A Sophisticated Feature Vectorization-Based Stacked Machine Learning Approach for Fake News Detection in Bangla and English

Md. Sabbir Hossen[1], Fahim Al Farid[2], Pabon Shaha[1,3], Md. Mowahibur Rahman Twake[1], Fahjimatus Sabah[1], K. M. Mursalin Billah Rezwan[1], Anichur Rahman[3,4*], Abu Saleh Musa Miah[5], Hezerul Abdul Karim[2*]

[1]Department of Computer Science and Engineering, Bangladesh University, Mohammadpur, Dhaka-1207, Bangladesh.
[2]Faculty of Engineering, Multimedia University, Cyberjaya 63100, Malaysia.
[3]Department of Computer Science and Engineering, Mawlana Bhashani Science and Technology University, Tangail-1902, Dhaka, Bangladesh.
[4]Department of Computer Science and Engineering, National Institute of Textile Engineering and Research (NITER), Constituent Institute of the University of Dhaka, Kohinoor gate, Dhaka-Aricha Hwy, Nayarhat, Savar, Dhaka, Bangladesh.
[5]Department of Computer Science and Engineering, Bangladesh Army University of Science and Technology (BAUST), Nilphamari, Bangladesh.

*Corresponding author(s). E-mail(s): anis_cse@niter.edu.bd; hezerul@mmu.edu.my;
Contributing authors: sabbirhossen5622@gmail.com; fahmid.farid@gmail.com; pabonshahacse15@gmail.com; twake.edu@gmail.com; fahjimatus.sabah@gmail.com; mursalin.cse56@gmail.com; abusalehcse.ru@gmail.com;

Abstract

The rapid spread of fake news through social media and the internet poses a major challenge, especially in developing countries like Bangladesh. Fake news,

consisting of misleading or fabricated content, can severely impact individuals, organizations, and society. Many existing detection methods are complex and resource-intensive, making them unsuitable for real-time applications or low-resource languages. To address this issue, we proposed an efficient fake news detection system using a stacked machine learning model with TF-IDF for feature extraction. TF-IDF converts text into a numerical representation by emphasizing key terms, while the stacked ensemble model improves classification accuracy by integrating multiple machine learning algorithms. We also explored various machine learning classifiers, as well as the mBERT and Word2Vec feature vectorization techniques. We evaluated our system using three datasets: one English and two Bangla fake news detection datasets. The TF-IDF + Stacking model achieved 99.6% accuracy and a 99.8% F1-score on the English dataset. For the first Bangla dataset, accuracy improved from 74.8% to 85.2% after applying SMOTE. On the second Bangla dataset, BanFakeNews, the model achieved 98.4% accuracy, demonstrating strong performance across languages. This research introduces a lightweight yet highly effective machine learning-based fake news detection system, making it suitable for real-world applications, especially in multilingual and resource-limited settings.

Keywords: Machine Learning, Fake News, Social Media, Natural Language Processing, Feature Extraction, Ensemble Method, Deep Learning.

1 Introduction

The modern generation has grown up heavily dependent on social media, with platforms like Facebook, Instagram, TikTok, and X (Twitter) having permeated every aspect of daily life. This evolution has been driven by the tremendous development in recent years, and the widespread use of wireless technologies like Wi-Fi, 3G, 4G, and 5G networks [1], [2]. These advancements have revolutionized how people connect with the Internet, making social network platforms among the most widely utilized services globally, accessible to people whenever, wherever [3]. People create profiles on social networking platforms to connect with friends, communicate with them, and publish or tweet content, more people are exposed to false news. Although these platforms provide a wealth of opportunities for communication, amusement, and information exchange, they also pose a number of significant challenges. One major challenge is determining whether news on social media is real or fake, especially now that we are more aware of fake news [4]. Fake news is misleading information aimed at swaying public opinion or disparaging an individual. Fake news includes rumors, opinions, conspiracy theories, spam, parody news, misinformation, disinformation, Clickbait, and fake reviews, to name a few [5], [6]. Fake news has been shown to get more attention on social media than real news, with 10 fake news items on the popular social networking site "Facebook" getting more user interaction than the top 10 real news items. Social media features like posting, commenting, sharing, and tagging friends in a post have been noticed as being relatively easy for spreading these stories [7]. Therefore, it is crucial to prevent the spread of both fake news. To address this, we developed a novel solution to identify fake news at an early stage. As an example, we discussed a fake news story regarding the Padma Bridge in Bangladesh.

The Padma Bridge is a significant milestone for Bangladesh. However, when construction began, a disturbing piece of fake news spread on Facebook, claiming that building the first pillar required the severed heads of children. It was falsely reported that kidnappers had been hired to abduct children and collect their heads. One day, a mother named Renu, appearing disheveled, picked up her daughter from school, and the child cried and pleaded for something. At that moment, another guardian named Riya shouted, "Child lifter!" which led to the surrounding crowd brutally beating Renu to death in front of her daughter [8]. This story motivates us to develop new solutions for identifying false news at an early stage.

The spread of misleading information through media channels such as online newspapers and social media posts has underscored the need also to identify trustworthy news sources [9], [10]. These digital media channels have significantly influenced our daily routines and online behaviors. Consequently, we must exercise great caution when posting and sharing content on social media. Users are responsible for assessing the credibility of information and filtering out fake news, as it is often widely disseminated through digital platforms. Now, we discuss how to address the issue of false news and review the existing work on this topic. Several methods have been attempted to address this issue, with the most popular approach being machine learning combined with NLP to identify and stop the spread of fake news. Machine learning has broad applications, including computer vision, face and speech recognition, image analysis, human-computer interaction, fraud detection, robotics, cybersecurity [11], loan credit approval, and bank failure prediction, among others. These methods have shown promise as effective tools [12]. Many researchers have utilized machine learning and deep learning in different ways to identify fake news. For example, [13] proposed a hybrid technique combining a curriculum strategy with KNN to enhance deep learning performance, suggesting a two-phase model using NLP and ML algorithms to extract new structural features from news samples. [14] applied DL methods such as LSTM, DNN, and CNN for fake news detection. Other works [15, 16, 17, 18, 19] used LSTM, Word2Vec, and TF-IDF with ML and DL algorithms for false news identification, while [20] leveraged BERT and NLP along with ML to identify fake news. Further studies have explored advanced deep learning architectures, such as the use of RoBERT in [21] for fake news detection, and the application of multi-head attention dual summarization in [22] to detect incongruent news articles. The aforementioned research has focused on identifying fake news using various techniques, yet some limitations need further attention. To tackle these challenges, we proposed a machine learning approach integrated with natural language processing for detecting fake news. This research examines how ML can effectively identify false news. The key contributions of our study are outlined below:

- We proposed an enhanced feature vectorization-based model for detecting fake news in both Bangla and English. This study employs Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction. Additionally, the Synthetic Minority Over-Sampling Technique (SMOTE) is applied to balance the Bangla dataset.
- We applied several machine learning algorithms, including SVM, DT, RF, LR, KNN, GNB, XGB, and LightXGB, along with AdaBoost, Voting Classifier, and the Stacked model. The Stacked model, an ensemble learning technique, improves classification accuracy by combining the strengths of multiple algorithms. In this model, we used five distinct machine learning algorithms: SVM, RF, LR, GB, and XGBoost.

- The Stacked model is our proposed approach, and its performance is evaluated using Accuracy, Precision, Recall, F1-score, and the ROC-AUC Curve. The source code is available in the below link: https://github.com/PabonSC/Fake-News-Detection

The following headings outline the remaining sections of our study: **Section 2 (Literature Reviews):** This chapter summarizes the research papers and surveys currently available on recognizing fake news using techniques. **Section 3 (Methodology):** This section explains the required specifications, associated diagrams, and system architecture of our approach. **Section 4 (Results and Discussion):** In this chapter, we detail the machine learning models used, supported by information, tables, graphs, charts, and visuals of the tools employed. **Section 5 (Conclusion):** This section summarizes the key findings to conclude the study and offers insights into potential future applications of the research.

2 Literature Review

Fake news is a common problem worldwide. Several studies, including the use of ML and DL, have been conducted to eliminate or stop the spread of false news. This section mentions a few of them that have been proposed before to detect fake news.

2.1 Fake News Detection Techniques

There are both conventional and contemporary methods for detecting fake news. Rule-based systems evaluate language patterns, like keyword frequency or sentence structure, to detect potentially erroneous information without the use of ML or DL. Cross-referencing claims with credible sources and fact-checking websites are other popular techniques. According to their author or place of origin, suspicious articles can be identified using methods such as content-based filtering and metadata analysis. When contrasted with ML and DL-based techniques, these strategies are less flexible and scalable.

2.2 Fake News Detection through NLP and ML

To detect fake news, *V. Asha et al.* [23] utilized machine learning algorithms RF, NB, and LR to detect various types of false information and achieve accurate news classifications. Among these, the RF algorithm demonstrated superior performance, achieving an experimental accuracy of 99.06%, surpassing both Logistic Regression and Naive Bayes. The Random Forest approach consistently produced better results. *L. Holla et al.* [24] presented an improved model to identify fake news that combines the Iterative Dichotomiser 3 (ID-3), NB, and RF classifiers with the Adaptive boosting ensemble classifier. The proposed Hybrid TF-IDF was utilized to extract features. The AdaBoost classifier classifies the news as true or false, and the characteristics are chosen using the Least Absolute Shrinkage and Selection Operator (LASSO). The findings demonstrated an increased classification accuracy of 98.98% for TF-IDF using the AdaBoost ensemble classifier. However, the proposed model was not designed to identify false news based on real-time applications. *D. Choudhury et al.* [25] compared the effectiveness of SVM, NB, RF, and LR classifiers for identifying false news across several datasets. Fake News datasets, Fake Job Posting, The Liar showed that the SVM reached the greatest accuracy at 96%, 97%, and 61%, respectively. Again, their innovative GA-based false news detection system considers the fitness functions of SVM, NB, RF, and LR. The LIAR dataset yielded 61% accuracy for both the SVM and LR

classifiers in their suggested approach, while the fake job posting dataset produced the highest accuracy of 97% for SVM and RF. *Altheneyan et al.* [26] used machine learning and Big data technology (Spark) to assess and compare the most sophisticated techniques for identifying fake news. This study's methodology used a decentralized Spark cluster to build stacked ensemble models. After extracting features with a count vectorizer, TF-IDF hashing, and N-grams, the authors applied the suggested stacked ensemble classification model. According to the findings, the Hasing TF-IDF ensemble achieved 93.45% of the highest accuracy. Also, the suggested approach performs better at classifying data than the baseline strategy, with an F1 score of 92.45% compared to 83.10%. *M.K. Jain et al.* [27] introduced a brand-new "ConFake" algorithm. This algorithm includes eighty content-based features to help users to spot fake news. The project made use of word vectors and content-based attributes that were taken directly from the text of news articles. Combining these traits, they were fed into machine learning classifiers. They conducted all of the tests on five publicly accessible datasets and one artificially created ConFake dataset that combines multiple datasets, including BuzzFeed, PolitiFact, Reuters, Kaggle, and McIntire, to validate the experimental results. Comparing the suggested model to other cutting-edge models, it obtained a maximum accuracy of 97.31%. However, finding false news on social media early on is not a suitable use of this model, as the text field of news stories is the only aspect the model examines. *J. Asha et al.* [28] utilized n-gram analysis and ML methods to identify false news. This study used Six machine learning techniques: KNN, DT, SVM, LSVM, LR, and stochastic gradient descent. With a 93.5% accuracy rate, the study reveals that the highest result is achieved while using the feature extraction approach TF-IDF, together with LSVM and Stochastic Gradient Descent algorithms. Performance tuning is used in Stochastic Gradient Descent to improve accuracy. When Random Search CV and Grid Search CV are compared, the latter provides more accuracy 94.2%.

