n-grams, brown clusters, linguistic features were primarily involved in this domain (Schmidt and Wiegand, 2019). In the pre-deep learning era, statistical models like Naive Bayes (Davidson et al., 2017) (Razavi et al., 2010) (Kwok and Wang, 2013), Support Vector Machine (Davidson et al., 2017) (Badjatiya et al., 2017)] (Del Vigna12 et al., 2017) (Saleem et al., 2017) (Warner and Hirschberg, 2012) (Xiang et al., 2012), Logistic

Regression (Badjatiya et al., 2017) (Bourgonje et al., 2017) (Djuric et al., 2015) (Waseem and Hovy, 2016), Decision Trees (Davidson et al., 2017) (Bourgonje et al., 2017), Random Forest (Davidson et al., 2017) (Xiang et al., 2012) were mainly used for the text classification task.

With the advent of deep learning, the recent task of text classification domain introduced different deep learning architecture (Chen et al., 2020) (Minaee et al., 2021) (Wadud and Rakib, 2021). Particularly, Natural Language Processing (Palivela, 2021) (Naredla and Adedoyin, 2022) models are the best fit for text classification tasks (Lai et al., 2015) (Kowsari et al., 2019). Founta, A. M.et al. (Founta et al., 2019) suggested a deep learning architecture that includes a wide variety of accessible metadata and combines this with derived hidden patterns inside tweet text to detect various inappropriate behavioral standards that are strongly interconnected. This unified design detects different sorts of inappropriate behavior in a fluid, transparent manner without the need for adjustment. Extracted features are metadata features such as tweet-based, used-based, network-based, and word vectors for this architecture. This architecture used four datasets, as Cyberbullying, Offensive, Hate, Sarcasm datasets, and obtained an accuracy score of 0.92. In (Badjatiya et al., 2017), Pinkesh Badjatiya et al. used a variety of classifiers, including Gradient Boosted Decision Trees (GBDTs), Random Forest, SVMs, Logistic Regression, and Deep Neural Networks (DNNs). In this experiment, the authors used 16k annotated tweets, and the experimented (LSTM + Random Embedding + GBDT) models

performed better than other models. Siwei Lai et al. (Lai et al., 2015) developed an RNN-based model for text classification that did not focus on human-designed traits. While learning word representations, the suggested RNN model captures contextual information that may be less noisy than standard techniques. They employed four well-used datasets to run this model: 20 Newsgroups, Fudan set, ACL Anthology Network, and Stanford sentiment treebank. The authors compared methods, and RCNN outperformed every dataset, respectively 96.49, 95.20, 49.19, and except one dataset. But, the effectiveness of RNNs reduces when the sequence of the text is extensive. Hence LSTM models are applied. Pitsilis, G. K et al. (Pitsilis et al., 2018) used a neural network approach made up of multiple Long Short-Term-Memory (LSTM) based classifiers. This classifier boosts performance by exploiting user behavioral factors, including racism or sexism. An experimental evaluation of the model using a dataset of 16k brief messages from Twitter, displaying the best classification performance attained.

Besides, CNNs based architecture has become very promising for text classification tasks (Nasir et al., 2021). Zampieri, M et al. (Zampieri et al., 2019) proposed a three-level hierarchical annotation scheme to detect and classify abusive language. They used a large-scale dataset, abusive Language Identification Dataset (OLID), an English tweet with high-quality annotation of the target and nature of crimes. Researchers experimented on different machine learning models like SVM, BiLSTM, CNN and compared the performance of the various models. CNN surpassed the RNN model with a macro-F1 value of 0.80; CNN surpassed the BiLSTM with a value of 0.69. Yoon Kim (Chen, 2015) demonstrated a simple CNN with slight hyperparameter adjustment and static vectors where Learning task-specific vectors through fine-tuning provide additional performance. The suggested model examined different CNN model variants using MR, SST-1, SST-2, Subj, TREC, CR, and MPQA datasets. The proposed model was created by training a basic CNN with one layer of convolution on top of word vectors generated by an unsupervised neural language model. The results are performed better for pre-trained CNN models, such as 89.6 for CNN-static, 88.1 for CNN-multichannel on SST-2 dataset, and 85.0 for the CR dataset compared to other variants. In (Wiedemann et al., 2018), a progressively integrated BiLSTM-CNN neural network consisting of three transfer learning methods, Supervised category transfer, semi-supervised category transfer, and Unsupervised category transfer, are used to improve classification performance given prior knowledge. The authors explored the impact of three distinct techniques for mitigating adverse effects during transfer learning and assessed the text classification task on the GermEval 2018 dataset. The performances for the two studies increased with some percentage up to 75.2% and 52.7%, respectively. Gambäck et al. (Gambäck and Sikdar, 2017) developed a deep learning-based text classification system for Twitter hate speech. Each tweet is classified into four predetermined categories by the classifier. Max-pooling was employed to reduce the feature set, and SoftMax function classified tweets. The model is developed based on word2vec embeddings scored best in 10-fold cross- validation, with increased efficiency over recall and a 78.3 percent F-score. Yenala, H. et al. (Yenala et al., 2018) provided a deep learning-based method for automatically detecting such improper language, focusing on solving this problem in two applications: query completion suggestions in search engines and user chats in messengers. Besides, for detecting toxic comments in a multilabel domain, Ashok Kumar et al. introduce a multichannel convolutional bidirectional gated recurrent unit (MCBiGRU). Toxic comments detection refers to the similar tasks of identifying offensive text detection (Kumar et al., 2021).

In addition, recently BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding model is widely using for language representation. This architecture is a finetuning-based approach that pro- duces a state-of-the-art result on an extensive suite of sentence-level, and token-level tasks (Devlin et al., 2018). So, this well-known model is extensively using for text classification and achieving satisfactory results (Sun et al., 2019) (Jin et al., 2020).

Table 3
Dataset Summary.

Attributes Name	Offensive (Co)	Non-offensive (Cn)
Total sentences	10000	10000
Total words	110083	102782
Minimum word in a sentence	1	3
Total Size (in bytes)	990747	1233348

The offensive comments categorization for the English language is a prevalent task. So many datasets are available for English, Hindi, and Spanish. Amongst them, HatebaseTwitter (Davidson et al., 2017), WaseemA (Waseem and Hovy, 2016), WaseemB (Waseem, 2016), Stormfront (De Gibert et al., 2018), TRAC (Facebook) (Kumar et al., 2018), TRAC (Twitter) (Kumar et al., 2018), HatEval (Basile et al., 2019), OffensEval (Zampieri et al., 2019) datasets are very popular. But this model is specially designed for the Bengali language. However, in our work, we have used different machine learning models as baseline architecture. Also, LSTM and PCA have been applied to enhance the performance of the Adaptive Boost ensemble method. Thus, this is the first research work to the best of our knowledge that adopted classical text classification architectures, including machine learning and deep learning to build a state-of-the-art architecture to classify Bengali abusive text on social media platforms.

3. Methodology

This section has introduced the abusive text apprehension algorithm engaging social platform datasets. At first, to train and test the proposed algorithm, we implemented few baseline architectures, including Support Vector Machine (SVM), Logistic Regression (LR), Multinomial Naive Bayes (MNB), Stochastic Gradient Descent (SGD), Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbors (KNN). These architectures are adopted to implement ensemble algorithms of abusive text classification. Figure 3 presents the process of the proposed LSTM-BOOST architecture. First, the data are pre-processed using different pre-processing techniques. Then, TF-IDF vectorizer, count vectorizer, word2vec, and fastText are applied for feature extraction and word matrix generation. Finally, we bundled the AdaBoost architecture with Principal Component Analysis (PCA) in conjunction with LSTM models to create a modified boosting architecture. We use PCA to focus on principal words from the data set and reduce the run time complexity.

3.1. Data Collection

We have used automatic data collection procedures using the data crawling process rather than the manual process. We have gathered data from numerous Bengali online newspapers such as banglanews24.com, kalerkantho.com, prothomalo.com etc. and Facebook and Twitter social platforms. We have accumulated 20000 posts, memes and comments from several Bengali websites. Amongst those, 10000 are offensive posts/comments/memes, and 10000 are non-offensive. The total number of offensive words in the dataset are 110083, and the non-offensive words are 102782. Furthermore, we eliminated all permalinks, user details, dates, and times for higher accuracy. We manually labeled congested datasets to recognize abusive texts. We named the dataset as Bengali offensive text from Social Platforms (BHSSP) dataset. Data have been collected year by year, and we keep track of the sources from which we collected the information. Finally, the dataset is split into 0.8 and 0.2 ratios to train and test the proposed algorithm. The statistics regarding the dataset is presented in Table 3.

3.2. Data Pre-processing

Standard automated text preprocessing involves punctuation, commas, and case conversions. A specific language like Bengali has its own

Table 4
Most frequently use emojis and emoticons for offensive and non-offensive text.

Frequently use Emojis andEmoticons for offensive texts		Frequently use Emojis andEmoticons for non-offensive texts	
Emojis	Emoticons	Emojis	Emoticons
	:(		:D
	:)		:=
	!D		:(
	:<		:)
	d:		:>
	;D		:<

Table 5
Different types of Bengali stop words with examples.

