L-Boost: Identifying Offensive Texts from Social Media Post in Bengali

M. F. MRIDHA 1 (Senior Member, IEEE), MD. ANWAR HUSSEIN WADUD1, MD. ABDUL HAMID 2, MUHAMMAD MOSTAFA MONOWAR 2(Member, IEEE), M. ABDULLAH-AL-WADUD3, AND ATIF ALAMRI3

1Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh
2Department of Information Technology, Faculty of Computing and Information Technology, King AbdulAziz University, Jeddah-21589, Saudi Arabia
3Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

Corresponding author: M. F. Mridha (e-mail: firoz@bubt.edu.bd).

The authors are grateful to the Deanship of Scientific Research at King Saud University for funding this work through Vice Deanship of Scientific Research Chairs: Chair of Pervasive and Mobile Computing.

ABSTRACT Due to the significant increase in Internet activity since the COVID-19 epidemic, many informal, unstructured, offensive, and even misspelled textual content has been used for online communication through various social media. The Bengali and Banglish (Bengali words written in English format) offensive texts have recently been widely used to harass and criticize people on various social media. Our deep excavation reveals that limited work has been done to identify offensive Bengali texts. In this study, we have engineered a detection mechanism using natural language processing to identify Bengali and Banglish offensive messages in social media that could abuse other people. First, different classifiers have been employed to classify the offensive text as baseline classifiers from real-life datasets. Then, we applied boosting algorithms based on baseline classifiers. AdaBoost is the most effective ensemble method called adaptive boosting, which enhances the outcomes of the classifiers. The long short-term memory (LSTM) model is used to eliminate long-term dependency problems when classifying text, but overfitting problems occur. AdaBoost has strong forecasting ability and overfitting problem does not occur easily. By considering these two powerful and diverse models, we propose L-Boost, the modified AdaBoost algorithm using bidirectional encoder representations from transformers (BERT) with LSTM models. We tested the L-Boost model on three separate datasets, including the BERT pre-trained word-embedding vector model. We find our proposed L-Boost’s efficacy better than all the baseline classification algorithms reaching an accuracy of 95.11%.

INDEX TERMS Offensive text, social media harassment, natural language processing, ensemble learning, BERT model.

I. INTRODUCTION

DURING the COVID-19 global pandemic, people have been protecting themselves by maintaining as much social distance as possible and communicating virtually with each other. For this purpose, they have been engaging in various social media such as Facebook, TikTok, FaceTime, WhatsApp, Zoom, etc., and have been in constant touch with others. As a result, online content has grown rapidly and people are constantly harassed through online media. The L1ght [1] organization has been monitoring cyberbullying content, such as online harassment or hate speech, since 2018. They found that hate speech on Twitter increased by 900% and traffic on several hate sites increased by 200% during COVID-19.

About 245 million people in the world use Bengali as a language and it is the 7th most spoken language among all languages. Bengali language processing research is not as rich as other languages, such as English, Arabic, or European languages. Many researchers have attempted to automate the Bengali language, but this is still in the development stage. Most Bengali people are...
connected to virtual social networks [2], where people are being personally attacked using offensive and inaccurate texts [3] that may threaten their lives, cause social media harassment, religious riots, eve-teasing and above all incites the people against the government.

Texts that are intentionally used to harass people are known as offensive [4] texts. Offensive content encourages people to engage in immoral activities against the government and law enforcement agencies, hurt religious feelings and encourage people to commit crimes without legitimate reasons [5]. Offensive texts can be spread on social media through different forms including text, images, audio, video, or graphical formats. However, the text format is the most commonly used form of social media. In addition, a text using hashtags can be easily converted into clickable links and the same hashtag can easily link text to another group. Thus, the text spreads faster than any other format and helps in the convenient retrieval of any information. In this study, we focus on the textual format of the offensive content and classify whether any text is offensive or not.

Undoubtedly, the manual detection of offensive text from a large number of Internet sources is difficult [6]. Thus, the enormous growth of web data obviates the need to develop an automated offensive text detection system. Recently, Bengali offensive texts have been extensively used to harass people on various platforms. Automated Bengali offensive text detection is thus required to prevent harassment on social media [4]. Such a system may also display a warning message to the user before reading the content. Thus, a user can verify the content of a website or other social media before reading or sharing the content written by others and eliminate online irritation. However, developing an automated offensive text classification model for the Bengali language is a challenging task. This is due to some unique characteristics of the Bengali language in word formation as well as different sentence structures compared to other languages. In Bengali, the same verb can give different meanings due to changes in subject, person, gender, tense, and a large number of synonyms and word senses. For example, in Table 1, the first Bengali sentence has the same meaning as the English sentence. However, in the second example, the literal translation of the Bengali sentence to English does not convey the same meaning. This is because the second example is an offensive text. The verb “eat” here has offensive meaning in Bengali while in the first example it is used as its normal meaning.

Bengali people write Bengali sentences using English letters known as the Banglish form. They can write a full paragraph in the Banglish form rather than writing a single sentence using Bengali characters. This is because writing a single Bengali word in pure Bengali format is more difficult than writing Bengali words in Banglish format, and they use different types of emoji while writing their comments. Table 2 shows an example of a Banglish offensive sentence in which a user uses various emojis and emotions to directly insult someone.

Processing Bengali or Banglish text is not similar to any other language. Therefore, existing offensive text detection methods in other languages cannot accurately detect the offensive Bengali text. However, recently [7] some text classification algorithms have been used to identify offensive Bengali texts. But, these algorithms do not eliminate long-term dependency problems when classifying text. Salim Sazzad [8] suggested abusive detection methods on a transliterated Bengali-English social media dataset. His dataset contains several English and Banglish words but not a single Bengali word. In their feature extraction step, they did not convert the text to the corresponding Bengali or English format but instead used the TF-IDF method to find the vectors of all the words and applied different classification algorithms to find the accuracy.

Existing offensive detection methods in Bengali are mainly limited to baseline experiments, and the other offensive text detection methods in different languages when applied in Bengali show poor accuracy results due to the unique syntactic and semantic structure of the Bengali language [7], [9]. The baseline classification methods have no weight gain scores that are incorrectly classified by the previous iterations. In addition, the baseline methods cannot fit all datasets into an optimal class [10]. However, ensemble methods can exhibit better accuracy with a smaller dataset [11]. Recently, an article [12] was published in Bengali based on ensemble prediction to identify hate speech. In this study, the authors used a variety of BERT approaches such as BanglaBERT based [13], Bangla-BERT-uncased, XLM-RoBERTa etc. in the ensemble method to increase their accuracy. They show that using the ensemble method, their accuracy boosts up to 7%. The most popular ensemble method for binary classification is the AdaBoost ensemble method [14]. The AdaBoost [15] ensemble method is used to improve the accuracy by training weak learners sequentially and increasing the

| Table 1. Example of Bengali offensive text |
|-------------------------------------------|
| Bengali | English | Actual Meaning | Type |
| আমি ভাঙ্গ রাইছি | I ate rice | I ate rice | Non-offensive |
| আমি তোরে রাইছি | I ate you | I will kill you | Offensive |

| Table 2. Example of Banglish offensive text |
|-------------------------------------------|
| Banglish | English |
| Ovinetri hole hole hucca 😂😡 | If you want to be an actress, you have to go to sexual relations with naughty producers |
results of text classification. This algorithm selects a weak classifier whose weighted error value is the lowest compared to other weak classifiers. Then, each sample is trained by selecting a new weight and a weak classifier by repeating T times. Finally, all the weak classifiers are combined to form strong classifiers. Sometimes, the AdaBoost algorithm selects a very poor classifier that raises low margin and overfitting issues [16]. Viola and Jones [16] used Haar-like features as weak classifiers in AdaBoost algorithm for object detection.