Mugdha et al. [29] proposed an approach that can precisely ascertain whether a story is real or not based on the headline of the news. The authors used the Gaussian NB to produce a unique dataset of the Bengali language, achieving their goal and reaching their target. Although they have experimented with several Machine Learning methods, their model has shown that the Gaussian NB Algorithm worked well. To choose the attribute, this technique used an Extra Tree Classifier in conjunction with utilizing a text feature that was dependent on TF-IDF. They achieved 87% accuracy using Gaussian NB, which was superior to any other technique they employed. *Hussain et al.* [30] presented experimental research to determine false news on Bangla social media. They used two supervised ML techniques, SVM and Multinomial NB, to identify false news in Bangla. Count Vectorizer and TF-IDF were employed as feature vectorization methods. Their proposed method identifies false news based on the connected post's polarity. Ultimately, their research indicated that SVM with a linear kernel outperforms Multinomial Naïve Bayes with a 93.32% accuracy, yielding 96.64% accuracy. *Coelho et al.* [31] utilized machine learning algorithms to identify false news in Malayalam languages. TF-IDF of word unigrams is used to train three distinct models to detect false news in code-mixed Malayalam language. These models are Multinomial Naive Bayes, LR, and an Ensemble method (SVM, LR, and MNB) with hard voting. The ensemble model performed the best out of the three, with a macro accuracy of 83.1%. *Farooq et al.*[32] suggested a methodology for identifying false news in Urdu. A bag of words containing n-gram combinations and the TF-IDF is used in the experiments. The authors used feature stacking, a technique that mixes verbs extracted from the text with the feature vectors of preprocessed text. Bagging

was done using SVM, KNN, and ensemble models such as RF and ExtraTreeClassifier, while stacking was done using RF and ET as base learners and LR as the meta learner. Stacking yields the maximum accuracy of 93.82%, according to experimental data.

2.3 Fake News Detection through NLP and DL

S.V. Balshetwar et al. [33] suggested a fake news identification model that uses Multiple Imputation Chain Equation (MICE) with multiple imputations to handle multivariate missing data in news or social media datasets. TF-IDF was employed to extract long-term features from the text. The classification was performed using passive-aggressive, Deep Neural Network (DNN), and NB classifiers, achieving a 99.8% accuracy rate in identifying false news *A. Pal et al.* [34] introduced a model that aids in identifying false news while dealing with data gathered from the internet. These articles served as their training dataset, and they compared the outcomes using a variety of ML algorithms, including NB, LR, DT, LSTM, and BERT. In their experiment, they were able to identify false news with 99.6% accuracy using the DT and 99.8% recall using the LSTM. *J. Alghamdi et al.* [35] introduced a model to detect false news related to COVID-19; this study investigated the efficacy of several ML method and the refinement of pre-trained transformer-based models, such as COVID-Twitter-BERT (CT-BERT) and BERT. The authors assess the effectiveness of several downstream neural network architectures, with frozen or unfrozen parameters, added to BERT and CT-BERT, such as BiGRU and CNN layers. With an accuracy of 98.46%, the COVID-19 false news dataset shows that adding BiGRU to the CT-BERT model produces exceptional results. *D.A. Salh et al.*[36] suggested a model to identify false news in Kurdish language. Word embedding, TF-IDF, and count vector were utilized to extract features from news texts. RF, SVM, and other ML and DL classifiers were used to identify the false news. The findings demonstrated that it was possible to spot fake news that contained text, particularly when convolutional neural networks were applied. The experimental findings of the study demonstrate that CNN outperforms the other methods, with an accuracy of more than 91%. *M. Azzeh et al.*[37] constructs a large annotated Arabic fake news corpus and develops machine learning models using Arabic-specific language models (ARBERT, AraBERT, CAMeLBERT, and AraVec), finding that deep learning classifiers with transformers, especially CAMeLBERT, outperform traditional machine learning approaches. How ever detecting fake news in Arabic presents unique challenges, including a lack of annotated data, dialectal variations, morphological complexity, semantic ambiguity, and cultural context. *M. Khalil et al.* [38] introduces Pegasos Quantum Support Vector Machines (PegasosQSVM), combining Pegasos SVM with quantum kernels and advanced data encoding, achieving up to 95.63% accuracy in local simulations and outperforming other ML models on the BUZZFEED dataset. While the approach shows promise, implementation on real Quantum Processing Units (QPUs) faces challenges due to noise effects, necessitating further research on error mitigation techniques for optimal performance. Table 1 shows the summary of existing works on Fake News Detection.

In summary, researchers have frequently employed various machine learning (ML) and deep learning (DL) algorithms alongside natural language processing (NLP) techniques to detect fake news in English, Arabic, and other languages. However, limited research has been conducted on the low-resource language Bangla. Moreover, existing approaches suffer from several key limitations: researchers focus on single-language

Table 1 Summary Table for Related Works on Fake News Detection

Ref.	Algorithms/Technique	Major Findings	Backdrop
[23]	RF, NB, and LR	Random Forest achieved 99.06% highest accuracy	No feature extraction technique used
[24]	ID-3, NB, RF classifiers with Adaptive Boosting	AdaBoost achieved 99.06% highest accuracy	Not designed for real-time application
[25]	SVM, NB, RF, and LR	SVM achieved 97% highest accuracy	Not suitable for fake news detection from social media
[26]	RF, DT, LR, Ensemble	HashingTF-IDF_Ensembler achieved 93.45%	Accuracy is comparatively lower than others
[27]	SVM, RF, KNN, NB, LR, Bagging, Boosting	Random Forest achieved 97.31% accuracy	Model examines only the text of news articles
[28]	DT, LR, KNN, SVM, LSLVM, SGD	Linear SVM achieved 93.5% accuracy	Not suitable for social media fake news detection
[29]	Gaussian NB, SVM, LR, RF, Multinomial NB, AdaBoost, GB, Voting Classifier	Gaussian NB achieved 87% accuracy	Accuracy is quite lower than others

fake news detection without experimentation in other languages, many are unsuitable for detecting fake news on social media, some exhibit lower accuracy, others lack feature extraction techniques, and several are not designed for real-time applications. To address these challenges, we introduced a novel machine learning model for multilingual fake news detection, leveraging advanced feature extraction techniques. Our proposed model combines feature extraction methods, including TF-IDF, mBERT, and Word2Vec, with eleven traditional and state-of-the-art classifiers to enhance classification performance. Designed to work across multiple languages, particularly low-resource languages like Bangla. The model was first tested on an English fake news detection dataset, achieving higher accuracy than previous studies. Additionally, we applied the model to Bangla fake news detection on a dataset with no prior research and obtained satisfactory accuracy rates, demonstrating its effectiveness across diverse linguistic contexts. We further evaluated the model on a second Bangla dataset, BanFakeNews, where it continued to perform strongly, confirming its robustness across multiple Bangla datasets.

3 Materials and Methods

In this study, we aimed to train a model that can accurately distinguish between true and fake news, with the goal of benefiting our community. By detecting fake news early, we sought to limit its dissemination and prevent further spread. Fig. 1 presents the proposed fake news identification model. This section outlines the step-by-step methodology used to develop the model. The process began by gathering datasets. After collecting the data, preprocessing steps were performed to prepare it for machine learning classifiers and feature extraction. Data preprocessing included handling null values, tokenization, stemming, lemmatization, and stop-word removal. Additionally, we employed SMOTE to balance the Bangla dataset (dataset-2). mBERT, Word2Vec, and TF-IDF vectorizer were utilized for feature extraction. After preprocessing, the extracted features were fed into various classifiers to predict fake news. The classifiers used included SVM, RF, DT, LR, GB, NB, XGB, AdaBoost, LightGBM, Voting, and Stacking models. The models' performance was assessed after training. using confusion matrices, and performance metrics such as accuracy, recall, F1-score, and precision were utilized. The procedures and models employed in this study are detailed in the following sections. However, certain procedures were excluded due to the inclusion

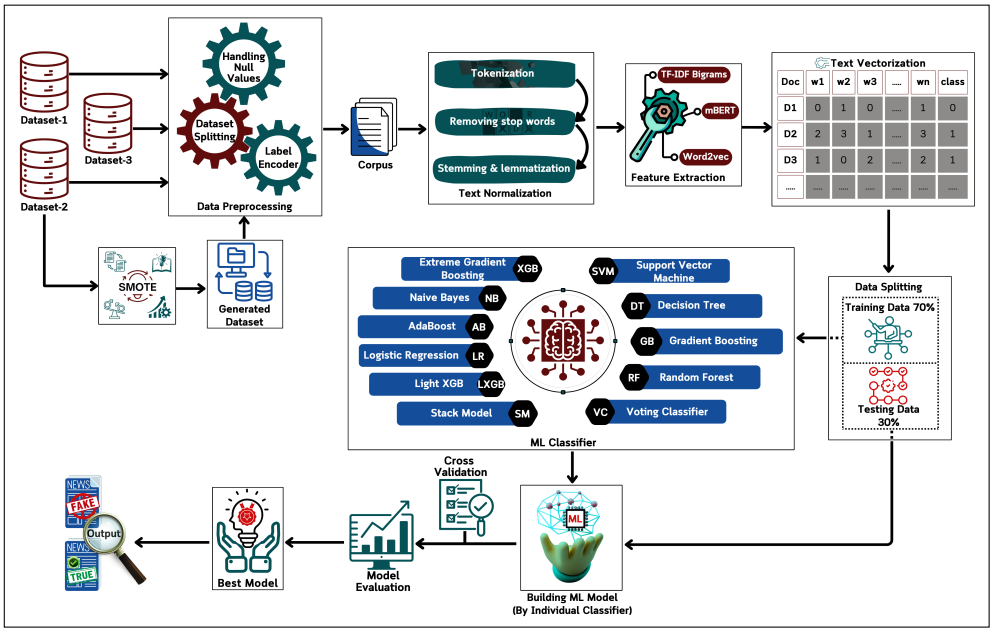


Fig. 1 The overall framework for the fake news detection system.

of the Bangla language alongside English, as some tools required for Bangla were unavailable.

3.1 Dataset Description

We used three datasets to distinguish between true and false news. Two of them were sourced from Kaggle: the Bangla and English Fake News Detection Datasets, comprising two separate CSV files *final_en.csv* for English fake news and *final_bn_data.csv* for Bangla fake news. The third dataset, titled **Banfakenews**, was collected from a published research article [39]. Together, these datasets provide a total of 34,353 data points. For the purposes of this study, we refer to the English Kaggle dataset as **Dataset-1**, the Bangla Kaggle dataset as **Dataset-2**, and the Banfakenews dataset as **Dataset-3**. Table 2 provides an overview of the dataset's properties used in the study.

Table 2 Dataset Description

Data set No.	Total Data	Real News	False News
Dataset 1	10000	5000	5000
Dataset 2	14537	10000	4537
Dataset 3	9816	7212	2604

3.1.1 *English Dataset*

English is one of the most extensively used languages in the world, and it is used to write data in the English dataset. The English dataset is structured in CSV format, utilizing a tabular layout of rows and columns. It contains 10,000 entries and 3 features, where each row represents a data point. The dataset is evenly balanced, with 5,000 entries classified as real (1) and 5,000 entries classified as false (0).