Types	Stop word	Example
Adjective	খুব Very	ছেলটা খুব মোটা আ রকু ৎিসত The boy is very fat and ugly
Pronoun	তিন He/She	তিন খুব মোটা এবং কু ৎিসত He is very fat and ugly
Adverb	ধীরে ধীরে Slowly	জারজকে ধীরে ধীরে মের ফেল Kill the bastard slowly
Conjunction	আ র And	মিনা আ ররিনাকে মের ফেল Kill Mina and Rina

syntactic and grammatical formations that require concocted adopting individual orders and practices. This section describes the text processing techniques we have employed for Bengali text classification. In this paper, we have used the BNLTK (Bnltk 2022) python toolkit to process the Bengali dataset.

3.2.1. Emoji and Emoticon Conversion

Variations of the data, including emoji, emoticons, and words, make text processing difficult. Comments in various social platforms use several sorts of emojis and emoticons usually. Table 4 displays the best six emojis and emoticons related to both offensive and standard text. Peoples usually use emojis or emoticons to express their feelings and these feelings can be offensive too. We convert emoji symbols to textual representations using Beautifulsoup4

(Beautifulsoup4 2022) Python library package. Then, we use the Python Translate tool to translate all emoji messages from English to Bengali (translate, 2022).

3.2.2. Eliminating Hashtags

When commenting on a post on social media (Gkikas et al., 2022) (Aswani et al., 2019), it is common to use hashtags to identify particular content. Considering hashtag texts regarding offensive words may help a lot to classify abusive texts. We remove all hashtag symbols (#) by substituting space and saving all text for future use. For illustration, let someone uses #Hortal hashtag in social media comments; at first, we substitute the (#) symbol by space then interpret it to Bengali #Hortal

=> Hortal => হরতাল. Then, we save ""হরতাল"" for identification and

subsequent processing.

3.2.3. Miscellaneous

We have used different standard text processing (Kumar et al., 2021) approaches, such as accent marks, removing punctuation, white space, converting numbers into words, stop words, etc. There is no accurate collection of stop words for the Bengali language. Haque et al. (ul Haque et al., 2019) divided Bengali stop words into four types. Table 5 presents the four types of Bengali stop words with examples. For the purpose of removing unwanted words and punctuation from the text, we have utilized Python and regex library.

3.2.4. Data Labelling

The data labeling process had been accomplished in two phases. First, all data are labeled manually by eight BUBT Advanced Machine Learning (AML) lab members. Then, another 5 BUBT AML lab members tested and certified the data labels. The whole process is explained in Figure 2.

3.3. Feature Engineering

We have applied TF-IDF (Jie and Li-chao, 2010), Word2vec (Rong, 2014), CountVectorizer (Kulkarni and Shivananda, 2021), and fastText (Bojanowski et al., 2017) to extract features(Wadud et al., 2022). These feature extraction models trans- form the text documents into a numeric vector of fixed length. The working mechanism of these models in our proposed algorithm is briefly explained as follows:

3.3.1. TF-IDF Vectorizer

Repeated text in a document may reduce the efficiency of text classification. Hence TF-IDF model is used in the proposed model to deal with the repeated text. The mathematical representation of the TF-IDF model is:

$$tf\_idf(t,d) := tf(t,d) \times idf(t) \quad (1)$$

where, *idf* (*t*) is inverse document frequency and *tf* (*t, d*) is term frequency.

3.3.2. Word2Vec

It is used to identify the vector of all phrases so that the similar text has related vectors. It is used in the semantic study of textual materials. It is composed of two layers of back propagation networks. It takes as input as a text corpus and produces a set of vectors, which are feature vectors for the words in the corpus. These feature vectors are implemented with the help of the Python scikit-learn package.

3.3.3. CountVectorizer

CountVectorizer is generally applied to transform any contents into a vector representation. It's also used to extract characteristics from time to time. By transforming a set of text documents, a token matrix is formed. The scipy.sparse.csr matrix is used to provide a sparse presentation of the counts. There is no provision for an apriori dictionary, and the analyzer is not utilized.

3.3.4. fastText

fastText is an updated word embedding method of Word2vec. Instead of feeding the individual words to the model directly, it breaks each word as an n-gram of characters. fastText serves excellently with unique words. Word2vec fails to produce any vector representation for words that are not in the training set but fastText can provide.

However, fastText, CountVectorizer, Word2Vec, and TF-IDF feature ex- traction approaches are applied to obtain the vector representations of all text. Table 6 illustrates the feature values of all feature extraction

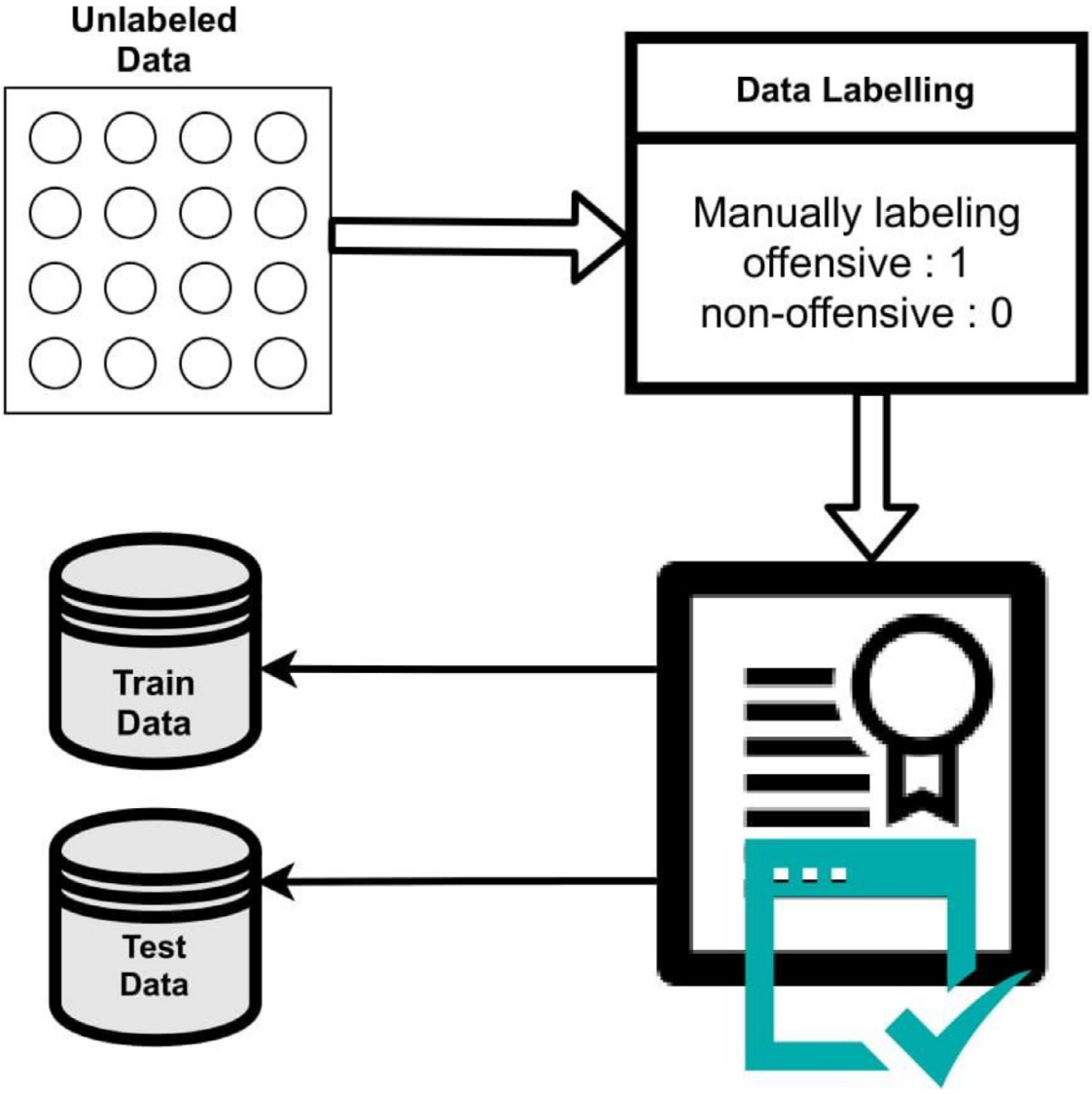


Figure 2. Data Labeling: Labelled and Verification.

Table 6
Sample Feature Values from the top four text pieces with five words.

r\c	VectorModel	*word*	1	*word*	2	*word*	3	*word*	4	$word_5$
Sample Feature Values										
	BoW	4	9		0		12		10	
$text_1$	TF-IDFWord2Vec	0.450.63	0.430.39		0.760.85		0.820.67		0.0030.10	
	fastText	0.74	0.69		0.90		0.85		0.22	
	BoW	8	3		5		2		6	
$text_2$	TF-IDFWord2Vec	0.340.05	0.540.65		0.720.78		0.080.32		0.100.65	
	fastText	0.45	0.52		0.85		0.35		0.55	
	BoW	14	0		7		11		9	
$text_3$	TF-IDFWord2Vec	0.760.89	0.040.1		0.560.45		0.880.70		0.670.73	
	fastText	0.89	0.12		0.52		0.79		0.65	
	BoW	9	5		11		4		6	
$text_4$	TF-IDFWord2Vec	0.670.73	0.720.78		0.880.73		0.450.53		0.560.47	
	fastText	0.77	0.80		0.82		0.53		10	

models by presenting the top five words ($word_1$, $word_2$, $word_3$, $word_4$, $word_5$) from 4 text documents ($text_1$, $text_2$, $text_3$, $text_4$). We have decided on the top 3000 featured words in every text document because we want to keep the size of a two-dimensional array as small as possible. The rows of the array are text documents (t_1, t_2, ..., t_{15000}), and the columns are feature words (w_1, w_2, ..., w_{3000}) of individual text record. Which indicates that we are working with a fixed-length word vector of size 15000×3000.