To address the challenges mentioned above, we propose the L-Boost ensemble method for Bengali offensive text identification. Our proposed model consists of three parts: the first part is proposed to predict offensive text using BERT [17] embedding with a long short-term memory (LSTM) [18] model. BERT is an unsupervised learning model based on bidirectional deep learning that is used for a variety of purposes for text processing [17]. It can be used as a feature-based approach or pretrained word embedding model, or fine tuning in text classification. The LSTM model is used to retain data over a long period of time due to its repetitive formation and gating procedure. In addition, the LSTM model is considered a sophisticated method for time series-related problems. Conventional classification models such as SVM and general neural networks make it difficult to extract long-term dependence on the offensive text where LSTM neural networks are better at it. The second part of our proposed model is AdaBoost with a custom BERT fine-tuning model. However, the LSTM model performs significant results in predicting offensive texts. But for its complex architecture, it may occur some problems such as overfitting, which is the loss of squared prediction error, the relationship between the model complexity and the decomposing material, and the relationship between the decomposing material and model fitting. BERT performs significantly better than other existing NLP models, but also has some limitations. In the BERT model, the first token must be [CLS], which is used for finding specific embedding, and the model uses another token [SEP] for segment separation. These tokens are used during the fine-tuning process, but tokens are unavailable during the prediction phase. On the other hand, AdaBoost has strong forecasting ability with low bias error, and overfitting problems do not occur easily. Considering the above advantages and disadvantages of these two models, the last part of our model is used to combine the first and second parts of the proposed model and the average of the result for the final output. Because both the LSTM and AdaBoost models have low bias errors, the proposed L-Boost model also has low bias errors. In this study, a number of supervised text classification algorithms were first applied to predict offensive Bengali text. Then, we employ the L-Boost ensemble architecture for detecting Bengali offensive text based on text classification algorithms. For feature extraction, we used fastText [19], BERT [20], TF-IDF [21] and Word2Vec [22]. Machine classification algorithms such as support vector machine (SVM) [23], decision tree (DT) [24], random forest (RF) [25], and LSTM [18] are implemented; and compared with the results. These text classification methods are known as baseline methods. The highest outcome is chosen as the benchmark. The BERT baseline methods were used in the proposed L-Boost model to boost the classification [26]. Finally, the L-Boost model was evaluated on a comprehensive Bengali Hate Speech dataset [12], and on the transliterated Bengali-English dataset [8]. Then shows the comparison between L-Boost and existing models. This research focuses on creating appropriate offensive text identification methods for offensive Bengali text on social media. The main contributions of this study are summarized as follows:

- We create a real-life dataset in both the Bengali and Banglish formats from different social media.
- We build a set of benchmark experiments based on machine learning and deep learning to detect the Banglish offensive text.
- We design the LSTM deep learning model with BERT embedding for text prediction.
- We propose the AdaBoost-BERT ensemble model by modifying AdaBoost with a BERT fine-tuning transformer for text prediction.
- We propose our L-Boost model by combining the modified LSTM model and the AdaBoost-BERT model.

The remainder of this paper is organized as follows. Section II presents the related works. The methodologies are described in section III. The results of the analyses are presented in section IV. Finally, section V concludes the work.

II. RELATED WORKS

Offensive text identification is a relatively new field in the field of natural language processing (NLP). However, most of the works in offensive text detection are in English, and very few works exist in Bengali language processing. In this section, we first discuss some of the existing methods for identifying offensive texts in languages other than Bangali and then, discuss existing works on the Bangali language.

Cavnar et al. [27] proposed an N-gram based method for text categorization in which textual errors were tolerated. In this method, a large range of text categorization is possible, as the N-gram has a trustworthy background for frequency. However, this method is suitable only for ASCII code. Yarowsky [28] implemented an unsupervised method to sense the proper meaning of a sentence. Manual preparation is required to train unprepared text.
using a supervised technique. This makes the supervised technique very slow. McCallum and Nigam [29] aimed to show the differences between the multivariate Bernoulli model and the multinomial model. The authors implemented these methods on a corpus of five text files and compared their performances. Hu et al. [30] assigned distinctive weights to the same word and considered the impact of these weights in the Bayes network. The authors considered these weights for the texts at the time of appearing in distinctive web page components such as the body, title, etc. Yin et al. [31] proposed an n-gram-based supervised classification technique to identify abusive text.

Chen et al. [32] identified the abusive words. They have tried to improve machine learning methods using context-specific features, structural features, and style features, as well as lexical features, to detect offensive language. Their process can tolerate the informal intent of any English writing style. Sood et al. [33] analyzed social media comments. They scrutinized and noticed that the effect of the current method was negligible. After identifying social differences based on the tolerance of pornography, the authors proposed a more advanced method of detecting contempt. Yang et al. [34] implemented a new method. Embedding can be properly used with the aim of mining new rules. Jurka et al. [35] offered comprehensive text categorization by interfacing with preprocessed words and machine learning algorithms with a new analytical function. Marmol et al. [36] proposed a method using SNSs to limit users from accusing reliable content. Kim [37] investigated sentence classification using convolutional neural networks (CNNs). Yadav and Manwatkar [38] proposed an approach of text filtration and classification using the Aho-Corasick string pattern matching algorithm using their dictionary to match keywords. Profane words were prevented from being published, and semantically related words were ignored. Shashank et al. [39] proposed a text filtration module and created a database of abusive keywords. In their research, they wanted to explain various text categorization methods. Nobata et al. [40] proposed a machine learning method to determine abusive speech from the comments of Internet users. Recent deep-learning-based offensive text detection methods have benefited from their machine-learning approach. Chu et al. [41] identify and categorize abusive comments using CNN and LSTM deep learning methods and a neural language processing technique. They tested three models: first, a long-term short-term memory cell (LSTM) and a recurrent neural network (RNN) with word embedding; a convolutional neural network (CNN) with word embedding; and, a CNN with character embedding. Wulczyn et al. [42] discussed the effectiveness of logistic regression and multilayer perceptron for abuse classification. They also compared their methods with a human baseline.

Bengali language processing is under development, and no proper dataset is available to experiment with offensive identification methods. Because of these, very few studies have been conducted on offensive text identification. Ishmam et al. [43] proposed the gated recurrent neural network (GRNN) method to detect hateful Bengali language and achieved 70.10% accuracy on 5K sized datasets including six classes. They used approximately 900 documents per class to train the model and to create accurate accuracy. The main reason for the poor accuracy is that they did not define classes correctly before training in the hate text classification. Eshan and Hasan [44] analyzed different machine classifiers, such as support vector machine (SVM) with linear kernel, multinomial naive Bayes (MNB), random forest, polynomial and sigmoid kernel of the radial basis function (RBF). Each classifier is compared and learned with the TF-IDF and CountVectorizer based on unigram, bigram, and trigram features. The support vector machine with a linear kernel outperforms and shows 85% accuracy with the TF-IDF vectorizer based on trigram features. This is due to the limited dataset and the baseline classification methods have no weight gain scores that were incorrectly classified by previous iterations and could not fit all datasets into the optimal class. Hussain et al. [45] extracted comments from different social media platforms and proposed an algorithm for the detection of Bengali abusive texts, where string unigram features show better performance. To identify an offensive text, Granizo et al. [46] used computer vision and natural language processing on Twitter and linked websites. They categorize illegal massage based on a number of factors, such as human trafficking, disappearances, and abductions. In [12], authors proposed the DeepHateExplainer model based on several BERT variants. They used Bangla-BERT, Multilingual BERT, and XML-RoBERTa to construct their models. They worked on different categories of Bengali hate text, which are all written in Bengali or Bengali with English words but not Banglish format. The authors in [47] used BERT and ELECTRA for Bengali text classification. They discussed the systematic approach of the BERT model for sentiment analysis in the Bengali document. Recently, several NLP researchers used the BERT transformer model for text classification [48–51]. A few researchers [48, 49, 51] used fine-tuning of BERT transformers, whereas reference [50] exploits pre-trained and fine-tuning during sequential text classification and showed better accuracy than other models. In [52], the authors used MNB, SVN, and CNN-LSTM with different kernels to identify threat and abusive text in the Bengali language. Their proposed system shows low accuracy because of the limited dataset, and pre-trained word embeddings are unavailable. They show
that SVM with a linear kernel performs better than the other two models. On the other hand, the authors in [53] proposed another Bengali hate speech detection model using an encoder-decoder based LSTM network. They used a 1D CNN model in an LSTM network and TF-IDF and word2Vec to solve the multiclass hate speech problem. In [54], the author proposed offensive text identification in different languages. They used a cross-lingual transformer model (XLM-R) to detect offensive text, including small datasets. However, Banglish text formats are not available in all existing research works.