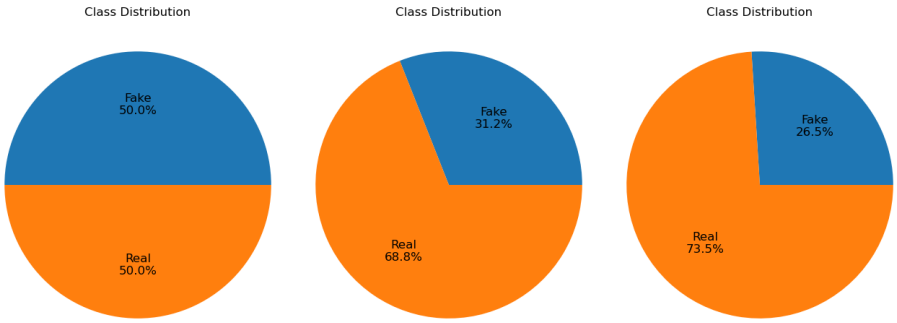
3.1.2 *Bangla Dataset*

The Bangla dataset consists of information written in Bengali, which is primarily spoken in Bangladesh and some regions of India. It is in CSV file format, a popular row-and-column tabular data format. The dataset comprises data or samples that have been determined as real (1) or false (0). The Bangla dataset has 14537 records and 4 features. In this case, 4537 news outlets are false, and 10,000 news outlets are real.

3.1.3 BanFakeNews Dataset

The Banfakenews dataset comprises news content written in the Bangla language, aimed at detecting misinformation in Bangladeshi media sources. It is provided in CSV file format, organized in a structured tabular form. The dataset includes a total of 9,816 entries, where each row represents a labeled news article. Of the total samples, 7,212 entries are labeled as real (1), while 2,604 entries are labeled as fake (0) news.

To visualize the data distribution, we employed pie charts. Figure 2 illustrates the distribution of real and fake news categories for the label classes.



(a) Dataset-1 (b) Dataset-2 (c) Dataset-3

Fig. 2 Class distribution of three datasets, Visualizing using Pie charts

3.2 Data Pre-processing Strategies

The dataset, being in natural language, inherently contains a lot of noise. To prepare the data for use with algorithms, it undergoes several preprocessing steps. Data preprocessing is the process of repeatedly converting raw, unprocessed data into a more useful and understandable format. Raw datasets often have issues such as errors,

missing values, inconsistencies, and lack of clear patterns or behavior. Preprocessing is crucial for addressing these problems by handling missing values and reconciling inconsistencies. To improve efficiency, preprocessing modifies the data before it is processed by algorithms, ensuring it is in a suitable form for further analysis.

3.2.1 Balancing the dataset using the Synthetic Minority Oversampling Technique (SMOTE)

The Bangla dataset originally consists of 14,537 data points, with 10,000 real news items and 4,537 fake news items, resulting in a class imbalance. To address this, we applied the Synthetic Minority Over-sampling Technique (SMOTE), which interpolates between samples of the minority class and their nearest neighbors to generate synthetic data points. By adding 5,463 synthetic samples to the fake news category, the dataset was expanded to 20,000 samples, achieving a balanced distribution of 10,000 true and 10,000 fake news. This process reduced bias towards the dominant class and improved model performance by providing a more equitable representation of both classes.

3.2.2 Handling Null Values

Handling null values is essential for maintaining data quality and improving model performance. Missing values can be addressed through deletion or imputation using methods like mean, median, or mode. Proper handling reduces errors and enhances model accuracy, ensuring a consistent and reliable dataset for training. For text data, missing values can be replaced with an empty string or imputed based on context. However, in the fake news dataset, no null values were found.

3.2.3 Label Encoding

Label Encoding is a method used to convert categorical labels into numerical values. It helps machine learning models process categorical data efficiently. In the fake news dataset with two classes (Real & False), Label Encoding assigns **0 to Real** and **1 to False**. This transformation ensures the model can interpret and analyze the labels effectively. It also preserves the categorical relationships within the dataset.

3.2.4 Splitting the Dataset

To split the dataset, the K-Fold cross-validation technique was used in our proposed method. K-Fold Cross-Validation aims to divide the dataset into k segments. Each iteration uses $(k-1)$ segments for training and the K^{th} segment for testing. This ensures that the entire dataset is used for both training and testing. Cross-validation has been employed to assess our performance. Here, $k = 5$ was employed for the procedure. We receive 30% of the data for testing and 70% for training in each cycle due to splitting the dataset into 5 folds. The dataset is labeled as (1) and (0), where (1) represents real news and (0) represents fake news. The system distinguishes between real and false, determines whether the news is accurate or inaccurate, and computes the accuracy of the models as a percentage based on the ratio of correct to incorrect predictions.

3.3 Text Normalization

Text normalization is a necessary step in natural language processing (NLP). To make text data consistent and useful for various NLP tasks, it must be cleaned and preprocessed. Many approaches were used in this procedure, including,

3.3.1 Tokenization

The method of encoding every word as a number is called tokenization. Within the fields of Machine Learning and NLP, the tokenization procedure for a text sequence entails partitioning it into smaller pieces called tokens. The length of these tokens might vary from word to character. Tokenization is the process of parsing text to eliminate particular words to use for predictive modeling.

3.3.2 Stop Words

The words ‘a,’ ‘an,’ ‘the,’ ‘are,’ and ‘is’ are a few instances. In NLP and text mining, stop words are frequently used to filter out overused words that don’t provide any valuable information. In this case, the stop word dictionary is provided by NLTK. First, every stop word in the text is eliminated to tidy it up. Stop words can be eliminated from the text because they are less informative and more common. Stop words that are frequently used are the conjunctions "and," "or," and "but." Because processing these less common whole words takes a long time, pre-processing data is crucial to natural language processing.

3.3.3 Stemming and Lemmatization

This phase involves either stemming or lemmatization. For lemmatization, the NLTK uses its WordNet Lemmatizer; for stemming, it uses its Snowball Stemmer implementation, which is based on the Porter2 stemming method.

- *Stemming:* To simplify an inflected word to its word stem, employ the stemming process. The common word stem "Detect" can be used to represent the terms "Detection," "Detector," and "Detects," for instance. In other words, the three inflection words before "Detect" can be substituted with "Detect."
- *Lemmatization* Another method for returning inflected words to their source word is lemmatization. It explains the algorithmic procedure for determining the dictionary form, or "lemma," of an inflected word by considering its intended meaning. To lemmatize a document usually means "doing things correctly." A lemmatization algorithm, for instance, should convert the terms "identify," "identifying," and "identification" into the lemma "identify."

3.4 Feature Extraction

The technique of feature extraction converts unprocessed data into numerical characteristics that may be used for additional processing while maintaining the information contained in the original data set. When compared to just training a machine with raw data, it is more efficient. Word2Vec, mBERT, and TF-IDF are the three categories of methods for feature extraction techniques we utilized in the proposed model.

3.4.1 TF-IDF

TF-IDF or Term Frequency-Inverse Document Frequency, is a machine-learning metric that quantifies the significance or pertinence of string representations (words, phrases, lemmas, etc.). Term Frequency or TF is a tool for determining a term's frequency of occurrence in a document. Given that all documents have different sizes, A term can occur more often in a longer text than in a shorter one. Frequently, TF is divided by the length of the document. TF is used for text summarization, search engine optimization, and document clustering [40].

$$\mathrm{TF}(t, d) = \frac{\text{Number of times term } t \text{ occurs in document } d}{\text{Total word count of document } d} \tag{1}$$

IDF, or Inverse Document Frequency, states that a word has little value if it appears in every document. A text may contain numerous instances of words that are not very important, such as "a," "an," "the," "on," and "of." IDF gives uncommon terms more weight while lowering the value of familiar terms. The word's IDF value increases its uniqueness [41].

$$\mathrm{IDF}(t, d) = \frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}} \tag{2}$$

When TF-IDF is utilized in the text body, each sentence's relative count is noted in the document matrix. Sparse matrices are produced using vectorizers. A sparse matrix is one in which the majority of the elements are 0. The following is the equation for TF-IDF.

$$\text{TF-IDF}(t, d) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t) \tag{3}$$

3.4.2 Word2Vec

The Word2Vec model functions based on the core idea that a word's meaning is captured by its context. This model, in short, creates each word as a multidimensional vector representation. At the same time, it maintains the syntactic and semantic connections between them. Word associations as seen by humans are reflected in the vector distances between individual words [42]. The two main designs of the Word2Vec model are Continuous Bag-of-Words (CBOW) and Skip-gram. Whereas, within a specific window, CBOW forecasts a target word determined by the enclosing context words [43]. The goal of skip-gram is to anticipate context words within a particular context window, given a target word. These structures can be thought of as simple neural networks having input, hidden, and output layers. The hidden layer's weights, which function as word vectors, are the training results of these networks rather than the output layer itself. To minimize computational complexity, Word2Vec uses algorithmic improvements, including hierarchical SoftMax and negative sampling, in addition to using a SoftMax activation operation for the output layer [44].

3.4.3 mBERT

Multilingual BERT (mBERT) is a transformer-based language representation model that provides contextual embeddings for over 100 languages, including low-resource languages like Bangla. Unlike traditional word vectorization techniques such as TF-IDF or Word2Vec, which are based on frequency or co-occurrence statistics, mBERT leverages deep bidirectional transformers to understand semantic meaning in context. It was pretrained on a large multilingual corpus using masked language modeling

and next sentence prediction objectives, allowing it to learn syntactic and semantic relationships across different languages [45].

mBERT processes each sentence by first converting it into tokens and then generating dense vector representations through its multiple attention-based layers. Typically, the contextualized embedding of the special classification token [CLS] is extracted and used to represent the entire sentence in a fixed-size vector of 768 dimensions. Alternatively, the embeddings of all tokens can be averaged or pooled for sentence-level representation. This contextual embedding captures both local and global linguistic information and is robust to polysemy and word order variations [46]. In this study, mBERT was employed to generate dense contextual embeddings for each input text. These embeddings, which capture semantic and syntactic information across languages, were used as feature vectors for downstream classification. The resulting vectors were then fed into eleven different machine learning classifiers to evaluate their performance on the classification task.

3.5 Data Visualization

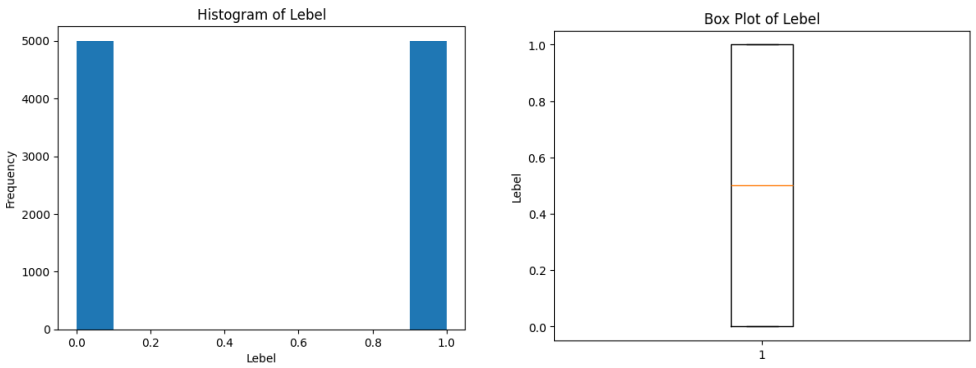
Information and data are represented graphically in data visualization. It uses visual elements such as charts, graphs, maps, and other tools to present data, making complex datasets more straightforward to understand, analyze, and interpret. In this research, we used the following techniques.

3.5.1 Histogram and Box Plot

A histogram is a graphical representation of the distribution of data. It divides the range of data into intervals, called bins, and counts how many data points fall into each bin. This helps visualize the frequency distribution of the dataset.

A box plot summarizes the distribution of a dataset by displaying its key descriptive statistics, including median, quartiles, and outliers.

We present two data visualization approaches for the original dataset. Fig. 3 (a) shows the histogram, and (b) depicts the box plot for Dataset-1 before applying TF-IDF.