Word2Vec approach employs several n-gram features for computing word matrix as presented in Table 7. We have applied multiple n-gram charac- teristics with feature extraction methods to recognize offensive text from a document.

Table 7
The representation of the N-gram feature for identifying offensive comments.

Uni-grams	'বলদের', 'বাচ্চা', 'কাজটি', ' ঠিকমত', ' করতে', 'পারল', 'না'
Bi-grams	' বলদের বাচ্চা', 'বাচ্চা কাজটি', 'কাজটি ঠিকমত', ' ঠিকমত করতে', ' করতে পারল', 'পারল না'
Tri-grams	' বলদের বাচ্চা কাজটি', 'বাচ্চা কাজটি ঠিকমত', 'বলেদর বাচ্চা করেত', 'বাচ্চা করতে পারল', ' ঠিকমত করতে পারল', 'বলেদর বাচ্চা না', ' কাজটি করতে না','বাচ্চা কাজটি বলদের '
Quadri-grams	'বলদের বাচ্চা কাজটি করতে', 'বাচ্চা কাজটি ঠিকমত না', 'বলেদর বাচ্চা করতে কাজটি', 'বাচ্চা করতে পারল কাজটি', ' ঠিকমত করতে পারল না', ' কাজটি বলদের বাচ্চা না', ' কাজটি করতে না বলদের', 'বাচ্চা কাজটি বলদের ঠিকমত'
N-grams	" বলদের বাচ্চা কাজটি ঠিকমত করতে পারল না"

Algorithm 1
Proposed LSTM-BOOST Algorithm.

Input: Dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, Base learning algorithm L, Number of learning rounds T.

$D_1(i) = \begin{cases} \frac{1}{2m} & if, y_i = 0 \\ \frac{1}{2n} & if, y_i = 1 \end{cases}$; /* Initialize the weight distribution, Here n and m are numbers of samples for class 0 and 1 respectively */

for $t = 1, 2, ...$ *to* T **do**

$\mu_t = \frac{1}{N}\sum_{i=1}^{N} D_t(i)x_i$

$S_t = \sum_{i=1}^{N} D_t(i)(x_i - \mu)(x_i - \mu)^T$; /* Scatter matrix construction */

$Sv_i = \lambda_i v_i, i = 1...d'$; /* Eigenvalue decomposition */

$V_{pca} = [v_1, v_2, ...v_{d'}] \in \mathbb{R}^{d \times d'}$; /* Projection matrix V_{pca} formation */

$h_t = LSTM(D, V_{pca}, D_t)$; /* Train using LSTM base learner h_t from D, V_{pca} using distribution D_t */

$\varepsilon_t = Pr_{i \sim D_i}[h_t(x_i \neq y_i)]$; /* Measure the error of h_t */

$\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$; /* Determine the weight of h_t */

$D_{t+1}(i) = \frac{D_t(i)}{Z_i} \times \begin{cases} exp(-\alpha_t), & \text{if } h_t(x_i) = y_i \\ exp(\alpha_t), & \text{if } h_t(x_i) \neq y_i \end{cases}$

$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$; /* Update the weights distribution, where Z_t is a normalization factor with enables D_{t+1} to be a distribution */

Result: The final strong classifier is:

$$H(x) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) > 0.5 \\ 0, & otherwise \end{cases}$$

4. The Proposed LSTM-BOOST Model

Common baseline architectures give low accuracy scores because the feature information is collected from various sources, and a single model cannot provide good decision boundaries. Hence the ensemble architecture is used. At the learning stage, ensemble classifiers combine the baseline method with different techniques for sampling the features. The boosting ensemble method is one of the most well-known ensemble procedures (Breiman, 1996).

Boosting ensemble architecture is applied to enhance efficiency by training weak learners sequentially. This approach can increase the classification's performance. The most conventional machine learning boosting algorithm is the Ada-Boost (Wang et al., 2011). To enhance the outputs of dimmer classifiers, it is necessary to measure mistakes and update the weights. As the new classifiers are forced to cope with difficult points, some data points may be sacrificed as a result. It has the potential to pollute the accuracy of the classifiers.

We have modified the Ada Boost algorithm. We categorized the data set after Principal Component Analysis (PCA) (Ham and Kwak, 2012). People usually use PCA to minimize the dimension of big data (Gupta et al., 2018) (Kushwaha et al., 2021) sets. The principal component analysis is a common mathematical investigation that is used to find out the highest data dissimilarity in a dataset as a principal component. It is a dimension compression method without supervision. It does not use the label indicated in the dataset. For our study, we used PCA to identify the most variation data portion and skip less variation data sample. Our entire dataset is not trained at once, because we have used individual baseline classifications for ensemble learning. The PCA extracts the core component from the datasets and the AdaBoost model easily generates the weak classifier for each component.

The flow chart of our suggested model is depicted in Figure 3, and the pseudo-code is provided in Algorithm 1. Figure 3 shows that we applied the PCA to determine the most inconsistent part of our dataset before sending the data to logistic regression. Initially LSTM-BOOST

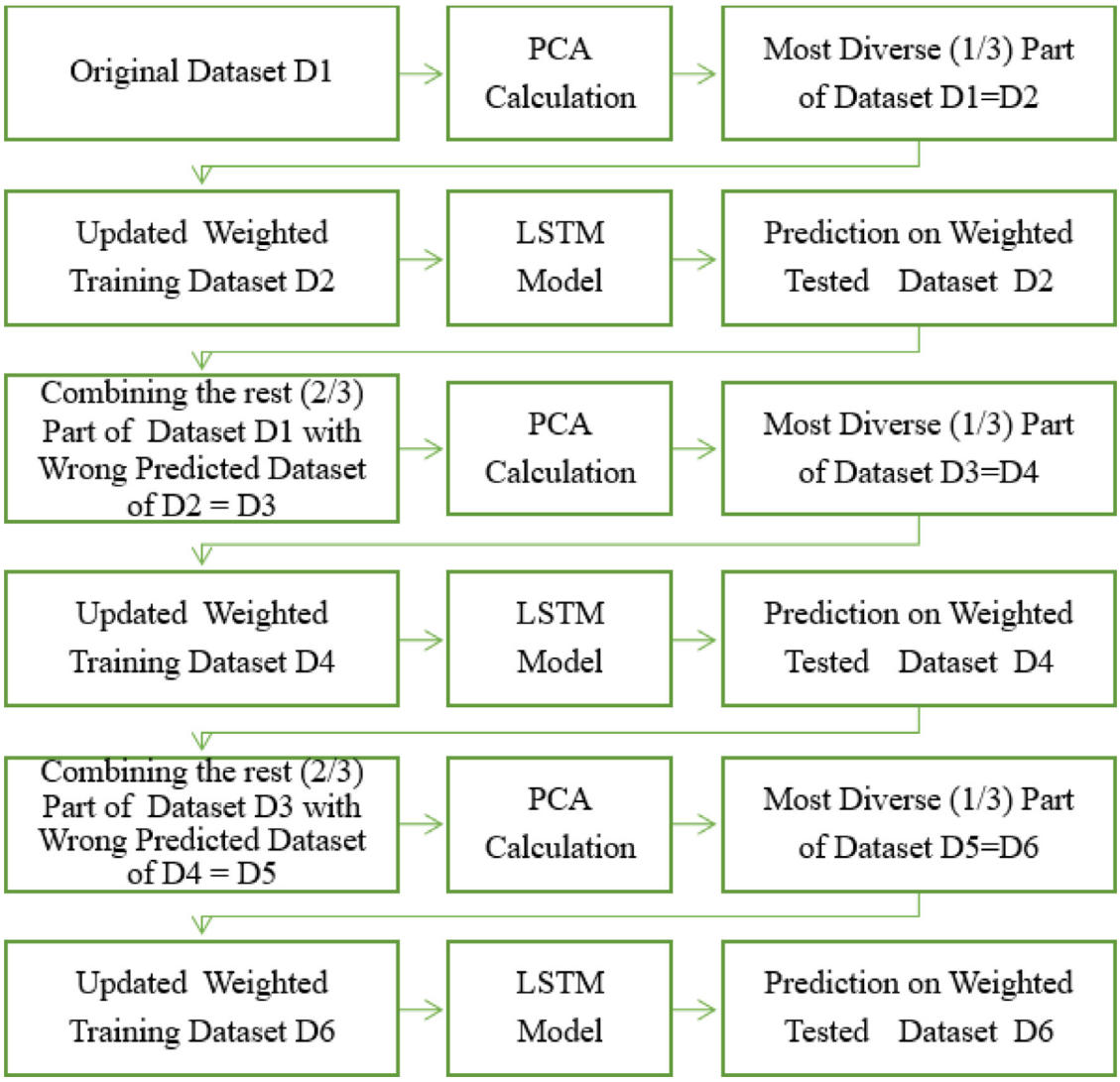


Figure 3. Architecture of Proposed Model: LSTM-BOOST.

model finds out principal component v_{pca} by performing the following operation as like as PCA method. The full dataset has been separated into three pieces to discover the component with the most significant variation among the three portions. At every data piece, the LSTM classifier is trained and measures the weak hypothesis (h_t) and calculates the error(ε_t) of the weak hypothesis. Then update the distribution of the weights to minimize the errors which are used in the next iteration. At first calculate the weighted sample mean as follows:

$$\mu_t = \frac{1}{N}\sum_{i=1}^{N} D_t(i)x_i \tag{2}$$

Then calculate the weighted scatter matrix based on weighted mean is:

$$S_t = \sum_{i=1}^{N} D_t(i)\left(x_i - \mu\right)\left(x_i - \mu\right)^T \tag{3}$$

For n data samples initially there have d-dimensional space and after PCA analysis the d-dimensional space of n data sample reduces to d'-dimensional space without losing any properties. Then the method calculates eigenvectors v_{pca} by applying eigenvalue decomposition to S as follows:

$$V_{pca} = \left[v_1, v_2, \ldots v_{d'}\right] \in \mathbb{R}^{d\times d'} \tag{4}$$

We select the first 1/3 part of the data from most variance datasets and apply Long Short-Term Memory (LSTM) network as base learner to predict the data as follows:

$$h_t = \mathrm{LSTM}\left(D, V_{pca}, D_t\right) \tag{5}$$

Next, we combine the rest 2/3 parts data and the poorly classified data and apply the PCA technique to minimize weighted error by performing weighted mean, threshold and projection vector. Then, we use LSTM techniques to learn best features between strong and weak classifiers. Finally, we use the PCA with LSTM classification to determine decision boundaries for specific classes. We use LSTM classifier to predict data based on all aspects of the decision boundary. After a few iterations, all weak classifiers have been transformed into solid classifiers.