From the above discussion, we can conclude that some studies have attempted to identify Bengali offensive text, including small datasets. However, limited research work has used the Banglish datasets to identify offensive texts. Furthermore, the baseline experiments with limited datasets were used by most researchers. The main limitation of baseline experiments is that they show low accuracy results for limited datasets. Moreover, none of the studies exhibited designed learning and did not focus on ensemble learning methods for Bengali offensive text identification. In our study, our main target is to develop the L-Boost model and: train and test the proposed model using available Bengali and Banglish datasets. We have implemented AdaBoost in a combination of baseline classification and reconstructed it by applying a bidirectional encoder representation of transformers with long short-term memory models.

III. METHODOLOGY

In this section, we propose an offensive text detection process using social media datasets and different classification algorithms. First, we applied some baseline methods such as support vector machine (SVM) with a linear kernel, decision tree (DT), and random forest (RF) to train and test the model for offensive text classification. These baseline methods are used when creating offensive text detection ensemble methods. Figure 1 shows the step-by-step process of the offensive text detection process using a baseline architecture. The TF-IDF vectorizer, word2vec model, fastText and BERT pre-trained embeddings were used for feature extraction and word matrix generation for text processing. Again, we use the popular ensemble method: Boosting (AdaBoost), using the baseline classification. We then discuss the proposed methodology. We modify the boosting (AdaBoost) method using a bidirectional encoder representation of transformers with long short-term memory (LSTM) models. We use the BERT transformer to focus on positional word embedding and custom fine-tuning during model training. In this study, we used different symbols, which are shown in Table 3.

A. DATA COLLECTION

Data collection from various social networking platforms can be done automatically, rather than developing a list of sentences manually to present an offensive text in Bengali. We used several Bengali websites, such as prothomalo.com, banglanews24.com, and social media platforms to collect offensive data. Figure 2 shows the data crawling process in which the BeautifulSoup [5] Python library is used to crawl all hashtags and emoji-related posts and comments from different social media. We collected 16,800 posts and comments and memes from different Bengali websites, blogs, and various social media platforms. We collected all comments and posts under a fixed-length size. We removed all permalinks, dates, times, and user details to ensure greater accuracy and privacy. We manually labeled crowded datasets to identify the offensive texts used to harass people. After crawling 16800 posts, Table 4 shows the data collection statistics. The dataset contains 4463 Bengali comments and 2480 Banglish comments, respectively. There were a total of 2932 offensive comments and 4011 non-offensive comments among all the Bengali and Banglish datasets. Among all datasets, 30% were set aside for testing purposes, and the rest were kept for training purposes.

B. DATA SET PREPROCESSING

Text preprocessing is the most distinctive part of automated text classification. Typical text preprocessing includes punctuation, commas, and case conversions, but in addition to these tasks, specific languages have their own syntactic and grammatical structures that need to be processed using unique methods and rules. The following text discusses the Bengali text preprocessing steps in detail.
Text processing is very important for identifying offensive content because there are multiple variations in the dataset, including emoji, emoticons, and words. Most of the comments in different social networks frequently use different types of emojis and emotions. Table 5 shows the top emojis and emoticons used for both the offensive and normal text. Most people use different types of emojis and emoticons during comments on social platforms to express their feelings and attitudes. Emoji and emoticons are very important criteria for identifying offensive comments. For example, #sexy_baby is an offensive text that can be identified primarily based on the female genitalia emoji symbol. The meaning of emoji is dependent on language, culture and location [55]. Some emojis have different meanings in Bengali whereas they have other meanings in English. For example, the “rooster” emoji in English usually refers to a specific bird, but in Bengali, the contextual meaning is male genitalia who is ready for sex. Table 6 shows the textual forms of some emoji symbols with the corresponding Bengali text representations. We convert emoji and emoticons into a textual format by extracting Unicode defined by Unicode Organization (Unicode.org). We used Python library package Beautifulsoup4 [5] to scrap all emojis and emotions defined in the Unicode Organization website where they provide all textual representations for each emoji and emotion. Then, we translated all emoji texts from English to Bengali using the Python Translate library package [56] and manually identified which emojis are objectionable and which are not objectionable in Bengali.

### TABLE 5. Top most 6 emoji and emoticons for offensive and normal text

| Emoji and Emotion (Emj) | Usages percentage of Emoji and Emotion (Emj) for Offensive text | Usages percentage of Emoji and Emotion (Emj) for normal text |
|------------------------|---------------------------------------------------------------|----------------------------------------------------------------|
| 🀄️ | 20% ;D | 42% | 🀄️ | 45% ;< | 15% |
| 🔥 | 14% ! | 7% | 🔥 | 4% ! | 27% |
| 🍁 | 10% ;< | 21% | 🍁 | 12% ;| | 11% |
| 😖 | 7% 😨 | 8% | 😖 | 9% ;> | 7% |
| 😥 | 5% :| | 6% | 😥 | 6% ;= | 5% |
| 😬 | 4% ;< | 3% | 😬 | 5% ;D | 4% |

1) Emoji and Emoticon Conversion

Text processing is very important for identifying offensive content because there are multiple variations in the dataset, including emoji, emoticons, and words. Most of the comments in different social networks frequently use different types of emojis and emotions. Table 5 shows the top emojis and emoticons used for both the offensive and normal text. Most people use different types of emojis and emoticons during comments on social platforms to express their feelings and attitudes. Emoji and emoticons are very important criteria for identifying offensive comments. For example, #sexy_baby is an offensive text that can be identified primarily based on the female genitalia emoji symbol. The meaning of emoji is dependent on language, culture and location [55]. Some emojis have different meanings in Bengali whereas they have other meanings in English. For example, the “rooster” emoji in English usually refers to a specific bird, but in Bengali, the contextual meaning is male genitalia who is ready for sex. Table 6 shows the textual forms of some emoji symbols with the corresponding Bengali text representations. We convert emoji and emoticons into a textual format by extracting Unicode defined by Unicode Organization (Unicode.org). We used Python library package Beautifulsoup4 [5] to scrap all emojis and emotions defined in the Unicode Organization website where they provide all textual representations for each emoji and emotion. Then, we translated all emoji texts from English to Bengali using the Python Translate library package [56] and manually identified which emojis are objectionable and which are not objectionable in Bengali.

2) Hashtag Segmentation

Hashtags are frequently used when commenting on discovering a post on social platforms such as Facebook and Twitter. If we consider all hashtag texts, it will be very effective in identifying offensive text. Therefore, during preprocessing we replace the space by removing all hashtag symbols (#) and saving all text for future
use. Hashtag can be in English or Bengali format, but we translate all hashtag text into the Bengali language. For example: one commenter uses #Boycott hashtag in English, at first, we remove the (#) symbol then translate it to Bengali #Boycott => Boycott => বয়কট. Then, we store “বয়কট” for understanding and processing in the next step.

3) Miscellaneous
We use other common text preprocessing methods, such as converting numbers into words, removing punctuation, white space, accent marks, stop words, etc. In the Bengali language, there is no specific stop word like the English language. This depends on a specific task. For example, in sentiment analysis we can use যান, সিহত, দেখ, দেখিয়েছ, দেখিয়ে, দেখেছ, দেখেছ, ডিট, ডিট, ডিট, ডিট, ডিট, ডিট, ডিট, হচ্ছ, দেখা, করার, করে, করে, বয়কট,” etc. words as stop words but for offensive text detection we can’t use these words as stop word because these words are used to make offensive sentence. For example, “সে বিবেক উপের তেলে কারে দিয়েছ।” He puts Bobby on top and fucks him. This is an offensive sentence in which these words are considered the main words of this sentence. We manually identified all stop words that were not offensive content and used the Python library and regex to remove unwanted words and punctuation from the text. In our research, we converted all Banglish datasets into the corresponding Bengali format during preprocessing. GAMITISA\(^1\) has been used in Bangllo to Bengali transliteration. GAMITISA is an online free tool with rich Bengali and Banglish vocabulary that converts Banglish words into corresponding Bengali words.