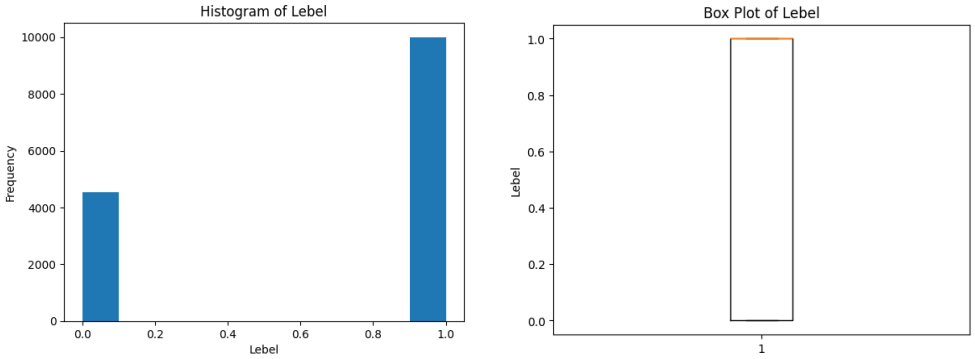


(a) Histogram (b) Box Plot

Fig. 3 Histogram and Box Plot representation for Dataset-1

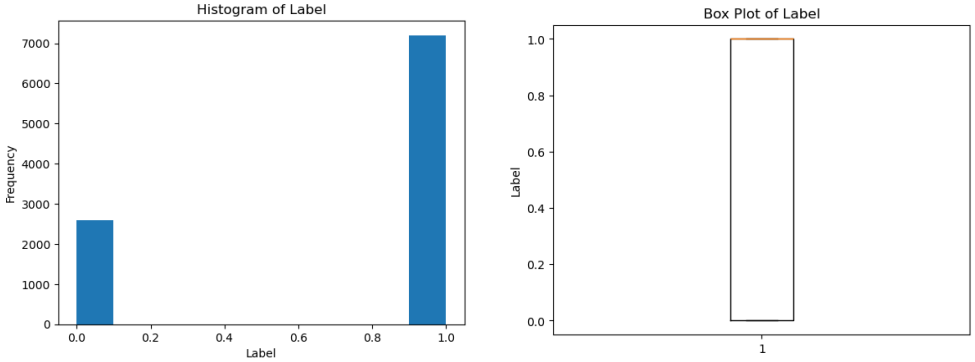
Fig. 4 (a) presents the histogram, while (b) illustrates the box plot for Dataset-2 before applying TF-IDF. The histogram indicates that Dataset-2 is imbalanced. To address this, we applied SMOTE to balance both classes in Dataset-2.

Fig. 5 (a) shows the histogram, and (b) depicts the box plot for Dataset-3 before applying TF-IDF.



(a) Histogram (b) Box Plot

Fig. 4 Histogram and Box Plot representation for Dataset-2



(a) Histogram (b) Box Plot

Fig. 5 Histogram and Box Plot representation for Dataset-3

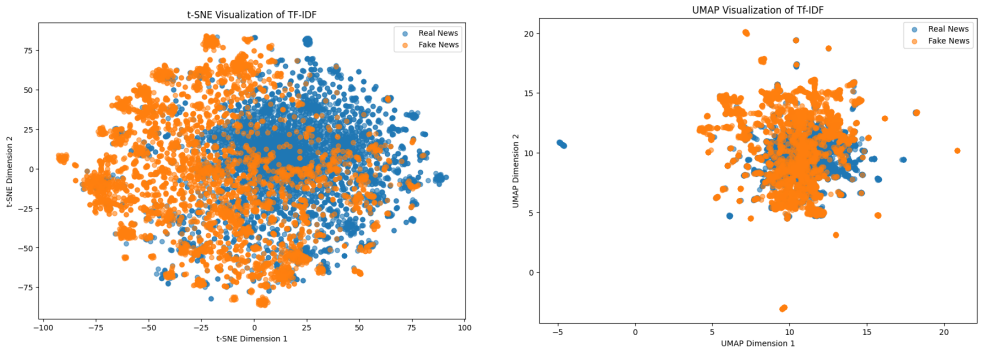
3.5.2 UMAP and t-SNE

We applied two feature vectorization techniques, TF-IDF and Word2Vec, to both datasets. TF-IDF consistently demonstrated superior performance. Therefore, we present t-SNE and UMAP data visualizations applied exclusively to the TF-IDF vectorizer.

t-Distributed Stochastic Neighbor Embedding, or t-SNE, is a dimensionality reduction technique that preserves local structure while visualizing high-dimensional data by mapping it onto a low-dimensional space. Although it is frequently used for data clustering and pattern recognition, it can be computationally demanding for large datasets.

An alternative to t-SNE that is quicker and preserves both local and global structures while reducing dimensionality is UMAP (Uniform Manifold Approximation and Projection). It can effectively visualize complex data distributions and is scalable.

Fig. 6 (a) shows the t-SNE visualization, while (b) presents the UMAP visualization of Dataset-1 after applying TF-IDF.

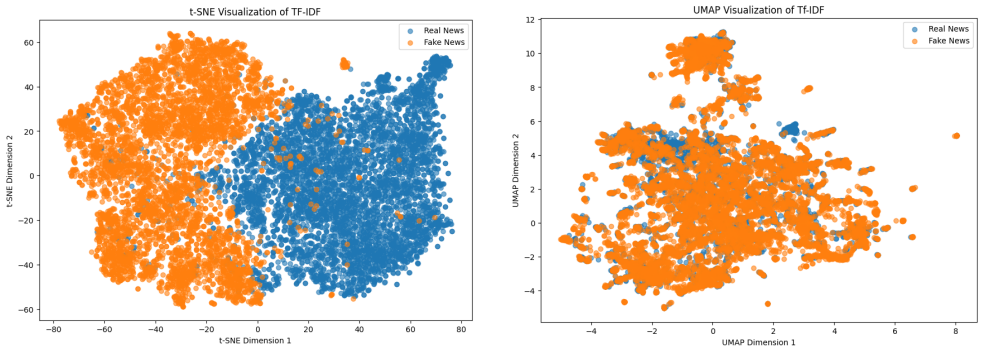


(a) t-SNE (b) UMAP

Fig. 6 t-SNE and UMAP for Dataset-1 after apply TF-IDF Vectorization.

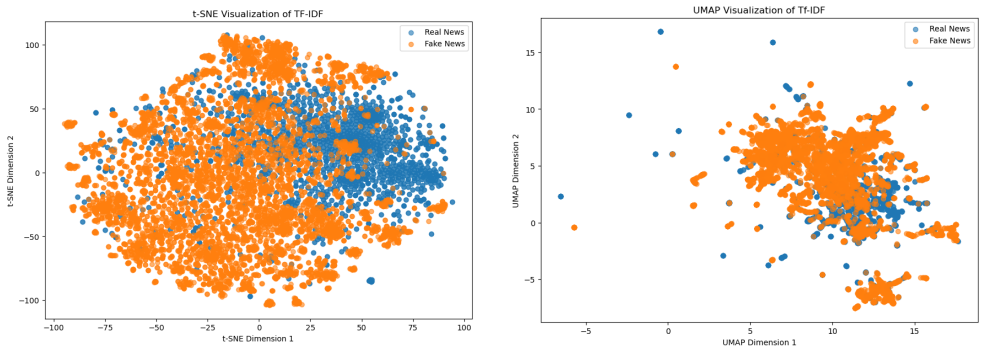
Fig. 7 (a) illustrates the t-SNE visualization, while (b) depicts the UMAP visualization of Dataset-2 after applying TF-IDF.

Fig. 8 (a) displays the t-SNE visualization, and (b) presents the UMAP visualization of Dataset-3 after applying TF-IDF.



(a) t-SNE (b) UMAP

Fig. 7 t-SNE and UMAP for Dataset-2 after apply TF-IDF Vectorization.



(a) t-SNE (b) UMAP

Fig. 8 t-SNE and UMAP for Dataset-3 after apply TF-IDF Vectorization.

3.6 Analysis of Classification Procedures

The multifaceted structure of fake news makes it difficult to identify which genre of news it belongs to. It is clear that to address the problem precisely, a practical technique needs to incorporate several views. For this reason, several Machine Learning algorithms were employed as classifiers, such as SVM, RF, DT, LR, GB, NB, XGB, AdaBoost, LightGBM, Voting, and Stacking models. Details of these models are as follows.

3.6.1 Support Vector Machine - SVM

SVM is a supervised learning technique effective for both regression and classification tasks, particularly binary classification. It uses labeled data to learn and determine the optimal hyper-plane that separates the dataset into two classes: true and false [47]. The decision boundaries, known as hyperplanes, help classify the data points. Fig. 9 illustrates how SVM uses a hyper-plane for data point classification.

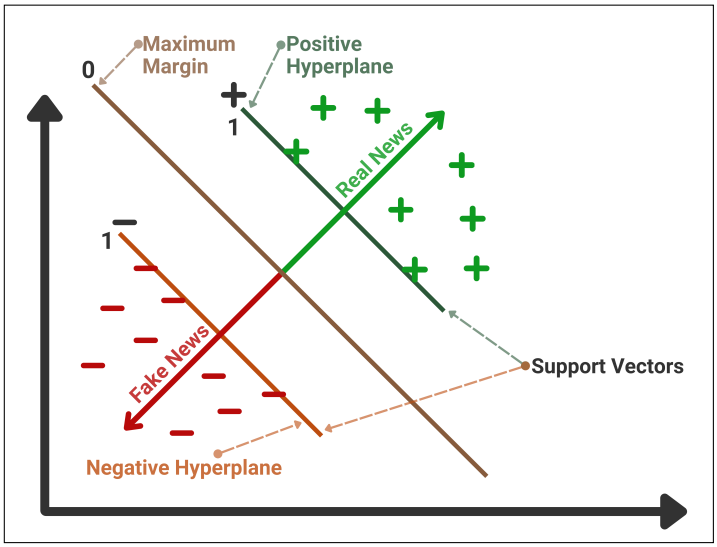


Fig. 9 Hyperplane Illustration for Support Vector Machine

In this study, the SVM algorithm generates hyper-planes based on feature vectors obtained through TF-IDF, Word2Vec,and mBERT, which capture the importance and semantic relationships of words. These vectors are used to construct hyperplanes that optimally separate the data into two classes: fake and real news. The SVM then classifies new articles by determining on which side of the hyperplane their feature vectors fall, accurately predicting whether the news is fake or real.

3.6.2 Random Forest - RF

One kind of supervised learning technique that works well for both regression and classification is called RF [48]. The RF is utilized in this study to identify whether the news is genuine or fake. Additionally, an RF is made up of decision trees on a sample of data, and the best solution is chosen by voting. The overfitting issues can be resolved, and the RF classifier improves detection accuracy [49], [50]. The following is a mathematical expression of the procedure used to train the data in RF found in Equation (4):

$$D = \left\{(f_i, C_i) \mid f_i \in \mathbb{R}^F, C_i \in \{1, 2, \ldots, c\}\right\}_{i=1}^{N} \tag{4}$$

Where N denotes the number of samples used for the training, f_i represents the features, and C_i represents the count of samples used for training. Each Decision Tree (DT) in the Random Forest (RF) classifier consists of a bag of samples (i.e., $D_1, D_2, \ldots, D_P$, etc.). The following formula, Equation (5), is used to determine the detection accuracy:

$$\text{D.A} = \frac{\text{Number of Correctly Classified Instances}}{\text{Total Number of Instances}} \times 100 \tag{5}$$

"D.A., which stands for Detection Accuracy."

$$\hat{C} = \text{majority}\left\{\hat{C}^d\right\}_1^d \tag{6}$$

In this study, numerous decision trees are generated based on feature vectors. A majority vote from these decision trees determines the final classification of the news as fake or real.

3.6.3 Decision Tree - DT

DT is a popular ML technique and a non-parametric supervised learning approach in both regression and classification. Using resource costs, chance events, and utility, a decision tree is a decision assistance tool that displays options and their potential results in a hierarchical model [51]. It has an internal, leaf, branches, and a root node, arranged like a hierarchical tree. Decision trees are used to construct models for tasks involving regression and classification, and are simple to comprehend and interpret, making them ideal for visual aids [52]. In this research, we employed the Decision Tree algorithm by following the steps:

- Decide on the desired attribute.
- Determine the Information Gain (I.G.) for the desired attribute.
- Utilizing the following formula to calculate the entropy of the remaining attributes:

$$\text{Entropy}(s) = \sum_{i=1}^{n} -P_i \log_2 P_i \tag{7}$$

- The Gain (G) of each attribute is calculated by deducting the entropy(s) from the Information Gain (I.G).