5. Evaluation

This section explains the evaluation procedures of the proposed LSTM- BOOST architecture on our Bengali dataset. We divide the evaluation process into three parts: statistical evaluation, graphical evaluation, and ensemble evaluation. Also, the evaluation metrics, experimental setup are demonstrated in this section.

5.1. Experimental Setup

The evaluation environments are configured using the Ubuntu 18.04 operating system on an Intel(R) Core(TM) i5-6500 processor including

16GB RAM. Python 3.6.9 with TensorFlow 2.2.1 is employed to implement offensive text categorization models. Data frames from Panda 1.0.3 and scikit-learn

0.22.2 are used to construct data sets from scratch for training and testing purposes. While generating the testing and training datasets, we randomly shuffle the whole amount of the text such that the testing and training datasets have a combination of non-offensive and offensive material. We employed 20 percent datasets for testing, and the remaining 80 percent being used for training purposes.

5.2. Evaluation Metric

A classification model's accuracy is measured using a metric applied to the model's estimation. In automated text categorization, accuracy is one of the often-used metrics to assess how well a model categorizes offensive text from a large amount of information. False positives (FP), False negatives (FN), True Positives (TP), and True Negatives (TN) are four measurements in a "confusion-matrix". The relative relevance of the four criteria might vary depending on the variety of accuracy measurements available. In order to evaluate the accuracy of the classification, the proportion of accurate classification overall classifications completed is determined as,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Precision metric is applied to measure the relevancy of the positive classification, estimated as,

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

Recall metric is applied to calculate the positive classifications over the ground positive classifications, measured as,

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

Finally, F1-score is practiced to interpret the discretion of precision and recall of the classified result, which is calculated as follows,

$$F1_{\text{score}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{9}$$

Besides, Area Under the Curve (AUC) and Receiver Characteristic Operator (ROC) curves are used for evaluation. AUC ROC curve is a performance measurement for the classification problems at different thresholds. ROC is a probability curve, and AUC represents the degree of separability. It explains how much the model can differentiate between classes. The ROC curve is plotted with TPR against the FPR, where TPR is on the y-axis and FPR is on the x-axis. The following equations are used to measure TPR and FPR.

$$\text{TRP} = \frac{TP}{TP + FN} \tag{10}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{11}$$

5.3. Experiments and Comparisons

The Python Machine Learning Library has been used to conduct a baseline experiment, which has been presented. We have adopted binary classification procedures employing six baseline classifiers in text classification: stochastic gradient descent (SGD), support vector machine (SVM), K-nearest neighbors (KNN), multinomial naive bayes (MNB), decision tree (DT), and logistic regression (LR). These classification techniques are fit for convex loss functions and linear classifiers.

In addition, all characteristic vectors consist of high-dimensional spaces. Therefore, hyperplanes look for minor steps to separate areas from belonging to distinct categories. When determining the intended hyperplane, numerous hyperplanes are frequently used to optimize the spacing between sub-classes.

Table 8
Features that are employed in the baseline and the offered model.

Feature Name	Feature Value
F1	Uni-gram+Bi-gram
F2	Bi-gram+Tri-gram
F3	Uni-gram+Bi-gram+Tri-gram
F4	Quadri-gram
F5	Tri-gram+Quadri-gram

Besides, the scikit-learn library is utilized for baseline estimation. Various classification methods each have their own set of parameters determined by the volume and format of the input variables, respectively. We em- ployed penalty=ȷl2ȷ, solver= ȷlbfgsʲ, max_iter=100 for LR and SGD classifiers. *max_features = n_features* is utilized for the DT and RF classification methods. However, the *entropy* criterion is employed in DT while the *gini* criterion is used in the RF classifier. For all classifiers, we practiced using learning_rate=*optimal* and random_state=0 settings. All baseline classifiers are evaluated in the next section, which depicts the evaluation results.

5.3.1. Statistical Evaluation

Overall, we looked at most of the baseline classifiers with multiple feature learning approaches that used five different sorts of features in total. Table 8 presents diverse features with their analogous substances. Following a series of practice sessions with different characteristics in Table 7, we listed the top five features that achieved the highest accuracy level. Table 9,10,11 displays the effects of benchmark classification methods, in which each classification model is used independently of the fastText, TF-IDF, Word2Vec, and BoW feature extraction approaches.

Results of the logistic regression algorithm's performance are presented in Table 9, with the highest accuracy achieved with the fastText feature extraction approach and the F3 feature at 87.84% and the lowest accuracy achieved with the Word2Vec feature extraction approach at 69.26% for the F1 attribute with the Word2Vec feature extraction approach. Logistic regression algorithm produces the most relevant result for all features with BoW FE method.

The accuracy of all feature extraction methodologies for the multinomial naive Bayes algorithm exhibits identical for all features, as shown in Table 9 for this algorithm. F1 feature with TF-IDF FE method operates the best score amongst all feature extraction approaches and features which is 83.23%. The highest recall score for the F5 feature with fastText is 95.92%, while the lowest recall score for the F5 feature with BoW FE technique is 58.82%, as shown in Table 11. Table 10 addresses the RF classifier's execution result of diverse features for Word2Vec, TF-IDF, and BoW FE methods. When using Word2Vec feature extraction procedures, RF classifiers get an accuracy score of 91.15% for the F1 feature, which is the greatest accuracy score for F1 attributes among all classification algorithms.

The stochastic gradient descent classifier delivers the highest 91.61% accu- racy for Word2Vec feature extraction approaches with F4 features. Stochastic gradient descent algorithm with TF-IDF and Word2Vec feature extraction approaches perform more significant results than BoW feature, and fastText extraction approach as presented in Table 9. DT and KNN classification algorithms produce scores among 68% to 86%, producing better outcomes for F1 and F4 scores for using feature extraction approaches. Amongst all abusive text classification systems, SVM achieves outstanding results for all of the characteristics from F1 to F5. From Table 9, the minimum accuracy in the SVM classification is 82.35%, and the maximum accuracy using various feature extraction approaches is 90.58%.

5.3.2. Graphical Evaluation

ROC curves or Receiver Operating Characteristic curves are commonly used to illustrate the link between sensitivity and specificity for a

Table 9
Accuracy Results of different baseline classifiers.

	Features	LR Classifier A(%)	MNB Classifier A(%)	RF Classifier A(%)	SGD Classifier A(%)	DT Classifier A(%)	KNN Classifier A(%)	SVM Classifier A(%)
BoW	F1	0.84	0.79	0.87	0.83	0.81	0.84	0.85
	F2	0.83	0.52	0.87	0.82	0.80	0.80	0.84
	F3	0.82	0.79	0.87	0.82	0.79	0.79	0.82
	F4	0.83	0.80	0.89	0.83	0.81	0.80	0.83
	F5	0.83	0.79	0.87	0.83	0.80	0.81	0.83
TF-IDF	F1	0.84	0.73	0.89	0.84	0.80	0.62	0.85
	F2	0.83	0.56	0.89	0.83	0.80	0.82	0.84
	F3	0.84	0.57	0.88	0.84	0.80	0.80	0.84
	F4	0.84	0.72	0.89	0.83	0.79	0.63	0.84
	F5	0.83	0.60	0.88	0.83	0.81	0.84	0.84
Word2Vec	F1	0.69	0.64	0.91	0.83	0.77	0.77	0.83
	F2	0.83	0.63	0.92	0.84	0.78	0.77	0.85
	F3	0.82	0.66	0.92	0.80	0.76	0.78	0.83
	F4	0.83	0.65	0.92	0.83	0.78	0.78	0.84
	F5	0.80	0.64	0.91	0.80	0.76	0.79	0.81
fastText	F1	0.83	0.83	0.85	0.83	0.86	0.84	0.85
	F2	0.81	0.80	0.85	0.81	0.83	0.81	0.86
	F3	0.88	0.78	0.86	0.79	0.83	0.80	0.86
	F4	0.78	0.79	0.87	0.78	0.86	0.79	0.87
	F5	0.82	0.82	0.86	0.82	0.84	0.82	0.87

Table 10
Precision Results of different baseline classifiers.