4) Data Labelling
The data labeling process is shown in Figure 3 where eight graduate students from the Advanced Machine Learning (AML) lab in Bangladesh Business and Technology University (BUBT) manually labeled all the data. The data labels are checked by the expert team of the Advanced Machine Learning (AML) lab. Professor or Ph.D. students with more than five years of experience or any researchers with extensive knowledge in the Bengali language processing area are considered as experts in the AML lab. Table 7 presents some examples of sample datasets, processed datasets after preprocessing and translation, and labeling of each process sentence. From these examples, we remove একটা, দিয়েছ, দিয়েছ, ইশ, আজেকর, হবে, গেলো, হবে as stop words during text preprocessing.

C. FEATURE ENGINEERING
The input values of each text-classification model must be a number vector representing a text document. When the words in a text document are encoded as floating-points or binary values of fixed-length vectors, this is known as feature extraction or vectorization. In this study, we used TF-IDF [21], Word2vec [22], fastText [19] and BERT [20] embedding to extract features.

1) TF-IDF vectorizer
When processing a huge text datasets, there are some words that occur repeatedly but contain inadequate necessary information. These data usually overshadow the frequency of more important data during text processing, including calculations. The TF-IDF feature extraction technique can resolve this problem. This can be defined as follows:

\[
\text{tf–idf}(t, d) := tf(t, d) \times \text{idf}(t)
\]

where, \(tf(t, d)\) is the term frequency, and \(\text{idf}(t)\) is the inverse document frequency.

2) Word2vec
Word2Vec is used to determine the vector of all terms in a manner in which the alike text has related vectors. It can be used in semantic analysis of texts. It is a two-layer neural network. Its input is a text corpus and its output is a set of vectors: feature vectors for words in the corpus. We used the Python scikit-learn library to implement these feature vectors.

3) FastText
In 2016 Facebook proposed the FastText word embedding model which is an extension of the Word2Vec model.
TABLE 7. Dataset Examples

| Example                      | Transliteration | Actual Meaning                                      | Data Label |
|------------------------------|-----------------|-----------------------------------------------------|------------|
| তুই একটা লুচা রাজাকার, পাখ দেখলে নিজের সামনে পাড়া না। | nirbacito ghar ghar dake | You are a naughty razakar, you can’t control yourself when you see woman | Offensive |
| Borke chade নয়োঁ চোর মেয়ে অম্বে হাড়ে বাঁধ দিল। | push the groom on the roof and insult him with Hog Plum in his hand | Offensive |
| েহি ব্রিটি কারেন আঁজের কেটা মাছতি বুড়ি হাড়ে গেলা। | Illegal today’s match has been postponed due to rain | Non-offensive |
| Abaidha শ বাঙালিদের ঘাঁটা নিয়ে nirbacito করা হবে। | Illegal Bangladeshis will be deported by pushing their necks (joke) | Offensive |

FIGURE 4. Pre-trained bi-directional BERT model

FastText splits the words into different sub-words and calculates the vector for all sub-words and then the sum of all sub-word vectors is the word embedding vector of the main word.

4) BERT

The Google AI research team developed a bidirectional encoder representation of transformers (BERT) deep learning-based bidirectional unsupervised model for text processing. The formation of the BERT transformer is a multilevel bidirectional transformer encoder. Deep learning model-based transformers are used as encoders and decoders for translation purposes. The sample BERT diagram is shown in Figure 4, where $E_1, E_2, ..., E_n$ are the inputs of the BERT model. The input can be a set of tokens, special symbols, words, etc. Several multilayer transformers exist after the input level. These transformers are bidirectional transformers used for encoding input text and producing the corresponding output vectors. BERT is another popular pre-trained word-embedding model that is applied to create token embedding. BERT supports token embedding, segment embedding and different types of position embedding as shown in Figure 5. In this study, we used the BERT positional embedding method to extract the Bengali and Banglish text features. The main objective of position embedding is to bind the pursuit nature of the word position in the sentence matrix. Position embedding has two different forms: absolute position embedding (AE), where every position is mapped to the space elements of the representation. Another form is relative position embedding (RE), where the positional distance (i.e., $x - y$ for $x, y \in \mathbb{N}$, where $\mathbb{N}$ is the set of all word position) among words in a sentence are mapped to the embedding space.

The BERT transformer calculates attention by considering the weighted value ($W_v$) weighted query ($W_q$) and weighted key ($W_k$) in the attention head. For example, let $x \in \mathbb{N}$ and $y \in \mathbb{N}$ are two positions, and $W_{Vx}$ is the word vector for the $x$ positional word, $E_x$ is the embeddings for the $x$ position, and $E_{x-y}$ is the embedding of the relative position. Then, the $q, v, k$ vector for the $x$ positional word can be calculated as follows:

$$AE: \begin{bmatrix} q_x \\ k_x \\ v_x \end{bmatrix} = (W_{Vx} + E_x) \times \begin{bmatrix} W_q \\ W_k \\ W_v \end{bmatrix} \quad (2)$$

$$RE: \begin{bmatrix} q_x \\ k_x \\ v_x \end{bmatrix} = (W_{Vx} + E_x) \times \begin{bmatrix} W_q \\ W_k \\ W_v \end{bmatrix} + \begin{bmatrix} 0 \\ E_{x-y} \\ E_{x-y} \end{bmatrix} \quad (3)$$

The sum of all attention($a$) head value is the final result where attention weight depends on $a = qk^T$, therefore:

$$Attention(q, k, v) := softmax(qk^T / \sqrt{d_k})v \quad (4)$$
FastText, BERT, TF-IDF and Word2Vec feature extraction methods were used to find the word vectors of all datasets. Sample feature values for all feature extraction techniques are depicted in Table 8 which shows five words \((w_1, w_2, w_3, w_4, w_5)\) from four text documents \((t_1, t_2, t_3, t_4)\). To reduce the size of a two-dimensional array, we considered the top 768 featured words in each text document where rows are considered text documents \((t_1, t_2, ..., t_{15000})\) and columns are considered as feature words \((w_1, w_2, ..., w_{768})\) of each text document. This means that we used a fixed-length word vector with a size of 15000 x 768.

TF-IDF measures the importance of a word in a document and calculates the score of a particular word, where word2vec combines all the senses of a word to create only one vector. Word2vec cannot work outside of the vocabulary where fast text can fix the word2vec constraint, and BERT uses attention-based positional encoding to find word vectors. The Bengali text in this article was processed using the fastText pre-trained Bengali word embedding vector [19], and BanglaBERT based model [13], a pretrained language model constructed with BERT-based mask language modeling. All feature extraction methods use various n-gram features to calculate the word matrix as shown in Table 9. To identify offensive text from a document and to find the most suitable model for detecting offensive text, we have used a variety of combinations of n-gram properties with feature extraction techniques.