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{i=1}^{n} \left(\frac{S_v}{S} \times \text{Entropy}(S_v) \right) \tag{8}$$

Feature vector data points from TF-IDF, Word2Vecand mBERT are used to build the decision tree. Important textual data, like word frequency (TF-IDF), semantic relationships (Word2Vec), contextual and multilingual semantics (mBERT), are encapsulated in these feature vectors and provide the basis for the construction of decision trees. In a decision tree, each internal node denotes a "decision" or test on a feature (such as the presence or weight of a particular term), each branch denotes the test's result, and each leaf node denotes a class label (such as fake or real news). The model is similar to a flowchart.

3.6.4 Logistic Regression - LR

Logistic regression is a core classification algorithm that predicts the likelihood of a binary outcome using one or more input variables. [53]. Unlike linear regression, it applies the logistic (sigmoid) function to map predictions to probabilities between 0 and 1, making it suitable for binary classification problems [54]. The algorithm estimates parameters using maximum likelihood estimation and outputs the likelihood of the target class. Its simplicity, efficiency, and ability to provide interpretable coefficients make logistic regression a go-to choice for tasks such as medical diagnosis, risk assessment, and customer churn prediction [55]. It estimates the probability of a sample being part of a specific class by utilizing the logistic function:

$$P(y = 1|X) = \frac{1}{1 + e^{-(w^T X + b)}} \tag{9}$$

where w represents the weight vector, X denotes the input feature vector, and b signifies the bias term.. The decision rule for classification is defined as:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|X) \geq 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

3.6.5 Naive Bayes - NB

This classifier has been employed since it is most appropriate for contextual data and the dataset used, Based on the Bayes theorem's basic idea. NB is the kind of classifier that uses the supervised learning algorithm [56]. For the ML models, it generates rapid predictions. Text classification is the optimal use for it. The Naive Bayes Classifier is employed in binary and multi-class classifications [57]. The Naive Bayes model comes in three varieties: Gaussian NB, Multinomial NB, and Binomial NB. Gaussian NB refers to a normal distribution. Multinomial NB is mostly utilized for text classification duty, where word frequency is utilized to predict results. Similar to multinomials, binomial classifiers are employed in classification tasks [58]. We have implemented the Gaussian Naive Bayes classifier in our model.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{11}$$

P(A|B): The probability that event A will occur in a way that already causes event B to occur.

Naive Bayes would approach the classification by calculating the probability that a news item is false or true based on the feature vectors. Unlike decision trees that rely on majority voting, Naive Bayes applies Bayes' theorem, assuming independence between features, to determine the likelihood of each class (fake or true) given the input features. The model then classifies the news article as fake or real based on which class has the higher probability.

3.7 Ensemble Methods

An Ensemble Method is a strategy that enhances the model's overall performance and generalization by integrating the outputs from multiple distinct models. The fundamental notion behind ensemble methods is that better overall predictions can be obtained by compensating for the flaws of individual models by combining the predictions of numerous models [59] [60]. Two ensemble methods that we employed in our model are explained below.

3.7.1 Gradient Boosting GB

Gradient boosting in machine learning is applied to regression and classification. It's an approach to boosting. It is an ensemble machine learning strategy that sequentially aggregates the predictions from several weak learners [61]. This technique's main idea is to build models one after the other, to minimize the errors made in the prior models. However, what is the best way to accomplish that? How can we lower the error? This is accomplished by creating a new model based on the residuals or errors of the old model [62], [63]. When the goal column is continuous, the GB Regressor is used; when the problem is one of classification, the GB Classifier is employed. Only one thing separates the two: the "Loss function." In this case, by adding weak learners using

gradient descent, reducing this loss function is the objective. We will have various loss functions for classification issues (log-likelihood, for example) and regression problems (mean squared error, or MSE) because it is based on a loss function.

$$\text{Probability} = \frac{e^{\log(\text{odd})}}{1 + e^{\log(\text{odd})}} \tag{12}$$

The first step to build this model is feature extraction, which converts textual input into numerical vectors using techniques like Word2Vec, mBERT, and TF-IDF. After that, the original model receives these feature vectors and uses them to generate predictions and identify any mistakes or residuals. A new model is trained to forecast residuals, integrating prior predictions with the new model to create an enhanced combined model. This process is repeated until performance no longer significantly improves, focusing on the combined model's faults to improve forecasts progressively.

3.7.2 Extreme Gradient Boosting - XGBoost

XGBoost is an efficient and highly flexible execution of gradient boosting, which builds an ensemble of decision trees by minimizing a loss function iteratively [64]. It introduces several enhancements, such as tree pruning, a regularization term (L1 and L2) to control overfitting, and a split-finding algorithm that efficiently handles sparse data. XGBoost also supports parallel processing, distributed computing, and GPU acceleration, making it faster than traditional boosting algorithms [65]. The objective function is defined as:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{13}$$

Here, $l(y_i, \hat{y}_i)$ represents the loss function (e.g., mean squared error), and $\Omega(f_k)$ is the regularization term defined as:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{14}$$

where T is the number of leaves in the tree, w is the weight of the leaves, γ is the complexity penalty, and λ is the regularization parameter. This regularization helps prevent overfitting and ensures model generalization.

3.7.3 Adaptive Boosting - AdaBoost

AdaBoost is a boosting method that improves the accuracy of weak classifiers, typically shallow decision trees, by iteratively combining them into a weighted ensemble. After every iteration, it gives misclassified samples more weight, compelling later models to concentrate on more challenging data points. The final model aggregates the forecasts of all weak classifiers with weights proportional to their performance. While AdaBoost works well with simple learners and handles multi-class and binary classification, it is sensitive to noisy data and outliers, impacting its performance [66]. The weight update rule is given as:

$$w_i^{(t+1)} = w_i^{(t)} \exp\left(\alpha_t \cdot I(y_i \neq \hat{y}_i)\right) \tag{15}$$

Where $w_i^{(t)}$ is the weight of the i-th sample at iteration t, $I(y_i \neq \hat{y}_i)$ is an indicator function that equals 1 if the sample is misclassified, and α_t is the weight of the weak learner, computed as:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right) \tag{16}$$

Here, e_t is the classification error of the weak learner at iteration t.

3.7.4 Light Gradient Boosting Machine - LGBM

LightGBM is a gradient boosting framework designed for speed, efficiency, and scalability. It employs unique techniques such as histogram-based learning, which reduces memory usage, and leaf-wise tree growth, which splits trees based on maximum information gain, resulting in deeper trees with fewer splits [67]. LightGBM can handle large datasets with high dimensionality and categorical features directly, without extensive preprocessing. Its ability to distribute training across multiple machines or GPUs makes it faster and more resource-efficient than traditional methods, like XGBoost, while achieving comparable or better predictive performance [68]. The objective function of LightGBM is expressed as:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \lambda \sum_{j=1}^{m} \|w_j\|^2 \tag{17}$$

Where $l(y_i, \hat{y}_i)$ represents the loss function, and the regularization term penalizes the model complexity.

3.7.5 Voting Classifier - VC

The capacity of ensemble learning to mix many base models to obtain greater predictive performance has made it very popular in the field of ML. A well-known ensemble technique is the Voting Classifier, which uses the collective intelligence of the crowd to produce reliable forecasts [69]. A voting classifier is an ML technique that maximizes the probability that a given class will be the output after learning from an ensemble of many models. It can be applied to resolve problems with both regression and classification [70]. Two voting methods are used through the Ensemble Voting Classifier: "Soft" voting and "Hard" Voting.

- *Soft Voting:* A probability value representing the chance that in a soft voting system, each classifier provides a particular data item that is an element of the target class. After the outputs are weighted and totaled, the significance of the classifier is taken into consideration. Next, the target label having the highest total weighted probability is chosen by popular voting [71]. Showing in the equation.

$$y = \arg\max_i \sum_{j=1}^{m} W_j P_{ij}, \quad i \in \{0, 1\}, \quad j \in \{1, 2, \ldots, m\} \tag{18}$$

- *Hard Voting:* When classes are voted on by each classifier in hard voting (also called majority voting), the class that receives the most votes wins. Simply put, the predicted label for each item's distribution mode represents the ensemble's expected target label [72]. Here, the hard voting of each classifier C would be used to determine the class mark Y in the equation:

$$Y = \text{mode}\{C_1(x), C_2(x), \ldots, C_m(x)\} \tag{19}$$

In this paper, to build the voting classifier, we choose hard voting. The first step in the process is feature extraction, which turns textual input into numerical vectors using Word2Vec, mBERT, and TF-IDF. Each base model uses these vectors as its input characteristics. These feature vectors are used to separately train a variety of classifiers, including SVM, random forest, and gradient boosting. Next, based on fresh data, each model predicts using hard voting. In hard voting, the class label with the most votes is elected as the final forecast, which is decided by majority rule.

3.7.6 Stack Model

Stacking is a sophisticated ensemble learning technique that merges predictions from various base models, like SVM and RF, to create a meta-model, often employing linear regression or logistic regression. The base models are trained separately, and their predictions on the training set serve as input features for the meta-model. This meta-model leverages the strengths of the base models while minimizing their individual weaknesses, resulting in a more precise overall prediction [73].

$$\hat{y} = g(f_1(X), f_2(X), \ldots, f_n(X)) \tag{20}$$

Where $f_1, f_2, \ldots, f_n$ are the base models, and g is the meta-model, such as logistic regression or a neural network.

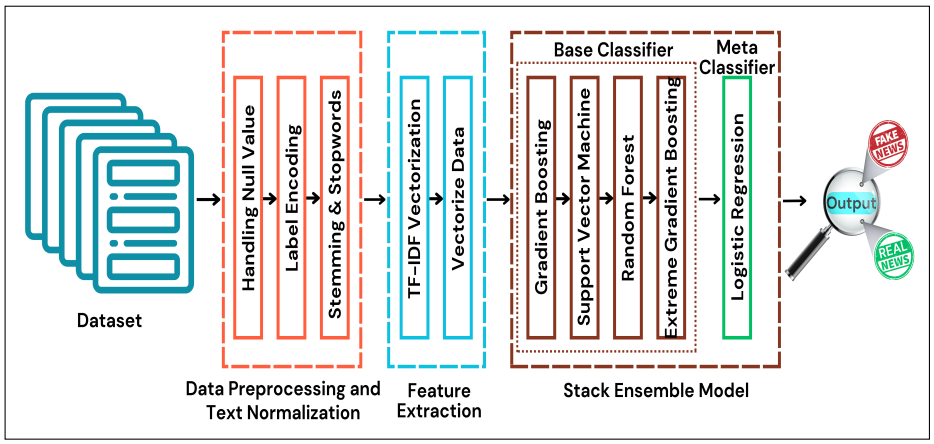


Fig. 10 Proposed Architecture for Stack Model to Detect Fake News

In this research, we developed a stacked ensemble model for fake news detection, employing the Random Forest, Support Vector Machine, Extreme Gradient Boosting, and Gradient Boosting as base classifiers, with Logistic Regression serving as the meta-classifier. The model integrates predictions from base classifiers to generate a final output, enhancing accuracy and robustness in fake news identification. Fig. 10 represents the architecture of the proposed model.