	Features	LR Classifier P(%)	MNB Classifier P(%)	RF Classifier P(%)	SGD Classifier P(%)	DT Classifier P(%)	KNN Classifier P(%)	SVM Classifier P(%)
BoW	F1	0.80	0.74	0.88	0.79	0.77	0.81	0.81
	F2	0.80	0.94	0.91	0.78	0.76	0.82	0.80
	F3	0.79	0.74	0.90	0.79	0.77	0.80	0.79
	F4	0.79	0.74	0.90	0.78	0.79	0.79	0.79
	F5	0.78	0.79	0.88	0.79	0.76	0.81	0.78
TF-IDF	F1	0.82	0.68	0.90	0.83	0.76	0.60	0.83
	F2	0.81	0.85	0.88	0.82	0.76	0.81	0.82
	F3	0.82	0.89	0.88	0.81	0.76	0.77	0.83
	F4	0.82	0.67	0.89	0.80	0.73	0.60	0.83
	F5	0.81	0.57	0.87	0.80	0.76	0.83	0.82
Word2Vec	F1	0.72	0.79	0.90	0.83	0.76	0.96	0.87
	F2	0.87	0.80	0.92	0.84	0.78	0.95	0.89
	F3	0.86	0.82	0.91	0.77	0.74	0.94	0.87
	F4	0.86	0.80	0.90	0.88	0.79	0.93	0.87
	F5	0.84	0.77	0.90	0.86	0.74	0.94	0.85
fastText	F1	0.83	0.80	0.81	0.83	0.82	0.80	0.80
	F2	0.81	0.95	0.93	0.81	0.78	0.82	0.82
	F3	0.85	0.91	0.84	0.80	0.81	0.80	0.83
	F4	0.78	0.76	0.82	0.78	0.83	0.77	0.82
	F5	0.82	0.76	0.82	0.82	0.80	0.82	0.82

Table 11
Recall Results of different baseline classifiers.

	Features	LR Classifier R(%)	MNB Classifier R(%)	RF Classifier R(%)	SGD Classifier R(%)	DT Classifier R(%)	KNN Classifier R(%)	SVM Classifier R(%)
BoW	F1	0.94	0.96	0.88	0.94	0.93	0.90	0.94
	F2	0.92	0.11	0.84	0.93	0.92	0.81	0.93
	F3	0.92	0.94	0.86	0.92	0.88	0.82	0.92
	F4	0.93	0.94	0.87	0.94	0.89	0.85	0.94
	F5	0.94	0.81	0.87	0.92	0.89	0.83	0.94
TF-IDF	F1	0.89	0.93	0.90	0.88	0.91	0.90	0.89
	F2	0.89	0.22	0.91	0.88	0.92	0.86	0.89
	F3	0.89	0.20	0.90	0.89	0.91	0.89	0.89
	F4	0.90	0.93	0.91	0.91	0.96	0.90	0.88
	F5	0.90	0.98	0.90	0.90	0.93	0.87	0.89
Word2Vec	F1	0.69	0.45	0.94	0.84	0.81	0.60	0.80
	F2	0.80	0.42	0.94	0.86	0.83	0.61	0.82
	F3	0.79	0.46	0.94	0.90	0.84	0.63	0.80
	F4	0.80	0.44	0.94	0.78	0.79	0.62	0.82
	F5	0.78	0.46	0.93	0.74	0.84	0.64	0.78
fastText	F1	0.86	0.90	0.93	0.86	0.95	0.92	0.86
	F2	0.84	0.66	0.77	0.83	0.93	0.83	0.94
	F3	0.95	0.65	0.91	0.82	0.91	0.83	0.94
	F4	0.82	0.89	0.95	0.83	0.92	0.92	0.95
	F5	0.84	0.96	0.94	0.84	0.92	0.83	0.88

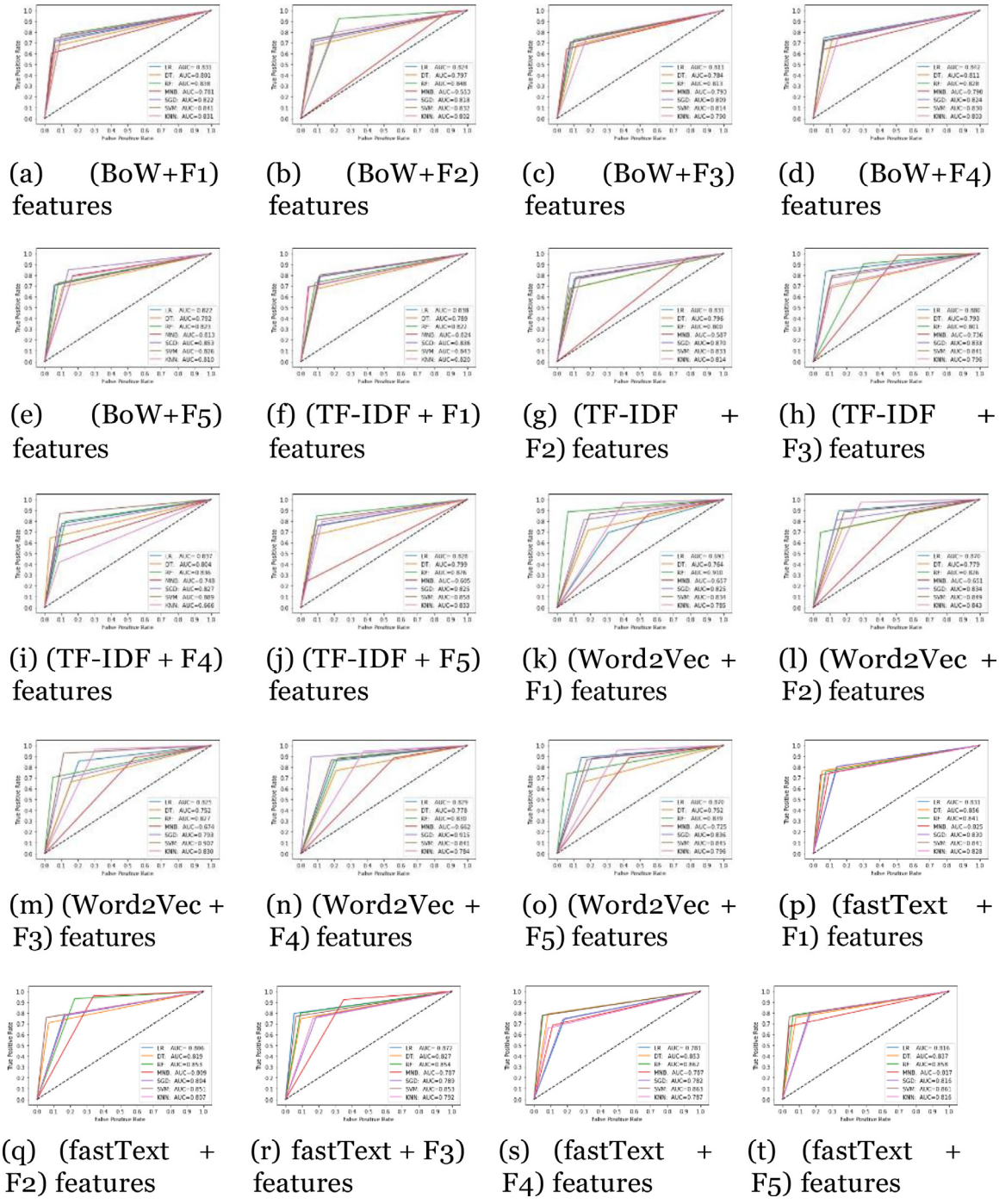


Figure 4. ROC curves for BoW, TF-IDF, Word2Vec, and fastText.