**D. PROPOSED L-BOOST MODEL**

Our proposed model has 3 parts. First, we applied the BERT embedding with an LSTM-based deep neural network. Figure 6 shows the LSTM model for offensive text prediction where there is an input layer, a BERT embedding layer, two LSTM and dense layer, and an output layer. The first LSTM layer produces the output of all hidden states and then the next LSTM layer produces the last hidden state as the output of the LSTM network. We would like to evaluate the focus of attention on word-level tasks, but BERT applies byte-pair tokenization, which means that certain words are divided into multiple sub-words as tokens. Therefore, we exchange (token, token) attention maps into (word, word) attention maps. To focus on the sub-word, we add the weight of the attention to its tokens. We take the average of the weights of attention on the sub-word, we add the weight of the attention to its tokens. We exchange (token, token) conversions maintain the quality in which the amount of attention from each word is 1.
LSTM defends the error of propagating backward using hidden layers where learning continues through different steps in the training session. LSTM was constructed to train the long-distance dependence between hierarchical data. Each LSTM cell, which integrates input, forgotten, and output gates, is assigned to capture, cancel, adjust, and decide the next step in retrieving a fraction of the data. The LSTM cell decides when access permissions are read, written, and deleted when data are transferred or blocked across the LSTM unit through the gates as illustrated in Figure 7. The functionality of the gates and the flow of information can be expressed using Equations 5 - 10.

\[
\begin{align*}
    f_{gt} &= \sigma(W_{fg}[hv_{t-1}, x_t] + b_{fg}) \\
    i_{gt} &= \sigma(W_{ig}[hv_{t-1}, x_t] + b_{ig}) \\
    cu'_{t} &= \tanh(W_{cu}[hv_{t-1}, x_t] + b_{cu}) \\
    cu_{t} &= f_{gt} \times cu_{t-1} + i_{gt} \times cu'_{t} \\
    o_{gt} &= \sigma(W_{og}[hv_{t-1}, x_t] + b_{og}) \\
    hv_{t} &= o_{gt} \times \tanh(cu_{t})
\end{align*}
\]

where: 
- \(x_t\) = input vector in time \(t\),
- \(hv_{t}\) = output vector,
- \(cu_{t}\) = store state information,
- \(i_{gt}\) = vector of input gate,
- \(o_{gt}\) = vector of output gate,
- \(f_{gt}\) = vector of forgotten gate,
- \(W_{*}\) = weight of different gate,
- \(b\) = shift vector and
- \(\sigma\) = activation function

All specific baseline classifiers show reasonable output with low accuracy scores, but these results must be filtered to increase the accuracy. In Section IV, we discuss this issue further. This means that baseline methods cannot completely learn an entire set of features from a dataset. This is because the information is stored from different websites or sources, where a single model cannot capture decision boundaries for all aspects. It may also indicate that the extraction of the uni-gram feature was not sufficient to present the latest signification of the abusive text or general text. Ensemble methods allow a combination of baseline classification and various other methods for sampling features during the training phase. The most well-known ensemble methods are bagging and boosting [57].

The boosting ensemble process was used to improve the accuracy by training weak learners sequentially. Boosting has both advantages and disadvantages. This can enhance the classification performance; however, it takes time for large datasets to train the model. The goal of the AdaBoost [14] boosting ensemble algorithm is to increase the results of the weaker classifiers by calculating the errors and updating the weights of the input data. In the ensemble process, a new classifier is added iteratively, and weak classifiers are identified from the dataset. Each of the newly added classifiers then qualifies for the previous iteration with a class of data for which the performance of the classifiers was very low. Another popular ensemble method is bagging, which generates a sample of a certain size from the original dataset. The bagging ensemble process trains classifiers for all dataset samples. It detects the possibilities and averages them. This procedure is run for each individual class on all baseline classifications in the ensemble techniques.

The second part of the proposed model is shown in Figure
8. AdaBoost with a custom BERT fine-tuning model with different strategies was applied to boost the model. In the weight allocation phase, this model incorporates BERT into the boosting architecture. We utilize the fine-tuning weight initialization approach to dynamically assign the parameters of transformer when adding a new base classifier to the ensemble in order to include the pre-trained language encoder BERT into the boosting. The BERT architecture has an encoder in which each encoder has 12 blocks of transformers, 12 self-focusing heads, and 768 hidden size representations. BERT supports up to 512 tokens of an input sequence where one or more segments can be considered in a sequence. In the BERT model, the first token must be [CLS] which is used for finding specific embedding and the model uses another token [SEP] for segment separation. BERT considers the last hidden state \( h \) of a sequence start with [CLS] to find specific meaning from a sequence during text classification. The multilayer perceptron (MLP) [58] model is used on top for every transformer encoder to improve the ensemble of each dominant BERT classifier. During the MLP network's training, the properties of each base classifier are preset. The MLP architecture is made up of multiple hidden layers that use each transformer encoder's output as an input layer. We add the weight of each base classifier to its output to exploit this knowledge since the base classifier has varying confidence scores on certain tasks. A dense layer is then added to the top of the last layer where the softmax classifier is used to predict the probability of a specific label \( S \).

\[
\text{Prob}(S|h) = \text{softmax}(Th)
\]

(11)

where \( T \) is the parameter matrix for a specific task. We use AdaBoost with multiple fine-tuning approaches to fine-tune the properties to maximize the probability of a specific label \( S \). We consider some factors during fine-tuning strategies to capture different syntactic and semantic levels for offensive text classification.

1) Long text pre-processing
BERT supported the highest sequence length of 512. To process the highest-length text, we applied the following ways in different iterations of the AdaBoost model.

Cutting approach: Typically, an article contains some basic information from the beginning and end of the text. In this proposed model, we applied three ways to cut text for BERT fine-tuning.

- Choosing head: Select the first 512 tokens from the beginning of the text
- Choosing tail: Select the last 512 tokens from the end-part of the text
- Choosing head and tail: Select the first 256 tokens and last 256 tokens to consider both head and tail information.

Hierarchical Approach: In this method, text longer than 512 tokens is subdivided into sub-texts. In our model, we applied \( t(\text{subtext}) = \text{fulltext}/510 \) for text splitting. Then each sub-text is fed into the BERT model for proper fine-tuning. Finally, several pooling methods, such as max pooling, mean pooling and self-attention have been applied to gather all sub-texts for finding the representation of the article.

2) The most effective layer selection
In the BERT transformer model different layers are used to capture different properties during offensive text classification. We examined the usefulness of features from distinct layers. We fine-tuned the model based on the performance of the test error rates of a specific layer.

3) Handle overfitting problem
We used different learning relates during fine-tuning to minimize the overfitting problem. We observe that in the bottom layer, a lower learning rate \( lr < 2.5e^{-5} \) performs best during BERT fine-tuning.

Generally, hybrid approaches perform better than normal single methods. In view of the hybridity and brilliant prediction abilities of the two models, we merge these two models to obtain the highest prediction accuracy. We refer to this model as the L-Boost model where Figure 9 illustrates the combined model. In our proposed architecture, the offensive texts are first predicted separately using the LSTM-BERT and AdaBoost-BERT models. The output results of the two models were averaged to generate the final prediction results. Then the dataset sequence is updated according to the results of the final forecast, which is used as the input of the model in the next iteration.

IV. RESULT ANALYSIS
The main objective of this research is to test the effectiveness of various machine-learning algorithms for combining different features. We present several graphical and statistical performance evaluation methods to discover a suitable model that can successfully classify offensive text. We used Ubuntu 18.04 OS for Intel (R) Core (TM) i5-6500CPU machine with 16GB of RAM to set up our experiments. We use Python 3.6.9 version with TensorFlow 2.2.1 to implement all offensive text classification models and we used Panda 1.0.3 data frame
and scikit-learn 0.22.2 to create datasets for training and testing purposes. During the process of creating datasets for training and testing, we randomly shuffled all offensive and general text so that the training and test datasets contained a mixture of offensive and non-offensive text. K-fold cross-validation was used to perform benchmarks on datasets, where K = 5. Hence, the dataset was divided into 80%-20% training-testing subsets. We implemented the model described in Section III.D and the list of parameters and hyperparameters used in this model is shown in Table 10.

A. PERFORMANCE ANALYSIS PARAMETERS

We evaluated our system performance based on accuracy, precision, recall, and F1-score.

1) Accuracy(A)

This is a mathematical measurement. This indicates that any classification was accurately measured. This is known as the consistency of the actual results for the number of samples tested.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (12)
\]

where, \(TP\) = True Positive; \(FP\) = False Positive; \(TN\) = True Negative; \(FN\) = False Negative.

2) Precision(P)

Precision is the ratio of correctly predicted positive observations to the total number of predicted positive observations.