3.8 Proposed Algorithm for Fake News Detection

The proposed algorithm aims to detect fake news using machine learning classifiers. It processes two datasets an English dataset and a Bangla dataset. During pre-processing, null values are handled, categorical variables are converted to a numerical

format using a label encoder, and the dataset is partitioned into training and testing sets. Text normalization involves tokenization, stemming or lemmatization, and stop words to reduce words to their root form. TF-IDF, mBERT, and Word2Vec representations were created for feature extraction to capture word significance and context. Eleven machine learning classifiers, SVM, RF, DT, NB, GB, LR, XGB, LGBM, Adaboost, Stacking, and a Voting Classifier, are then initialized, trained, and used to predict the test data, with performance evaluated using metrics such as recall, accuracy, precision, and F1-score. Algorithm 1 outlines the step-by-step procedures of the fake news detection system.

Algorithm 1 Fake News Detection Procedure

Initialize: DS, PP, TN, FE, CL ▷ Initialize components
$DS \leftarrow$ Dataset, $X \leftarrow DS$[Content], $Y \leftarrow DS$[Label]
$PP \leftarrow$ [HNV, LE], $TN \leftarrow$ [TNZT, SWR, SL]
$FE \leftarrow$ [TF-IDF, Word2Vec, mBERT]
$CL \leftarrow$ [SVM, RF, DT, NB, GB, LR, XGB, AdaBoost, LGBM, Voting, Stack]
for $i = 1$ to $|FE|$ **do**
    $X_v \leftarrow FE[i](X)$ ▷ Vectorize X
    for $j = 1$ to $|FE|$ **do**
        Split X_v, Y into X_T, Y_T, x_t, y_t using seed j ▷ Split the Dataset
        for $k = 1$ to $|CL|$ **do**
            Train $CL[k]$ with (X_T, Y_T) ▷ Train Classifier
            Predict using $CL[k]$ on x_t ▷ Test the Classifier
            Display → Accuracy, Precision, Recall, F1 -Score ▷ Performance Metrics
        end for
    end for
end for
Deinitialize: DS, PP, TN, FE, CL ▷ Shutdown components

3.9 Experimental Setup

3.9.1 Environmental Setting

Experiments were conducted using an X64 processor with an Intel(R) Core i5-8265U CPU, running Windows 11 and operating at 1.60 GHz to 3.90 GHz. The processor was outfitted with 12 GB of DDR4 RAM. Google Colab and Jupyter Notebook were used to run machine learning and natural language processing applications. Testing the suggested model takes around an hour, and training it takes more than sixteen hours. This extended duration is due to the use of three datasets, with SMOTE applied to one of them, and the need to run each experiment across three different feature extraction techniques: TF-IDF, Word2Vec, and mBERT. These combinations require multiple training cycles, extending the total training time to over sixteen hours. Additionally, the use of 5-fold cross-validation and complex ensemble models, such as stacking and voting classifiers, contributes further to the computational load and time consumption.

3.9.2 Parameters of ML Classifier:

This section state the the parameter of different machine learning models which is used in this research. Each ml model is configured with specific hyperparameters

Table 3 Parameters of different ML models

Classifier	Parameters
SVM	(probability=True, random_state=42)
RF	(n_estimators=100, random_state=42)
DT	(max_depth=10, random_state=42)
LR	(max_iter=500, random_state=42)
GB	(n_estimators=100, random_state=42)
XGB	(use_label_encoder=False, eval_metric='mlogloss')
Adaboost	(n_estimators=50, random_state=42)
LightGBM	(n_estimators=100, learning_rate=0.1, random_state=42)

such as the number of estimators, maximum depth, iteration limits, learning rates, and random states. These parameters ensure consistency and reproducibility in the experiments. The table 3 provides a clear reference for the settings applied to each classifier, contributing to the reliability of the results.

3.9.3 Performance Parameters

We have employed three datasets, each of which has undergone testing and training using various machine learning techniques. Precision, F1 score, Recall, and Accuracy were used to determine each model's performance. The following are the performance parameters-

- ***Confusion Matrix:***

  Although a confusion matrix is not a performance matrix, it is required to construct other performance matrices. The classifier's traditional way to describe performance on a test data set whose true values are known is through a table called a confusion matrix. It makes it possible to see how an algorithm performs visually [15]. A confusion Matrix is shown in Table 4.

Table 4 Confusion Matrix for Real News and Fake News Classification

	Predicted Real News	**Predicted Fake News**
Actual Real News	TN	FN
Actual Fake News	FP	TP

Where,

TN = Machine predicted Fake and it's actually Fake
FN = Machine predicted Fake and it's actually Real
FP = Machine predicted Real and it's actually Fake
TP = Machine predicted Real and it's actually Real

- ***Accuracy:***
  When calculating the percentage of real or fake observations that are accurately anticipated, accuracy is frequently the most commonly utilized metric. The accuracy of an algorithm's performance can be evaluated by using the formula 12:

$$\text{Accuracy} = \left(\frac{TN + TP}{(FN + TN) + (TP + FP)} \right) \times 100 \tag{21}$$

- ***Precision:*** The precision score is the proportion of all instances that are forecasted as true to true positives. Among all the articles with positive predictions (true), precision in our instance indicates how many articles are identified as true. Equation-13 can be utilized to compute precision:

$$\text{Precision} = \left(\frac{TP}{FP + TP} \right) \times 100 \tag{22}$$

- ***Recall:***
  Total positive classifications that do not belong in the actual class are represented by the recall. For us, it means how many of the entire count of true articles are anticipated to be true. Recall can be calculated utilizing the equation 14 following:

$$\text{Recall} = \left(\frac{TP}{FN + TP} \right) \times 100 \tag{23}$$

- ***F1 Score:***
  The precision versus recall trade-off is demonstrated by the F1-score. Between each of the two, the harmonic mean is computed. For us, it takes into account both false negative and positive results. The F1-score can be identified utilizing the following formula-15.

$$\text{F1 Score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \times 100 \tag{24}$$

- ***ROC curve (Receiver Operating Characteristic curve):***
  The performance of a classifier across all classification thresholds is represented graphically by the Receiver Operating Characteristic curve, or ROC curve. The True Positive Rate (TPR), which is defined as follows,

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{25}$$

  and FPR is defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{26}$$

  The sensitivity vs. specificity trade-off is evaluated with the aid of the ROC curve. The overall capacity of the classifier to differentiate between classes is shown by the area under the ROC curve (AUC). The model performs better when its AUC is higher.

4 Results Analysis and Discussion

In this section of the paper, we analyze the results of the proposed model. Subsections 4.1 to 4.4 present the experimental results, including the analysis of the confusion matrix and ROC curve for each of the models tested. Subsection 4.5 provides an overall discussion of the proposed work, comparing it with existing approaches. Additionally, this subsection highlights the limitations of the current study and suggests areas for future improvement.

4.1 Result Analysis

The findings of the proposed model utilizing Word2Vec, mBERT and TF-IDF Bi-gram as feature extraction methods are presented in this section. Accuracy, recall, F1-score, and precision are utilized to analyze the efficiency of our proposed model. We used eleven distinct machine learning classifiers to determine fake news: SVM, RF, DT, LR, NB, GB, XGB, LGBM, Adaboost, VC, and the Stacking algorithm. To increase the model's efficacy, we employed the K-Fold cross-validation technique. All the outcomes mentioned in this paper are the mean results of the K-Fold (5-fold) method. Based on dataset-1, the best result was achieved by the Stacking model with 99.6% accuracy using TF-IDF, followed by Word2Vec + Logistic Regression (98.8%) and mBERT + Logistic Regression (99.4%). For Dataset-2, before SMOTE, the Stacking model with TF-IDF achieved 74.8%, SVM with Word2Vec reached 74.2%, and mBERT + Stacking scored 72.8%. After applying SMOTE, performance improved, with Stacking + TF-IDF reaching 85.2%, Word2Vec at 79.9%, and mBERT achieving 79.2%. On Dataset-3, the best result was again with TF-IDF + Stacking (98.4%), followed by Word2Vec (87.7%) and mBERT (96.1%). We have covered the outcomes of all models using TF-IDF Bi-gram, mBERT and Word2Vec for all datasets in this section.

4.1.1 Analysis of the Result for Dataset-1

The Table 5 summarizes the performance of ML classifiers applied to Dataset-1, using TF-IDF Bigram, Word2Vec, and mBERT as feature vectorization techniques across accuracy, precision, recall, and F1-score. For TF-IDF Bigram, the Stack model achieved the highest performance with an accuracy of 99.6%, and 99.8% F1-score, followed closely by ensemble methods like Voting and classifiers such as Gradient Boosting, XGB, and Adaboost. For Word2Vec, the Stack Model again stands out with the best results with 98.8% accuracy and an F1-score of 99.0%, demonstrating its consistent superiority. Although Word2Vec combined with SVM also achieved 98.8% accuracy, it lagged in other metrics such as precision, recall, and F1-score. Additionally, mBERT embeddings combined with the Logistic regression achieved 99.4% accuracy and 99.6% F1-score, followed by the stack model with an accuracy of 99.3% and F1-score of 99.2%, marking a strong performance in the contextual embedding setting. These findings highlight that the TF-IDF Bigram-based Stack Model delivers the most robust performance for dataset 1, offering marginally better results than Word2Vec and mBERT across most metrics.

4.1.2 Analysis of Results for Dataset-2 without using SMOTE

Table 6 presents the performance of ML classifiers for Dataset-2 without using SMOTE, comparing TF-IDF, Word2Vec, and mBERT as feature vectorization techniques across accuracy, precision, recall, and F1-score. For TF-IDF, the Stacking Model achieved the highest performance with an accuracy of 74.8% and an F1-score of

Table 5 Performance comparison using different feature vectorization techniques and classifiers for Dataset-1.

FV Techniques	ML Classifier	Accuracy	Precision	Recall	F1-Score
TF-IDF	SVM	98.6%	98.6%	98.6%	98.6%
	RF	98.2%	98.2%	98.2%	98.2%
	DT	99.0%	99.0%	99.0%	99.0%
	LR	97.8%	98.0%	98.0%	98.0%
	GB	99.5%	99.6%	99.4%	99.4%
	Gaussian NB	84.6%	84.6%	84.6%	84.6%
	XGB	99.5%	99.4%	99.4%	99.4%
	Adaboost	99.6%	99.6%	99.6%	99.6%
	LightGBM	99.4%	99.4%	99.4%	99.4%
	Voting Classifier	99.5%	99.5%	99.5%	99.5%
	Stack Model	**99.6%**	**99.8%**	**99.8%**	**99.8%**
Word2Vec	SVM	98.8%	98.7%	98.7%	98.7%
	RF	97.8%	97.8%	97.8%	97.8%
	DT	94.6%	94.6%	94.6%	94.6%
	LR	**98.8%**	**99.0%**	**99.0%**	**99.0%**
	GB	98.0%	98.0%	98.0%	98.0%
	Gaussian NB	93.0%	93.0%	93.0%	93.0%
	XGB	98.2%	98.2%	98.2%	98.2%
	Adaboost	97.2%	97.0%	97.0%	97.0%
	LightGBM	98.1%	98.0%	98.0%	98.0%
	Voting Classifier	98.8%	98.6%	98.6%	98.6%
	Stack Model	98.6%	98.6%	98.6%	98.6%
mBERT	SVM	89.4%	87.6%	85.6%	86.3%
	RF	97.8%	97.8%	97.8%	97.8%
	DT	95.6%	94.4%	94.0%	95.4%
	LR	**99.4%**	**99.6%**	**99.6%**	**99.6%**
	GB	89.2%	87.6%	84.4%	85.8%
	Gaussian NB	77.2%	74.2%	79.4%	74.4%
	XGB	95.8%	94.2%	95.0%	94.6%
	Adaboost	85.2%	82.0%	79.4%	80.4%
	LightGBM	95.5%	94.0%	95.0%	94.4%
	Voting Classifier	90.6%	89.0%	86.8%	87.6%
	Stack Model	99.3%	99.2%	99.2%	99.2%

68%, outperforming individual classifiers such as XGB and LightGBM. For Word2Vec, the Support Vector Machine recorded the best accuracy at 74.2% and an F1-score of 63%, closely followed by the Stacking Model, which obtained 73.9% accuracy and 61.2% F1-score. In the case of mBERT, the Stacking Model again stood out with the highest accuracy of 72.8% and an F1-score of 58.4%, followed by SVM, DT, LR, and Voting classifier, while other classifiers showed comparatively lower scores. These findings highlight that, although mBERT lags slightly behind in raw performance, it still provides a viable contextual embedding alternative. Among the three techniques, the TF-IDF-based Stacking Model continues to deliver the most robust results for Dataset-2 before using SMOTE, with Word2Vec and mBERT offering moderately strong alternatives depending on classifier choice.