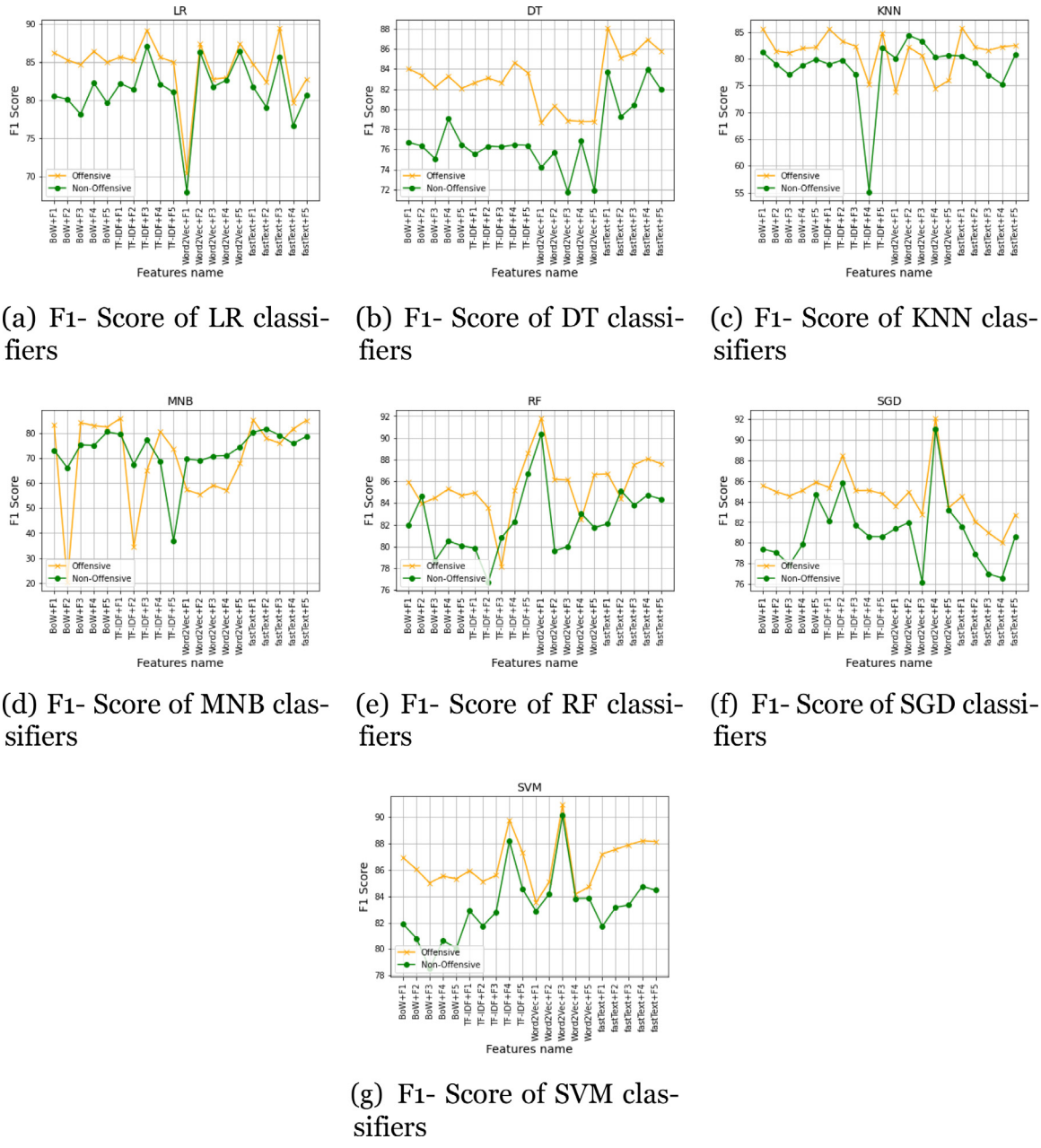
possible cut-off point. Figures 4(a) to (e) presents the ROC curve of BoW feature selection approach. We observe SGD provides the most suitable area beneath the ROC Curve (AUC) 85.3% value with the F5 feature. Another three assaulting classification models, SVM, RF, and LR, using the BoW feature extraction approaches produce reasonable results.

ROC curves for the IF-IDF FE method are presented in Figures 4(f) to (j), the best AUC is 88.9% for SVM with F4 feature, and the minimum value is 58.7% for MNB with F2 feature. The SVM algorithm delivers overall good accuracy with F1 to F5 features.

Thus, Figures 4(k) to (o) exhibit the Word2Vec FE method ROC curve with several features F1 to F5. Practicing the Word2Vec, In the F2 feature, the ROC curve shows that all classifiers perform better AUC than other features. The highest AOC value of 91.5% is produced employing SGD for F4 feature. Using the F1 feature, the RF classifier achieved a 91.0% AUC score. For all F1 through F5 features, the SVM and LR classifiers produce overall satisfactory AUC scores.

Lastly, Figure 4(p)-(t) presents the ROC curve for the fastText FE method. The maximum AUC value of 87.2%generates with F3 features using logistic regression and the minimum value is produced from the F4 feature for logistic regression.

Figures 5(a)-(g) exhibits the F1-scores of all baseline classifiers and Figures 6(a)-(c) presents the F1-scores of three feature extraction approaches. Using the fastText Feature extraction technique, the highest



(a) F1- Score of LR classifiers

(b) F1- Score of DT classifiers

(c) F1- Score of KNN classifiers

(d) F1- Score of MNB classifiers

(e) F1- Score of RF classifiers

(f) F1- Score of SGD classifiers

(g) F1- Score of SVM classifiers

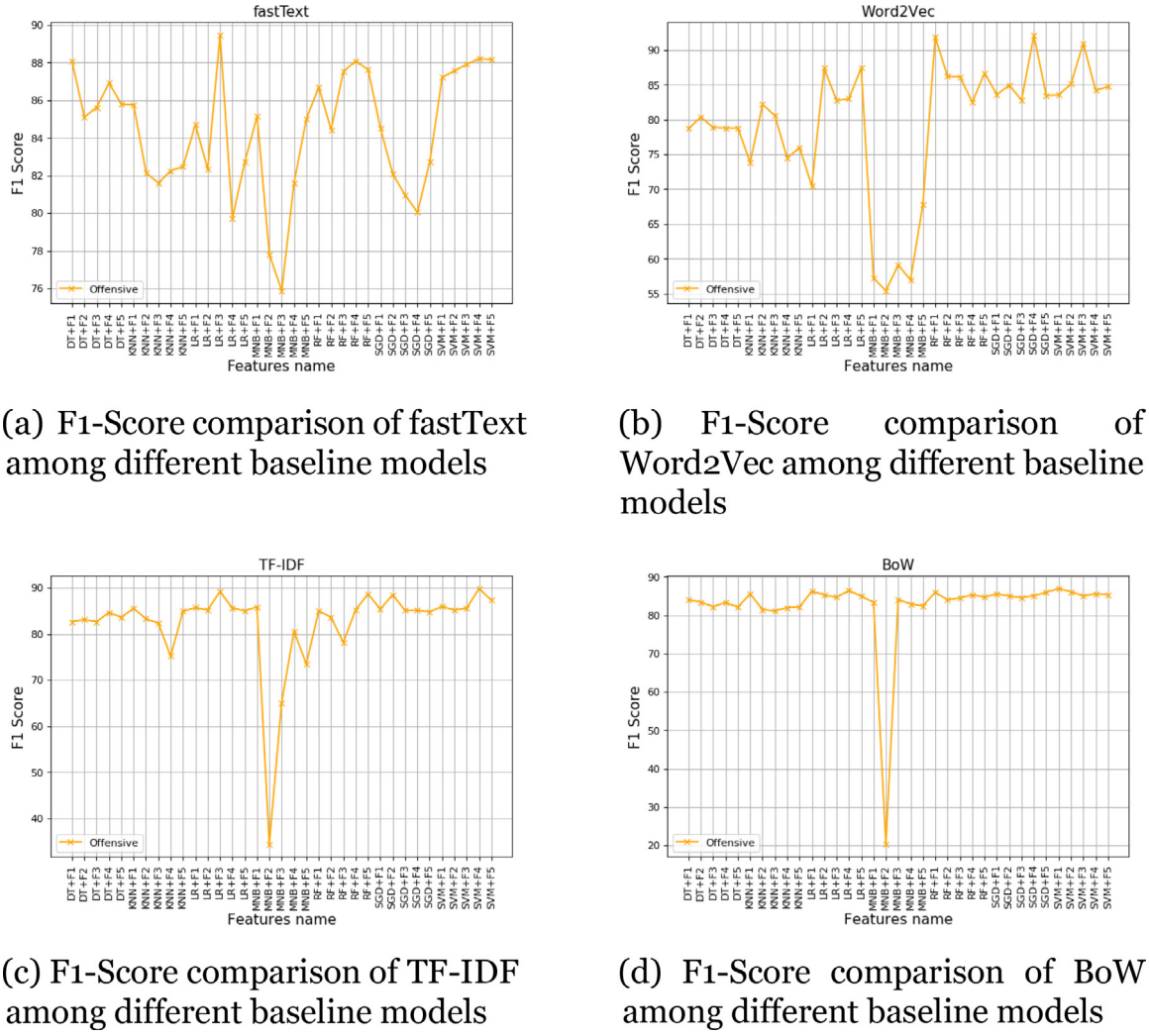
Figure 5. F1-score of various baseline methods.

F1-score is produced for the abusive class from the F3 attribute, but the lowest F1-score is derived from the F1 feature, which is obtained using the Word2Vec approach. The F1-score for k-nearest neighbors is similar to the LR in terms of accuracy.

The decision tree's F1-score of abusive and non-abusive forms is signifi- cantly more complex than other feature extraction methods for all n-gram attributes. Stochastic gradient descent and random forest classifiers deter- mine the F1-score, which is very unregulated. The F1-score is too low for the abusive class in the multinominal naive Bayes classification model. The F1-score of non-abusive text in the support vector machine is very moderate for the F3 feature with BoW feature extraction approach and the maximum in the Word2Vec.

However, for the BoW feature extraction approach, SVM produces the greatest F1-score out of all the classifiers tested for F1 feature. Logistic regression obtained an F1-score of 86.9% for both the F1 and F3 characteristics. The maximum F1-score boundary in BoW strategy is 87.85% and the minimum F1-score boundary is 20.01% for the abusive class. On the contrary, The IF-IDF approach has the highest F1-score of 90.05 percent for the F4 property and the lowest F1-score of 33.35 percent for the F2 feature, indicating superior. Figure 4 shows the F1 scores of many classifiers using the Word2Vec feature extraction technique, with the maximum score of 94.85 percent achieved by RF and SGD and the lowest score of 55.0 percent achieved by MNB classifier using feature F2. In addition, the maximum F1-score for fastText FE method is 87.2% with F3 feature and LR classifier. On the contrary, the minimum score is 78.1% with F4 feature and LR classifier.

However, amongst the four feature extraction strategies, fastText achieves good accuracy for all varieties of features. The Word2Vec FE method delivers good outcomes for some features, produces mediocre accuracy for some features, and reveals that it is not stable in experimenting with abusive text classification. Besides, the BoW FE method needs significant time for massive datasets, and the distinction of the result for some characteristics is too large. TF-IDF provides satisfactory results but fastText outperforms all the strategies. Hence, we have used the



(a) F1-Score comparison of fastText among different baseline models

(b) F1-Score comparison of Word2Vec among different baseline models

(c) F1-Score comparison of TF-IDF among different baseline models

(d) F1-Score comparison of BoW among different baseline models

Figure 6. F1-Score of fastText, Word2Vec, TF-IDF, and BoW with various classifiers.