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (13)
\]

3) Recall(R)

Recall is the ratio of correctly predicted positive observations to all observations in the actual class

\[
Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \quad (14)
\]

4) F1-score(FS)

The F1-score represents the weighted mean of precision and recall. Therefore, this score considers both false positives and false negatives. Intuitively, accuracy is not as easy to understand, but F1-score is usually more effective than accuracy, specifically if the class distribution is unequal. Accuracy works best when there is the same cost for false positives and false negatives. If the cost of false negatives and false positives are very different, it is preferable to pay attention to both precision and recall.

\[
F1 - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (15)
\]

B. BASELINE EVALUATION

A baseline test was performed using the Python machine learning library. We used binary classification processes using three baseline classifiers and a deep neural network model for text classification. These are support vector machine (SVM), decision tree (DT), random forest (RF) and long short term memory (LSTM). The SVM classification technique demonstrates excellent performance with a small amount of data. All feature vectors were organized in a high-dimensional space. Hyperplanes search for the minimum steps to detach space so that the point belongs to different classes. Sometimes, multiple hyperplanes are used to find the target hyperplane, which can maximize the distance between classes. The decision tree classification algorithm is used in a wide variety of cases for taxonomy. The structure of this strategy is a hierarchical fragmentation of the data space.

The scikit-learn machine learning library was used for the baseline assessment. Different classifiers have different parameters and the parameters depend on the size and model of the input data. \(max\_features = n\_features\) used for DT and RF classifiers, but the \(entropy\) criterion is used in DT, where RF uses the \(Gini\) criterion. We used \(learning\_rate=optimal\) and \(random\_state=0\) for all classifiers. In this section, we discuss the performance analysis of all baseline classifiers in both statistical and graphical formats.

| 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 |
TABLE 12. Performance Results of different baseline classifiers (The underline indicates the top score among the various features)

| Vector Model | Features | RF Classifiers | DT Classifiers | SVM Classifiers | LSTM Classifiers |
|-------------|---------|---------------|---------------|----------------|-----------------|
|             |         | A(%) | FS(%) | A(%) | FS(%) | A(%) | FS(%) | A(%) | FS(%) |
| TF-IDF      | F1      | 82.76 | 84.95 | 79.68 | 82.62 | 84.57 | 85.94 | 87.84 | 89.25 |
|             | F2      | 80.74 | 83.58 | 80.28 | 83.11 | 83.61 | 85.13 | 90.41 | 91.53 |
|             | F3      | 79.57 | 78.15 | 79.92 | 82.62 | 84.32 | 85.59 | 92.27 | 93.09 |
|             | F4      | 83.88 | 85.21 | 81.40 | 84.62 | 89.05 | 89.79 | 93.83 | 94.50 |
|             | F5      | 87.74 | 88.63 | 80.63 | 83.58 | 86.04 | 87.29 | 92.76 | 93.52 |
| Word2Vec    | F1      | 90.96 | 91.66 | 76.52 | 78.60 | 83.04 | 83.39 | 89.92 | 90.57 |
|             | F2      | 83.53 | 86.19 | 78.26 | 80.33 | 84.65 | 85.09 | 92.90 | 93.48 |
|             | F3      | 83.61 | 86.12 | 75.83 | 78.86 | 90.55 | 90.96 | 92.43 | 93.07 |
|             | F4      | 82.76 | 82.47 | 77.85 | 78.78 | 83.99 | 84.16 | 91.94 | 92.19 |
|             | F5      | 84.57 | 86.64 | 75.74 | 78.75 | 84.05 | 84.53 | 83.15 | 84.51 |
| fastText    | F1      | 84.73 | 86.69 | 86.23 | 88.07 | 84.95 | 87.21 | 93.53 | 94.31 |
|             | F2      | 84.79 | 84.43 | 82.66 | 85.10 | 85.69 | 87.55 | 93.83 | 94.53 |
|             | F3      | 85.91 | 87.52 | 83.42 | 85.61 | 85.99 | 87.90 | 92.30 | 93.38 |
|             | F4      | 86.62 | 88.08 | 85.58 | 86.90 | 86.70 | 88.21 | 91.37 | 92.05 |
|             | F5      | 86.18 | 87.62 | 84.10 | 85.78 | 86.56 | 88.16 | 90.88 | 91.96 |

1) Statistical Evaluation

We implemented all baseline classification algorithms for the three feature extraction models with five different types of features. Table 11 lists the different features with their corresponding values. We used several combinations of features from Table 9 and selected the top five features with the highest accuracy score. Table 12 shows the performance results of the baseline classification where each classifier is applied separately to the TF-IDF, Word2Vec, and fastText feature extraction methods.

The performance results of the RF classification algorithm is shown in Table 12, where the maximum accuracy value for the Word2Vec feature extraction (FE) strategy and the F1 feature is 90.96%, and the lowest score is 79.57% for the F3 attribute with the TF-IDF FE method. The LSTM technique performed best for F4 features with the TF-IDF FE technique.

For the DT classifiers, all FE techniques show the nearest accuracy score for all features as shown in Table 12. The F1 feature with the fastText FE technique performs best among all FE techniques and features, which is 86.23%. The highest recall (R) score was 96.60% for the F4 feature with TF-IDF, and the lowest recall score was 78.88% for the F4 feature with the Word2Vec FE technique. Table 12 presents the SVM classifier performance accuracy scores of different features for TF-IDF, Word2Vec, and fastText FE techniques. SVM classifiers showed a 90.58% accuracy score for the F3 feature with Word2Vec FE techniques, and the lowest score was 83.04% for the F1 attribute with the Word2Vec FE method. FastText feature extraction techniques have similar accuracy scores for the SVM classification method. We have applied the LSTM deep neural model as a baseline classifier. LSTM classifiers achieved the highest 93.83% accuracy score for the fastText FE technique with F5 features, which is the highest score for F1 features among all classification techniques. This algorithm with TF-IDF FE techniques performs better than other Word2Vec FE techniques, as shown in Table 12.

The F1-score considers all the limitations of the accuracy metric which is more powerful than the accuracy assessment metric of machine learning. Table 12 shows the F1-scores of all baseline classification algorithms of the three feature extraction techniques. In the DT classification algorithm, the highest F1-score for the offensive class is derived from the F1 attribute using the fastText FE method, and the lowest F1-score comes from the F5 attribute with the Word2Vec FE method. The F1-score in SVM classification for different properties with TF-IDF attribute extraction methods shows better results than other attribute extraction methods.

The F1-score of offensive and non-offensive classes in the LSTM classification algorithm has better performance for all n-gram features with different feature extraction techniques. The RF classification algorithms show that the F1-score is very unstructured and the F1-score is too low for the class that is offensive in RF classifiers. The F1-score of non-offensive text in the DT classification was very low for the F3 and F5 features with the Word2Vec FE method and the highest in the fastText FE method with the F1 feature. For the TF-IDF FE technique, SVM with an F4 feature showed the highest F1-score among all core baseline classification algorithms. The F4 features in the LSTM model with the TF-IDF FE technique shows 95.02% F1-score. RF achieved 86.9% F1-score for F5 features. The highest F1-score limit in the TF-IDF strategy was 95.2%,
FIGURE 10. ROC curve for TF-IDF, Word2Vec, and fastText with different features
and the lowest F1-score limit was 78.01% for the offensive class. In contrast, the IF-IDF strategy has a maximum F1-score of 90.05% for the F4 feature and a minimum score of 33.35% for the F2 feature. The F1-score of different classifiers with the Word2Vec FE technique shown in Table 12, where RF and SGD achieved the highest score of 94.85% and MNB classifier with feature F2 achieved the lowest (55.0%) F1-score.

2) Graphical Evaluation

Receiver operating characteristic (ROC) curves are often employed to exhibit the graphical method of the relationship between sensitivity and specificity for a possible cut-off for an experimental combination. Figures 10(a)-(e) show the ROC curve analysis of the TF-IDF feature extraction method. SVM provided the best area under the ROC curve (AUC) of 88.9% with F4 among the core baseline classifiers. For the deep neural-based classifier, LSTM produced 93.4% AUC for F4 features under the TF-IDF feature extraction technique. The other two offensive classification algorithms, DT and RF, using the TF-IDF FE technique, yielded good results for different feature characteristics.

ROC curves for the Word2Vec feature extraction technique are shown in Figures 10(f)-(j), where the highest AUC value is 90.7% for RF classifiers with F1 feature and lowest is 75.2% for the DT classifiers with F3 and F5 features. The LSTM classification method produces a 92.8% AUC for the F2 feature, which is the highest AUC for the Word2Vec feature extraction technique. The SVM classifier performed overall good results for all features of F1 to F5. Other text classification algorithms, such as RF techniques, perform the quality results.