4.1.3 Analysis of Results for Dataset-2 using SMOTE

The Table 7 presents a performance comparison of various machine learning classifiers using TF-IDF, Word2Vec, and and mBERT as feature vectorization techniques for Dataset-2 after balancing the dataset using SMOTE. The performances are assessed across accuracy, precision, recall, and F1-score. For TF-IDF, the Stack Model achieved

Table 6 Performance comparison of different feature vectorization techniques and classifiers on Dataset-2 before applying SMOTE.

FV Techniques	ML Classifier	Accuracy	Precision	Recall	F1-Score
TF-IDF	SVM	71.0%	66.0%	56.0%	54.2%
	RF	72.2%	69.0%	59.4%	59.0%
	DT	68.4%	62.8%	62.6%	62.6%
	LR	71.4%	66.2%	59.2%	59.2%
	GB	74.3%	77.4%	63.8%	67.2%
	Gaussian NB	71.2%	65.8%	59.4%	58.2%
	XGB	74.8%	71.6%	64.4%	66.0%
	Adaboost	73.6%	70.6%	61.6%	62.0%
	LightGBM	74.7%	71.4%	65.4%	66.0%
	Voting Classifier	73.6%	70.0%	60.0%	60.0%
	Stack Model	**74.8%**	**73.4%**	**66.4%**	**68.0%**
Word2Vec	**SVM**	**74.2%**	**77.0%**	**61.4%**	**63.0%**
	RF	72.4%	67.4%	60.6%	61.0%
	DT	61.8%	56.4%	56.6%	56.4%
	LR	73.4%	71.6%	60.2%	60.0%
	GB	73.0%	70.8%	59.6%	59.2%
	Gaussian NB	61.2%	58.2%	59.4%	58.2%
	XGB	70.9%	64.6%	61.2%	61.2%
	Adaboost	72.0%	67.4%	59.4%	59.8%
	LightGBM	72.9%	68.6%	61.0%	61.0%
	Voting Classifier	73.8%	70.8%	62.0%	62.6%
	Stack Model	73.9%	73.0%	61.2%	61.2%
mBERT	SVM	72.2%	76.2%	55.8%	53.0%
	RF	68.6%	60.8%	54.0%	51.8%
	DT	72.2%	69.6%	58.4%	57.6%
	LR	72.2%	69.6%	58.4%	72.2%
	GB	71.6%	68.6%	56.0%	54.0%
	Gaussian NB	53.0%	55.6%	56.2%	52.2%
	XGB	67.6%	59.0%	56.0%	56.0%
	Adaboost	69.6%	61.8%	56.2%	55.4%
	LightGBM	69.0%	63.4%	56.4%	55.4%
	Voting	72.4%	71.2%	57.4%	55.8%
	Stack Model	**72.8%**	**72.0%**	**59.0%**	**58.4%**

the best performance with an accuracy of 85.2% and an F1-score of 85%, followed by SVM and the Voting Classifier. In contrast, for Word2Vec, the Stack Model again outperformed other classifiers with an accuracy of 79.9% and F1-score of 80%, with Random Forest showing competitive results among individual classifiers. For mBERT, the Stack Model achieved the highest scores among the contextual embedding-based models, with an accuracy of 79.2% and an F1-score of 79.8%. These findings highlight that, while Word2Vec mBERT demonstrates strong performance under balanced conditions, the TF-IDF-based Stack Model remains the most effective overall. TF-IDF + Stack Model is our best model for identifying fake news in Bangla.

4.1.4 Analysis of Results for Dataset-3

Table 8 illustrates the performance outcomes of multiple machine learning classifiers applied to Dataset-3, using TF-IDF, Word2Vec, and mBERT as feature vectorization techniques. Across all metrics, accuracy, precision, recall, and F1-score, the Stacking Model consistently delivered the strongest results for each representation method. With TF-IDF, it achieved the highest scores (98.4% accuracy and 98.0% F1-score), narrowly outperforming other individual models such as XGB, LGBM, and RF. Under

Table 7 Performance comparison of different feature vectorization techniques and classifiers on Dataset-2 After applying SMOTE.

FV Techniques	ML Classifier	Accuracy	Precision	Recall	F1-Score
TF-IDF	SVM	84.1%	84.0%	84.0%	84.0%
	RF	82.2%	82.6%	82.6%	82.6%
	DT	72.6%	72.6%	72.6%	72.6%
	LR	77.1%	77.4%	77.4%	77.4%
	GB	79.1%	79.2%	79.2%	79.2%
	Gaussian NB	74.2%	74.2%	74.2%	74.2%
	XGB	81.5%	81.5%	81.5%	81.5%
	Adaboost	75.8%	75.8%	75.8%	75.8%
	LightGBM	83.2%	83.0%	83.0%	83.0%
	Voting Classifier	84.3%	84.0%	84.0%	84.0%
	Stack Model	**85.2%**	**85.0%**	**85.0%**	**85.0%**
Word2Vec	SVM	66.2%	66.8%	66.4%	66.2%
	RF	79.9%	80.0%	80.0%	80.0%
	DT	67.1%	67.2%	67.2%	67.2%
	LR	61.8%	61.8%	61.8%	61.8%
	GB	65.8%	66.0%	65.8%	65.4%
	Gaussian NB	58.7%	58.6%	58.6%	58.6%
	XGB	75.5%	75.6%	75.6%	75.6%
	Adaboost	61.6%	61.4%	61.4%	61.4%
	LightGBM	73.2%	73.2%	73.2%	73.2%
	Voting CLassifier	74.6%	74.6%	74.6%	74.6%
	Stack Model	**79.9%**	**80.0%**	**80.0%**	**80.0%**
mBERT	SVM	65.8%	65.8%	65.8%	65.8%
	RF	78.6%	78.6%	78.6%	78.6%
	DT	63.6%	63.6%	63.6%	63.0%
	LR	65.4%	65.4%	65.4%	65.4%
	GB	67.4%	67.4%	67.4%	67.4%
	Gaussian NB	56.8%	58.2%	56.8%	55.0%
	XGB	76.8%	76.8%	76.8%	76.8%
	Adaboost	62.6%	62.6%	62.6%	62.6%
	LightGBM	74.0%	74.0%	74.0%	74.0%
	Voting	70.2%	70.2%	70.2%	70.2%
	Stack Model	**79.2%**	**79.8%**	**79.2%**	**79.8%**

the Word2Vec representation, the Stacking approach again led the pack with 97.7% accuracy and a 97.8% F1-score, closely followed by LGBM and XGB. This further emphasizes its reliability when handling dense vector embeddings. Although the overall performance of mBERT embeddings was slightly lower than the other two methods, the Stacking Model still came out on top with 96.1% accuracy and a 95.6% F1-score. Notably, classifiers such as LightGBM and XGB also performed competitively, highlighting the effectiveness of contextual embeddings in conjunction with traditional machine learning models. Overall, while all three vectorization strategies benefited from ensemble-based approaches, TF-IDF combined with Stacking yielded the best results for Dataset-3, with Word2Vec close behind, and mBERT offering a strong alternative in contextual feature representation.

4.2 Comparing Overall Performance

After applying two different feature extraction techniques to eleven distinct Machine Learning classifiers, in this part, we will compare the overall performance of both Bangla and English datasets and compare our model's Accuracy using different bar charts.

Table 8 Performance comparison of different feature vectorization techniques and classifiers on Dataset-3.

FV Techniques	ML Classifier	Accuracy	Precision	Recall	F1-Score
TF-IDF	SVM	95.6%	94.5%	93.8%	94.4%
	RF	98.1%	98.6%	96.8%	97.6%
	DT	93.2%	92.6%	89.6%	91.0%
	LR	93.2%	92.4%	89.6%	91%
	GB	92.7%	93.4%	87.6%	90.0%
	Gaussian NB	86.6%	82.8%	89.0%	84.8%
	XGB	98.1%	97.6%	97.8%	97.8%
	Adaboost	90.6%	88.9%	86.8%	87.8%
	LightGBM	98.2%	97.6%	97.6%	97.6%
	Voting Classifier	97.2%	96.8%	95.8%	96.4%
	Stack Model	**98.4%**	**98.0%**	**98.0%**	**98.0%**
Word2Vec	SVM	92.6%	91.0%	89.8%	90.6%
	RF	97.0%	95.8%	96.8%	96.8%
	DT	92.4%	90.8%	89.6%	90.2%
	LR	92.4%	90.8%	89.6%	90.2%
	GB	92.8%	91.6%	90.0%	90.6%
	Gaussian NB	84.5%	80.6%	86.0%	82.0%
	XGB	97.5%	96.6%	97.6%	97.0%
	Adaboost	91.3%	89.0%	88.6%	88.8%
	LightGBM	97.4%	96.2%	97.6%	96.2%
	Voting CLassifier	93.5%	92.2%	91.6%	91.6%
	Stack Model	**97.7%**	**97.0%**	**97.4%**	**97.8%**
mBERT	SVM	89.7%	87.6%	85.6%	86.6%
	RF	95.5%	94.6%	94.0%	94.8%
	DT	89.0%	87.0%	84.6%	85.6%
	LR	89.0%	87.0%	84.4%	84.4%
	GB	89.2%	87.6%	84.6%	85.8%
	Gaussian NB	77.2%	74.2%	79.6%	74.4%
	XGB	95.8%	94.8%	95.0%	94.6%
	Adaboost	85.2%	82.0%	79.6%	80.6%
	LightGBM	95.5%	94.0%	95.0%	94.8%
	Voting	90.6%	89.0%	86.8%	87.8%
	Stack Model	**96.1%**	**95.4%**	**94.8%**	**95.6%**

4.2.1 Accuracy comparison for Dataset-1

Fig. 11 compares the accuracy of machine learning models using TF-IDF, Word2Vec, and mBERT for Dataset-1. Across most classifiers, TF-IDF shows slightly higher accuracy compared to Word2Vec and mBERT; the Stack Model achieved the best accuracy for TF-IDF (99.6%), and the logistic regression achieved the best accuracy for Word2Vec (98.8%) , while mBERT also demonstrated strong performance, with the Stacking model achieving 99.4%. Classifiers like Gradient Boosting, XGB, and Voting Classifier also exhibit strong accuracy for TF-IDF. Word2Vec generally performs marginally lower across models, with notable drops in classifiers like Naive Bayes, which achieved only 84.6% accuracy. mBERT showed competitive performance in several classifiers but also suffered noticeable accuracy drops in models such as GNB (77.2%) and AdaBoost (85.2%). This analysis reinforces the superiority of the TF-IDF-based Stack Model, particularly for dataset 1.