Table 12
The Proposed Model's Performance Results.

Classifier	ClassName	Features	A(%)	P(%)	R(%)	F1(%)
LSTM-BOOST Model	Offensive	F1	87.33	86.56	90.21	88.35
		F2	90.44	86.81	96.65	91.46
		F3	92.11	91.05	94.22	92.61
		F4	90.58	89.89	92.63	91.24
		F5	89.27	83.13	100.00	90.79

fastText feature extraction approach in our proposed LSTM-BOOST architecture. The F1-Score of fastText, Word2Vec, TF-IDF, and BoW, with various classifiers are presented in Figure 6.

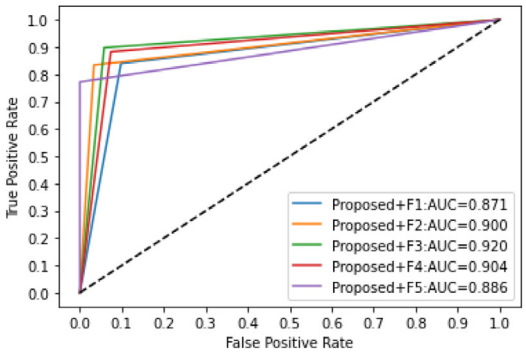
6. Discussion

We have preferred Principal Component Analysis (PCA) and LSTM model to generate the proposed LSTM-BOOST architecture. To estimate our ensemble classification strategy, we used a rule of ten cross-validations, which we followed this method to train all baseline classifiers on a recurrent basis. We used identical datasets to test and train the assessment procedure that we had previously used to evaluate the prior classifiers. After ten repetitions, we measured the outcomes. Table 12 determines the performance result of the proposed model. LSTM-Boost architecture presents the maximum accuracy of 92.11% for F3 features with the fastText feature extraction method where baseline classifiers show the highest 88% for F3 features.

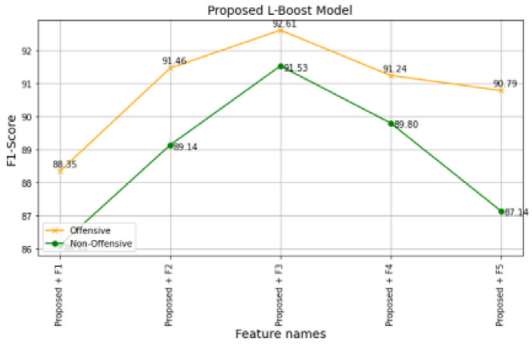
ROC curve investigation of the LSTM-BOOST architecture is presented in Figure 7 for F1, F2, F3, F4, and F5 features. The recommended method delivers the highest AUC value of 92.0% for the F3 feature, whereas the RF classifiers exhibit a 91.0% AUC value which is the largest value between all baseline classifiers. The ROC curve for the F1-featured LSTM-Boost model displays the lowest AOC values of 87.1% between all features. The highest F1-score is received for the F3 feature as displayed in Figure 7 whereas the meanest F1-score is 88.35% from the F1 feature. The suggested architecture has an F1-score of more than 90%, with features from F2 to F5, but the performance of F1-scores as a single F1 feature is relatively lower than other features.

6.1. Contribution to literature

To prove the significance of the proposed architecture, the model is trained and evaluated on the extended Bengali offensive text Dataset (Karim et al., 2021). The LSTM-BOOST architecture was applied on that dataset and found promising results as shown in Table 13. Table 13 reveals that LSTM-BOOST architecture outperforms all deep learning, machine learning and baseline algorithms and DeepHateExplainer architecture by a significant margin. Amongst five machine learning baselines, RF gives a maximum F1-score of 68%. When compared to the other two deep learning baselines, CNN and Bi-LSTM, Conv-LSTM achieves



(a) ROC curve for Proposed Model with different features



(b) F1-Score comparison of Proposed Model with different features

Figure 7. ROC Curve and F1-score comparison of Proposed Model among different features.

Table 13
Performance comparison of LSTM-BOOST on Extended Bengali offensive text dataset with different models.

Model	Precision(%)	Recall(%)	F1-score (%)
RF	69	69	68
KNN	67	67	66
LR	68	68	67
SVM	67	67	66
NB	65	65	64
Conv-LSTM	79	78	78
Bi-LSTM	75	75	75
CNN	74	73	73
DeepHateExplainer	88	88	88
LSTM-BOOST	**89.48**	**93.38**	**91.39**

the greatest F1-score of 0.78 percent. Table 13 also presents that LSTM-BOOST outperforms the DeepHateExplainer by a good margin.

6.2. Practical implication

This paper contributes an empirical study on ensembled Long-Short-Term Memory and Adaptive Boosting for Bengali abusive text classification on social media platforms. Our proposed LSTM-Boost model is helpful for monitoring various posts, comments, and messages on different social platforms. If anyone wants to integrate artificial intelligence support to detect offensive text in their system, they can use this model.

7. Conclusion

In this research work, the modified Adaboost algorithm used the LSTM network as a weak learner to boost the performance of the classifiers by some iteration, and PCA is used to identify the most variation data portion and skip less variation data sample. We compared the efficiency of the proposed method using different feature extraction approaches, including BoW, TF-IDF, Word2Vec, and fastText. The study analyzed the result of several baseline architectures, including Linear Regression, Decision Tree, K-Nearest Neighbor, Multinomial Naive Bayes, Random Forest, Stochastic Gradient Descent, Support Vector Machine separately. Then, the modified ensembled method with LSTM was applied to mentioned baseline architectures and found a more efficient abusive text recognition method. As the Bengali peoples talk differently based on the place, this system might give comparatively poor accuracy on such cases. Also, this model cannot handle an infinitely long sequence text which is our future research plan. In the future, we will try to engage a dataset and propose a novel model that can help to solve this problem. Also, we will try to detect abusive content from images, pdf, videos, audios, etc. Another future plan is that offensive text can identify from the semi-supervised model. The machine can predict with a combination of the supervised and unsupervised classification model in a semi-supervised model. Furthermore, research work can be scaled up using unsupervised models. And using unsupervised clustering models, a large dataset can be annotated automatically. Then the annotated dataset can be fed into a supervised model for better accuracy.

Declaration of interests

None

Acknowledgements

We would like to acknowledge the support of the Bangladesh University of Business & Technology and AML lab for their suggestion and resource sharing.

References

Afroz, U. S., Khan, R. H., Khushbu, S. A., & Masum, A. K. M. (2021). Refinement of bengali obscene words using sequence to sequence rnns. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1–4). IEEE. pages.

Aswani, R., Kar, A. K., & Ilavarasan, P. V. (2019). Experience: managing misinformation in social media—insights for policymakers from Twitter analytics. *Journal of Data and Information Quality (JDIQ), 12*(1), 1–18.

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

Basile, V., Bosco, C., Fersini, E., Debora, N., Patti, V., Pardo, F. M. R., ... & Sanguinetti, M. (2019). Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.

Batra, J., Jain, R., Tikkiwal, V. A., & Chakraborty, A. (2021). A comprehensive study of spam detection in e-mails using bio-inspired optimization techniques. *International Journal of Information Management Data Insights, 1*(1), Article 100006.

Beautifulsoup4. *Available online: https://pypi.org/project/beautiful-soup4/*, (accessed on 10 January 2022).

Bnltk (bangla natural language processing toolkit). *Available online: https://pypi.org/project/bnltk/*, (accessed on 10 January 2022).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146.

Bourgonje, P., Moreno-Schneider, J., Srivastava, A., & Rehm, G. (2017). Automatic classification of abusive language and personal attacks in various forms of online communication. In *International Conference of the German Society for Computational Linguistics and Language Technology* (pp. 180–191). Cham: Springer. pages.

Breiman, L. (1996). Bagging predictors. *Machine learning, 24*(2), 123–140.

Burnap, P., & Williams, M. L. (2015). Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & internet, 7*(2), 223–242.

Chen, K., Zhang, Z., Long, J., & Zhang, H. (2016). Turning from tf-idf to tf-igm for term weighting in text classification. *Expert Systems with Applications, 66*, 245–260.

Chen, T., Xu, R., He, Y., & Wang, X. (2017). Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications, 72*, 221–230.

Chen, J., Yan, S., & Wong, K. C. (2020). Verbal aggression detection on twitter comments: Convolutional neural network for short- text sentiment analysis. *Neural Computing and Applications, 32*(15), 10809–10818.

Chen, Y. (2015). Convolutional neural networks for sentence classification (Master's thesis, University of Waterloo).

Colladon, A. F., & Gloor, P. A. (2019). Measuring the impact of spammers on e-mail and Twitter networks. *International Journal of Information Management, 48*, 254–262.

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media, 11*(1), 512–515.

De Gibert, O., Perez, N., García-Pablos, A., & Cuadros, M. (2018). Hate speech dataset from a white supremacy forum. *arXiv preprint arXiv:1809.04444*.

Del Vigna12, F., Cimino23, A., Dell'Orletta, F., Petrocchi, M., & Tesconi, M. (2017). Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)* (pp. 86–95). pages.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language under- standing. *arXiv preprint arXiv:1810.04805*.

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., & Bhamidipati, N. (2015). Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web* (pp. 29–30). pages.