Similarly, Figures 10(k)-(o) show the ROC curve for the fastText feature extraction technique with different properties F1 to F5. Using the fastText feature extraction technique, the ROC curve for the F2 feature shows that all classification algorithms perform better AUC than TF-IDF and Word2Vec feature extraction techniques. The maximum AUC value of 86.3% was achieved using SVM classifiers with the F4 feature. The RF classification algorithm achieved 86.2% AUC value using the F4 feature. The SVM and RF algorithms yielded overall satisfactory AUC results for all features of F1 to F5. The highest AUC score was 93.4% achieved for the F2 feature in the LSTM model.

To find the best feature extraction model, we applied conventional machine learning algorithms using the whole sentence instead of various features to calculate performance. Among the four feature extraction technologies, BERT showed satisfactory scores for all types of classification algorithms as shown in Figure 11. The fastText feature extraction technique performs optimally for some classifiers and also gives poor results for some machine learning algorithms and shows that it is not stable for the offensive text classification. In addition, the Word2Vec feature extraction technique requires a large amount of time for big datasets, and the accuracy score is poor compared to the TF-IDF feature extraction method. For the above reason, our proposed method uses BERT feature extraction techniques to process large datasets and extract useful information from chaotic datasets. To further clarify, in this article, we used different N-Grams for the traditional machine learning models and full text for the BERT-based model.

C. ENSEMBLE EVALUATION

We chose the LSTM deep neural classification algorithm to create the proposed offensive text detection model. We used the BERT FE technique with the LSTM classification algorithm to reduce the problem of overfitting and underfitting in machine learning. We used a method of 10 cross-validations to evaluate our ensemble classification scheme. All classifiers were trained recurrently in this manner. We used the same number of datasets to test and train the simulation method used for the previous baseline classification. The results were calculated after 10 repetitions. Table 13 presents the performance results of the proposed model. The LSTM-BERT achieves an accuracy score of 89.45%, and AdaBoost-BERT achieves a 92.16% accuracy score on our dataset. After updating the dataset based on the prediction scores of the two models, the L-Boost model performed the best. The L-Boost model showed an accuracy of 95.11%, where the Bangla BERT based baseline classifier showed the highest accuracy of 88.47%, and conventional machine learning-based baseline classifiers showed the highest accuracy score of 93% for the F2 feature, but the proposed method shows a 2% improvement over the accuracy score of 95.11%.

Also, the model was trained and evaluated on a comprehensive Bengali Hate Speech dataset [12] and transliterated Bengali-English dataset [8] to demonstrate the importance of the proposed architecture. We have evaluated L-Boost using the Bengali Hate Speech dataset [12] and the results are encouraging. The L-BOOST architecture can be seen in Table 14 by all machine learning and deep learning baseline algorithms, as well as the DeepHateExplainer architecture by substantial margins. RF has the highest F1-score (68%) of all of the five machine learning baselines. Among the deep learning baselines (CNN, Bi-LSTM, and Conv-LSTM) Conv-LSTM has the greatest F1-score of 78 percent. In the translated Bengali corpus [8], Table 14
highlights the performance of several ML classifiers for identifying offensive language. The recall and precision scores of LR and SVM are comparable to the three classic ML classifiers. The recall score of the RF classifier is similar to that of LR and SVM, but its precision score is lower. It is worth noting that both LR and SVM provide lower recall scores than precision scores. The deep learning-based architecture BiLSTM achieves a lower F1-score than LR and SVM, which might be ascribed to the corpus's limited size (i.e., 3000 comments).

In addition, Table 14 shows that L-BOOST outperforms the translated Bengali corpus by a good margin. In our proposed architecture, it first converts the entire dataset into the corresponding Bengali format and then applies various pre-trained feature extraction methods and classification methods to detect offensive text. Since we have Bengali and Banglish words in our dataset, we convert all Banglish words into corresponding Bengali format before applying the feature extraction method. However, when we try to convert all the words of [8] to the corresponding Bengali format, the online tool cannot convert some English words to the corresponding Bengali format. For this reason, the L-Boost model shows better accuracy on our dataset than transliterated [8] dataset.
Also, transliterated Bengali-English [8] dataset size is limited (i.e., 3000 comments) compare to our dataset (i.e., above 6000 comments).

The Friedman test [59], [60] and the Nemenyi post-hoc test [59], [60] were used to make a more detailed comparison of the performance of the models mentioned above. Table 15 shows the ranking values of the different baseline models, and Table 16 shows the comparison ranking of methods across different datasets. To compare the average rank of various approaches, the Friedman experiment was utilized. The following Equations 16 and 17 are used to calculate the Friedman statistics $\tau_F$:

$$\tau_F = \frac{(N-1)\tau_x^2}{N(k-1)-\tau_x^2}$$  \hspace{1cm} (16)

$$\tau_x^2 = \frac{12N(k+1)}{k(k+1)}\left(\sum_{i=1}^{k}i^2 - \frac{k(k+1)^2}{4}\right)$$  \hspace{1cm} (17)

Where $N$ represents the total number of datasets and $k$ represents the total number of methods. Using the above equation, for the baseline classification we get $\tau_F = 6.6$ and the $p$-value is 0.086, while $\alpha = 0.05$. In Friedman test, if $p$ is less than $\alpha$ then performance of all methods will be the same. Since $p > \alpha$, which means all models have markedly varied performance characteristics. Also, Friedman's statistic ($\tau_F$) is 4.4 and the value of $p$ is 0.34 for all existing and proposed methods which are greater than $\alpha$. The Nemenyi test was then used to further differentiate between the models. According to the Nemenyi test, the critical difference (CD) is defined as:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$  \hspace{1cm} (18)

Figure 12 and 13 shows the visual representation of the Friedman test and the Nemenyi post-hoc analysis results, wherein the average rankings of each technique are indicated through the horizontal line. It has been flipped such that the lowest (best performance) ratings are on the left and the highest (worst performance) on the right. Our L-Boost was statistically the finest of the examined approaches, as shown in Figure 13. In addition, according to the Nemenyi test, the performance of L-Boost and RF differed slightly.

Finally, Table 17 shows a few examples from our testing dataset where the proposed L-Boost system incorrectly predicted a sentence. In Table 17, the third example is labeled as offensive because the contextual meaning indicates that there are some offensive conversations but our L-Boost system cannot detect the text as offensive text because there are no offensive words in this text so, L-Boost incorrectly predicted the class.
V. CONCLUSION
We presented two different types of machine learning tests, baseline and ensemble analysis, for offensive text classification. In the baseline testing, we used different types of machine learning and deep learning text classification algorithms. Of all the classification algorithms, LSTM showed overall good performance for all features, and SVM showed average performance for all features. We used four feature extraction algorithms to extract useful features from the random databases. Among all feature extraction algorithms, BERT showed better performance than the others. We have used five types of N-gram properties and applied all classification algorithms to different properties, including different feature extraction methods. We then selected LSTM with BERT text classifiers to reduce the underfitting and overfitting problems in the classification of machine learning. For the ensemble testing, we used the AdaBoost algorithm with different types of BERT fine-tuning classification models. We have modified the AdaBoost with BERT transformer to learn to ensemble classification of machines that show excellent performance compared to any other efforts developed in offensive text detection. We anticipate that this research work will be further improved if some issues are addressed. First, setting up data for case studies might be sufficiently diversified to achieve excellent results. Second, since people from different places speak different versions of the Bengali language, there is a need to integrate those phenomena. Finally, endeavors may be exercised for the detection of offensiveness from any format of Bengali content, such as images, pdf, videos, or any speech.

ACKNOWLEDGEMENT
We would like to thank the Advanced Machine Learning (AML) lan and Bangladesh University of Business and Technology (BUBT) for their support.

REFERENCES
[1] L1ght Organization. Rising levels of hate speech online toxicity during this time of crisis. 1light.com/Toxicity_during_coronavirus_Report-L1ght.pdf, 2020.
[2] Daniel Chandler and Rod Munday. A dictionary of social media. Oxford University Press, 2016.
[3] Sara Sood, Judd Antin, and Elizabeth Churchill. Profanity use in online communities. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 1481–1490, 2012.
[4] Rogers Pelle, Cleber Alcântara, and Viviane P Moreira. A classifier ensemble for offensive text detection. In Proceedings of the 24th Brazilian Symposium on Multimedia and the Web, pages 237–243, 2018.
[5] Iqbal H Sarker, ASM Kayes, Shahzir Badsha, Hamed Alqahtani, Paul Watters, and Alex Ng. Cybersecurity data science: an overview from machine learning perspective. Journal of Big Data, 7(1):1–29, 2020.
[6] Jochen Hartmann, Juliana Huppertz, Christina Schamp, and Mark Heittmann. Comparing automated text classification methods. International Journal of Research in Marketing, 36(1):20–38, 2019.
[7] Omar Sharif, Mohammed Moshiul Hoque, ASM Kayes, Raza Nowroz, and Iqbal H Sarker. Detecting suspicious texts using machine learning techniques. Applied Sciences, 10(18):6527, 2020.
[8] Salim Sazzed. Abusive content detection in transliterated bengali-english social media corpus. In Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 125–130, 2021.
[9] Md. Anwar Husien Wadud and Md. Rashadul Hasan Rakib. Text coherence analysis based on misspelling oblivious word embeddings and deep neural network. International Journal of Advanced Computer Science and Applications, 12(1), 2021.
[10] Monisha Kanakaraj and Ram Mohana Reddy Gudetti. Performance analysis of ensemble methods on twitter sentiment analysis using nlp techniques. In Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), pages 169–170. IEEE, 2015.
[11] Rohi Polikar. Ensemble learning. In Ensemble machine learning, pages 1–34. Springer, 2012.
[12] Md Karim, Sumon Kanti Dey, Bharathii Raja Chakravarthi, et al. Deepfakeexplainner: Explainable hate speech detection in under-resourced bengali language. arXiv preprint arXiv:2012.14355, 2020.
[13] Sagar Sarker. Banglabert: Bengali mask language model for bengali language understanding, 2020.
[14] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780):1612, 1999.
[15] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. Statistics and its Interface, 2(3):349–360, 2009.
[16] Paul Viola and Michael J Jones. Robust real-time face detection. International journal of computer vision, 57(2):137–154, 2004.
[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[18] Klaus Greff, Rupesh K Srivastava, Jan Koutnìk, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10):2222–2232, 2016.
[19] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. Fasttext: Compressing text classification models. arXiv preprint arXiv:1612.03651, 2016.
[20] Nile Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084, 2019.
[21] Gan Jie and Chen Li-choa. Research of improved if-idf weighting algorithm. In The 2nd International Conference on Information Science and Engineering, pages 2304–2307. IEEE, 2010.
[22] Xin Rong, word2vec parameter learning explained. arXiv preprint arXiv:1411.2738, 2014.
[23] Marti A. Hearst, Susan T Dumas, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. IEEE Intelligent Systems and their applications, 13(4):18–28, 1998.
[24] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3):660–674, 1991.
M. F. Mridha et al.: L-Boost: Identifying Offensive Texts from Social Media Post in Bengali

[57] Leo Breiman. Bagging predictors. Machine learning, 24(2):123–140, 1996.

[58] Sankar K Pal and Sushmita Mitra. Multilayer perceptron, fuzzy sets, classification. 1992.

[59] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 7:1–30, 2006.

[60] Jun Xie, Bo Chen, Xinglong Gu, Fengmei Liang, and Xinying Xu. Self-attention-based blstm model for short text fine-grained sentiment classification. IEEE Access, 7:180558–180570, 2019.

M. F. Mridha (Senior Member, IEEE) received the Ph.D. degree in AI/ML from Jahangirnagar University, in 2017. He joined the Department of Computer Science and Engineering, Stamford University Bangladesh, in June 2007, as a Lecturer, where he was promoted a Senior Lecturer and an Assistant Professor, in October 2010 and October 2011, respectively. Then, he joined UAP, in May 2012, as an Assistant Professor. He is currently working as an associate professor with the Department of Computer Science and Engineering, Bangladesh University of Business and Technology. He also worked as a faculty member of the CSE Department, University of Asia Pacific, and as a graduate coordinator, from 2012 to 2019. His research experience, within both academia and industry, has resulted in over 80 journal and conference publications. His research interests include artificial intelligence (AI), machine learning, deep learning, big data analysis, and natural language processing (NLP). For more than ten years, he has been with the masters and undergraduate students, as a supervisor of their thesis work. He has served as a program committee member of several international conferences and workshops. He also served as an associate editor in several journals.

M. F. Mridha

MD. ANWAR HUSSEN WADUD is a lecturer in the Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka, Bangladesh. He received his B.Sc. and M.Sc. Engineering degree in CSE from Mawlana Bhashani Science and Technology University, Tangail, Bangladesh. He participated in several ACM ICPC programming contests during his university life. He worked on several programming platforms such as Java Spring Hibernate, Android apps developments, Python NumPy, Keras etc. for big data and deep learning analysis in several software companies. His area of interest is Big Data Analysis, Deep Learning, Natural Language Processing, Internet of Things and Machine Learning.
MD. ABDUL HAMID has been working as a Professor with the Department of Information Technology, King Abdul Aziz University, Jeddah, Kingdom of Saudi Arabia since 2012. He was an Assistant Professor with the Department of Computer Science and Engineering, Taibah University, Madinah, Saudi Arabia from 2013 to 2016. He was an associate professor with the Department of Computer Science, Faculty of Science and Information Technology, American International University Bangladesh, Dhaka, from 2016 to 2017. He was an associate professor and a professor with the Department of Computer Science and Engineering, University of Asia Pacific, Dhaka, from 2017 to 2019.

MUHAMMAD MOSTAFA MONOWAR (Senior Member, IEEE) is currently working as an associate professor at the Department of Information Technology at King Abdul Aziz University, Kingdom of Saudi Arabia. He also worked as a faculty member at the Department of Computer Science and Engineering at the University of Chittagong, Bangladesh. He received his Ph.D. degree in Computer Engineering in 2011 from Kyung Hee University, South Korea. He received B.Sc. degree in Computer Science and Information Technology from the Islamic University of Technology (IUT), Bangladesh, in 2003. His research interests include Wireless networks, mostly ad-hoc, sensor, and mesh networks, including routing protocols, MAC mechanisms, IP and transport layer issues, cross-layer design, QoS provisioning; security and privacy issues; and natural language processing. He has served as a program committee member in several international conferences/workshops. He served as an editor of a couple of books published by CRC press and the Taylor Francis Group. He also served as a guest editor of several journals.

M. ABDULLAH-AL-WADUD (M13) received the B.S. degree in computer science and the M.S. degree in computer science and engineering from the University of Dhaka, Bangladesh, in 2003 and 2004, respectively, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2009. He served as a Faculty Member with the Department of Industrial and Management Engineering, Hankuk University of Foreign Studies, South Korea, from 2009 to 2014. He also served as a lecturer with the Faculty of Sciences and Information Technology, Daffodil International University, Bangladesh, in 2003, and with the Faculty of Sciences and Engineering, East West University, Bangladesh, in 2004. He is currently an associate professor with the Department of Software Engineering, King Saud University, Saudi Arabia. His research interests include pattern recognition, optimization, computer vision, cloud computing, recommender systems, sensors, and ad hoc networks.

ATIF ALAMRI received the Ph.D. degree in Computer Science from the School of Information Technology and Engineering, University of Ottawa, Canada, in 2010. He is currently a full professor with the Software Engineering Department, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia. He is a founding member of the Chair of Pervasive and Mobile Computing (CPMC), CCIS, KSU, successfully managed its research program, which transformed the chair as one of the best chairs of research excellence in the college. His research interests include multimedia-assisted health systems, ambient intelligence, service-oriented architecture, multimedia cloud, sensor cloud, the Internet of Things, big data, mobile cloud, social networks, and recommender systems.

***