Founta, A. M., Djouvas, C., Chatzakou, D., Leontiadis, I., Blackburn, J., Stringhini, G., . . . &, & Kourtellis, N. (2018). Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings of the International AAAI Conference on Web and Social Media: 12*.

Founta, A. M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A., & Leontiadis, I. (2019). A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM conference on web science* (pp. 105–114). pages.

Gambäck, B., & Sikdar, U. K. (2017). Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online* (pp. 85–90). pages.

Garg, R., Kiwelekar, A. W., Netak, L. D., & Ghodake, A. (2021). i-Pulse: An NLP based novel approach for employee engagement in logistics organization. *International Journal of Information Management Data Insights, 1*(1).

Genkin, A., Lewis, D. D., & Madigan, D. (2007). Large-scale bayesian logistic regression for text categorization. *technometrics, 49*(3), 291–304.

Gkikas, D. C., Tzafilkou, K., Theodoridis, P. K., Garmpis, A., & Gkikas, M. C. (2022). How do text characteristics impact user engagement in social media posts: Modeling content readability, length, and hashtags number in Facebook. *International Journal of Information Management Data Insights, 2*(1).

Gupta, S., Kar, A. K., Baabdullah, A., & Al-Khowaiter, W. A (2018). Big data with cognitive computing: A review for the future. *International Journal of Information Management, 42*, 78–89.

Ham, S. L., & Kwak, N. (2012). Boosted-pca for binary classification problems. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1219–1222). IEEE. pages.

Hua, T., Chen, F., Zhao, L., Lu, C. T., & Ramakrishnan, N. (2013). Sted: semi-supervised targeted-interest event detectionin in twitter. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1466–1469). pages.

Jiang, M., Liang, Y., Feng, X., Fan, X., Pei, Z., Xue, Y., & Guan, R. (2018). Text classification based on deep belief network and softmax regression. *Neural Computing and Applications, 29*(1), 61–70.

Jie, G., & Li-chao, C. (2010). Research of improved if-idf weighting algo- rithm. In *The 2nd International Conference on Information Science and Engineering* (pp. 2304–2307). IEEE. pages.

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2020). Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI conference on artificial intelligence, 34*, 8018–8025 pages.

Karim, M. R., Dey, S. K., Islam, T., Sarker, S., Menon, M. H., Hossain, K., ... & Decker, S. (2021 , October). Deep- hateexplainer: Explainable hate speech detection in under-resourced bengali language. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA) (pp.* 1-10).IEEE.

Karim, M. A. (Ed.). (2013). *Technical challenges and design issues in bangla language processing*. IGI Global.

Kim, S. B., Han, K. S., Rim, H. C., & Myaeng, S. H. (2006). Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering, 18*(11), 1457–1466.

Kowsari, K., Heidarysafa, M., Brown, D. E., Meimandi, K. J., & Barnes, L. E. (2018). Rmdl: Random multimodel deep learn- ing for classification. In *Proceedings of the 2nd International Conference on Information System and Data Mining* (pp. 19–28). pages.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information, 10*(4), 150.

Kulkarni, A., & Shivananda, A. (2021). Converting text to features. In *Natural Language Processing Recipes*, pages 63–106, Apress, Berkeley, CA.

Kumar, R., Ojha, A. K., Malmasi, S., & Zampieri, M. (2018). Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC- 2018)* (pp. 1–11). pages.

Kumar, A., Abirami, S., Trueman, T. E., & Cambria, E. (2021). Comment toxicity detection via a multichannel convolutional bidirec- tional gated recurrent unit. *Neurocomputing, 441*, 272–278.

Kumar, S., Kar, A. K., & Ilavarasan, P. V. (2021). Applications of text mining in services management: A systematic literature review. *International Journal of Information Management Data Insights, 1*(1), Article 100008.

Kushwaha, A. K., Kar, A. K., & Dwivedi, Y. K. (2021). Applications of big data in emerging management disciplines: A literature review using text mining. *International Journal of Information Management Data Insights, 1*(2), Article 100017.

Kwok, I., & Wang, Y. (2013). Locate the hate: Detecting tweets against blacks. In *Proceedings of the AAAI Conference on Artificial Intelligence: 27* (p. 2013).

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence: 29*.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research, 2*(Feb), 419–444.

Mandal, A. K., & Sen, R. (2014). Supervised learning methods for bangla web document categorization. *arXiv preprint arXiv:*1410.2045.

Maqsood, H., Mehmood, I., Maqsood, M., Yasir, M., Afzal, S., Aadil, F., . . . &, & Muhammad, K. (2020). A local and global event sentiment based efficient stock exchange forecasting using deep learning. *International Journal of Information Management, 50*.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep learning–based text classi- fication: A comprehensive review. *ACM Computing Surveys (CSUR), 54*(3), 1–40.

Mridha, M. F., Wadud, M. A. H., Hamid, M. A., Monowar, M. M., Abdullah-Al-Wadud, M., & Alamri, A. (2021). In *L-Boost: Identifying Offensive Texts From Social Media Post in Bengali: 9* (pp. 164681–164699). Ieee Access.

Naredla, N. R., & Adedoyin, F. F. (2022). Detection of hyperpartisan news articles using natural language processing technique. *International Journal of Information Management Data Insights, 2*(1).

Nasir, J. A., Khan, O. S., & Varlamis, I. (2021). Fake news detection: A hybrid CNN-RNN based deep learning approach. *International Journal of Information Management Data Insights, 1*(1).

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., & Chang, Y. (2016). Abusive language detection in online user content. In *Pro- ceedings of the 25th international conference on world wide web* (pp. 145–153). pages.

Palivela, H. (2021). Optimization of paraphrase generation and identification using language models in natural language processing. *International Journal of Information Management Data Insights, 1*(2).

Pandarachalil, R., Sendhilkumar, S., & Mahalakshmi, G. S. (2015). Twitter sentiment analysis for large-scale data: an unsupervised approach. *Cognitive computation, 7*(2), 254–262.

Pitsilis, G. K., Ramampiaro, H., & Langseth, H. (2018). Effective hate-speech detection in twitter data using recurrent neural networks. *Applied Intelligence, 48*(12), 4730–4742.

Pradhan, R., Chaturvedi, A., Tripathi, A., & Sharma, D. K. (2020). A review on offensive language detection. In *Advances in Data and Information Sciences*, pages 433–439. Springer.

Razavi, A. H., Inkpen, D., Uritsky, S., & Matwin, S. (2010). Of- fensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence* (pp. 16–27). Springer. pages.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Saleem, H. M., Dillon, K. P., Benesch, S., & Ruths, D. (2017). A web of hate: Tackling hateful speech in online social spaces. *arXiv preprint arXiv:*1709.10159.

Santos, C. N. D., Melnyk, I., & Padhi, I. (2018). Fighting offensive language on social media with unsupervised text style transfer. *arXiv preprint arXiv:*1805.07685.

Schmidt, A., & Wiegand, M. (2019). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media* (pp. 1–10). pages.

Sharma, S., Rana, V., & Kumar, V. (2021). Deep learning based semantic personalized recommendation system. *International Journal of Information Management Data Insights, 1*(2).

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune bert for text classification? *China National Conference on Chinese Computational Linguistics*, 194–206 pages.

Python translate. *Available online: https://pypi.org/project/translate/*, (accessed on 10 January 2022).

ul Haque, R., Mehera, P., Mridha, M. F., & Hamid, M. A. (2019). A complete bengali stop word detection mechanism. In *2019 Joint 8th international conference on informatics, electronics & vision (ICIEV) and 2019 3rd international conference on imaging, vision & pattern recognition (icIVPR)* (pp. 103–107). IEEE. pages.

Wadud, M. A. H., & Rakib, M. R. H. (2021). Text coherence analysis based on misspelling oblivious word embeddings and deep neural network. *International Journal of Advanced Computer Science and Applications, 12*(1).

Wadud, M. A. H., Mridha, M. F., & Rahman, M. M. (2022). Word Embedding Methods for Word Representation in Deep Learning for Natural Language Processing. *Iraqi Journal of Science*, 1349–1361. 10.24996/ijs.2022.63.3.37.

Wadud, M. A. H., Mridha, M. F., Shin, J., Nur, K., & Saha, A. K. (2022). Deep-BERT: Transfer Learning for Classifying Multilingual Offensive Texts on Social Media. *Computer Systems Science and Engineering, 44*(2), 1775–1791. 10.32604/csse.2023.027841.

Wang, G., Hao, J., Ma, J., & Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications, 38*(1), 223–230.

Warner, W., & Hirschberg, J. (2012). Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media* (pp. 19–26). pages.

Waseem, Z., & Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop* (pp. 88–93). pages.

Waseem, Z. (2016). Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science* (pp. 138–142). pages.

Wiedemann, G., Ruppert, E., Jindal, R., & Biemann, C. (2018). Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. *arXiv preprint arXiv:*1811.02906.

Xiang, G., Fan, B., Wang, L., Hong, J., & Rose, C. (2012). Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (pp. 1980–1984). pages.

Yenala, H., Jhanwar, A., Chinnakotla, M. K., & Goyal, J. (2018). Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics, 6*(4), 273–286.

Yin, D., Xue, Z., Hong, L., Davison, B. D., Kontostathis, A., & Edwards, L. (2009). Detection of harassment on web 2.0. *Proceedings of the Content Analysis in the WEB, 2*, 1–7.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:*1902.09666.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*, 2019.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolu- tional networks for text classification. *arXiv preprint arXiv:*1509.01626.