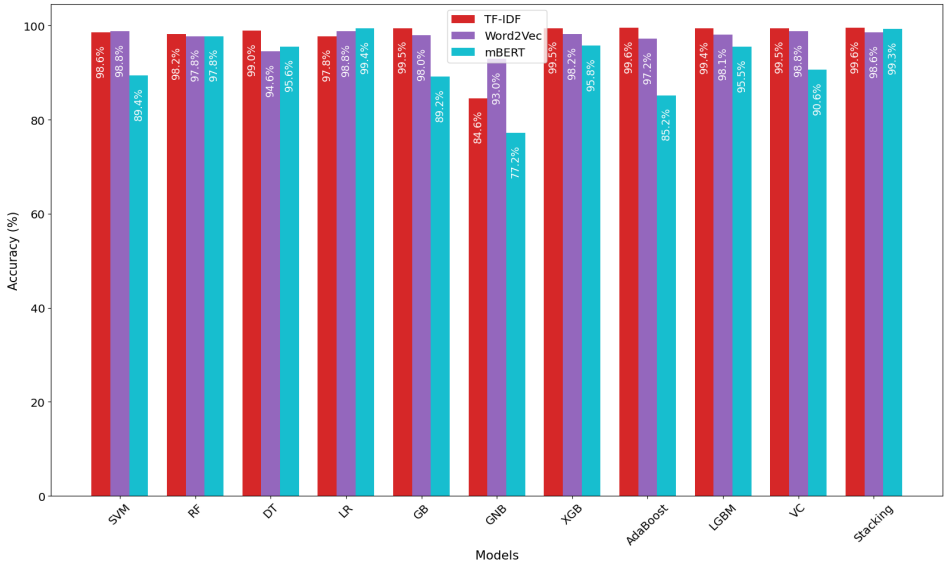


Fig. 11 Comparing the Accuracy of TF-IDF, Word2Vec, and mBERT for all Models for Dataset-1

4.2.2 Accuracy comparison for Dataset-2 Before using SMOTE

Fig. 12 compares the accuracy of machine learning models for Dataset-2 before applying SMOTE, using TF-IDF, Word2Vec, and mBERT as feature vectorization techniques. While all three techniques yield comparable results, TF-IDF generally performs slightly better across most classifiers. The Stack Model demonstrates the highest accuracy for TF-IDF, which achieved 74.8% accuracy, and the support vector machine achieved the highest for Word2Vec, which reached 74.2% accuracy , while mBERT's best result was observed in the stacking model, achieving 72.8% accuracy. Notable variations include classifiers like Naive Bayes, where TF-IDF outperforms Word2Vec and mBERT significantly. For instance, mBERT yielded only 53.0% accuracy in the GNB model, marking substantial performance drops. Overall, the results highlight that TF-IDF-based models, especially the Stack Model, consistently achieved superior accuracy in dataset 2 without oversampling.

4.2.3 Accuracy comparison for Dataset-2 After using SMOTE

Fig. 13 illustrates the accuracy of machine learning models for Dataset-2 after applying SMOTE, using TF-IDF, Word2Vec, and mBERT as feature vectorization techniques. While all three methods exhibit varying levels of effectiveness across classifiers, TF-IDF consistently achieves the highest overall accuracy. The Stack Model demonstrates the best performance across all three techniques, achieving 85.2% with TF-IDF, 79.9% with Word2Vec, and 79.2% with mBERT. Among individual classifiers, Random Forest and XGBoost perform notably well with TF-IDF, reaching 82.2% and 81.5%, respectively. Word2Vec and mBERT also show competitive results in some ensemble models, with mBERT achieving 76.8% in XGBoost and 74.0% in LGBM. However, mBERT underperforms in specific models such as Gaussian Naive Bayes, Adaboost, and Decision Trees, with accuracies of just 56.8%, 62.6%, and 63.6%, respectively, indicating weaker compatibility with simpler classifiers. These results

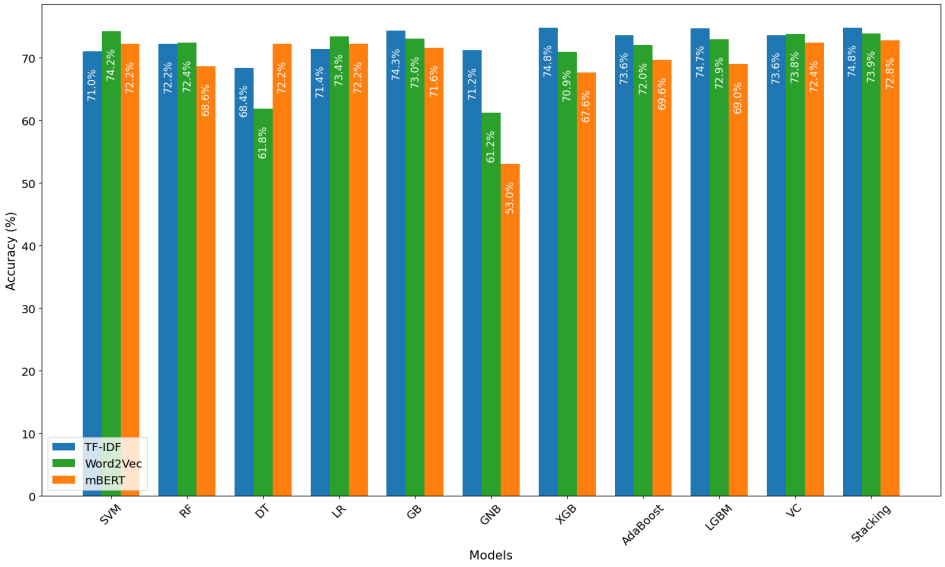


Fig. 12 Comparing the Accuracy of TF-IDF, Word2Vec, and mBERT for all Models for Dataset-2 Before Balancing the Dataset using SMOTE

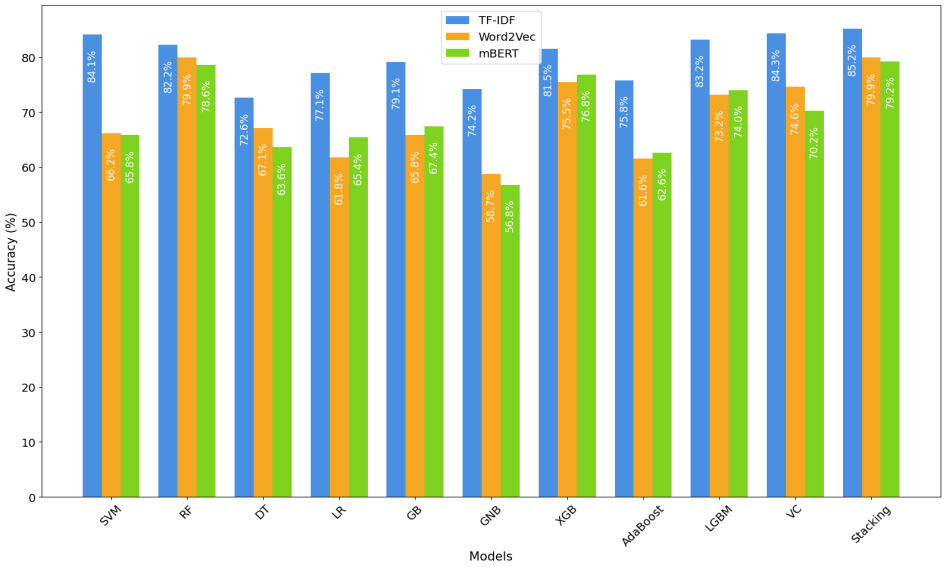


Fig. 13 Comparing the Accuracy of TF-IDF, Word2Vec, and mBERT for all Models for Dataset-2 after Balancing the Dataset using SMOTE

confirm the effectiveness of TF-IDF-based Stack models, particularly when performed with SMOTE, for enhancing classification accuracy in an imbalanced dataset, while also demonstrating that mBERT has potential in complex ensemble methods but is less robust in basic classification models.

4.2.4 Accuracy comparison for Dataset-3

Fig. 14 illustrates the accuracy of machine learning models for Dataset-3 using TF-IDF, Word2Vec, and mBERT as feature vectorization techniques. TF-IDF consistently achieves the highest accuracy across nearly all classifiers. The Stack Model achieves the top performance with TF-IDF at 98.4%, followed closely by Word2Vec (97.7%) and mBERT (96.1%). Individual classifiers such as XGBoost and LGBM also perform notably well with TF-IDF, reaching 98.1% and 98.2%, respectively. Word2Vec maintains high performance in ensemble models, while mBERT shows strong results in classifiers like XGBoost (95.8%) and LGBM (95.5%). Nonetheless, mBERT exhibits reduced performance in simpler models such as Gaussian Naive Bayes (77.2%) and Adaboost (85.2%) compared to the other methods. These results confirm that TF-IDF remains a robust feature representation approach, particularly when used with ensemble classifiers. At the same time, mBERT demonstrates strong potential in complex models but is less reliable in basic classifiers.

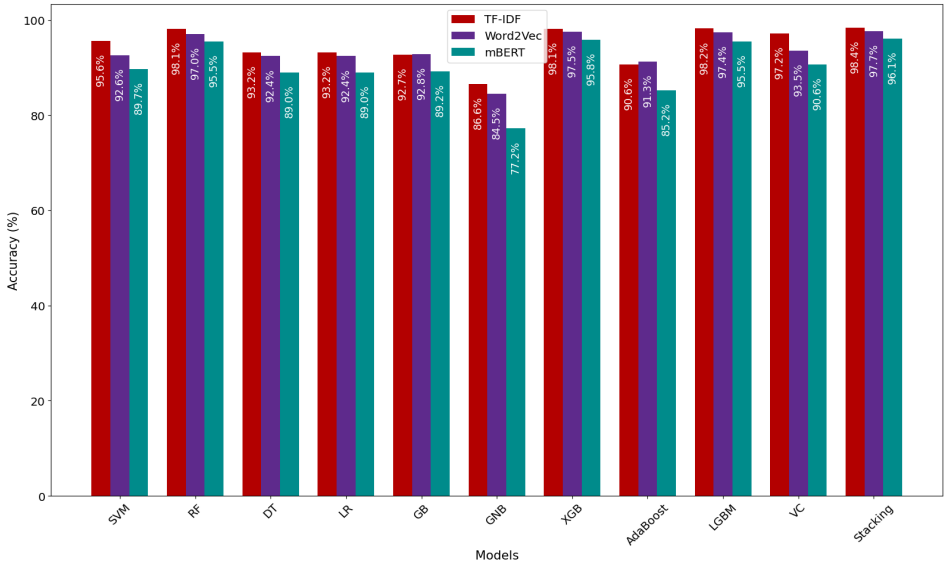
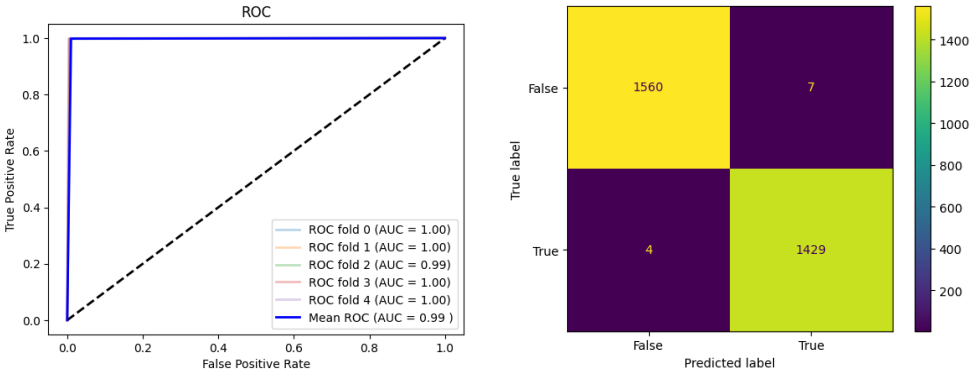


Fig. 14 Comparing the Accuracy of TF-IDF Word2Vec, and mBERT for all Models for Dataset-3

4.3 ROC Curve and Confusion Matrix Representation

In this section, we present the ROC curves and confusion matrices to provide a deeper evaluation of model performance. As identified in the above result analysis and comparisons, the combination of TF-IDF and the Stacking model consistently delivered the highest accuracy across all three datasets. Therefore, we focus exclusively on this best-performing configuration and display its ROC curves and confusion matrices for each dataset.

Fig. 15 represents the ROC Curve and Confusion Matrix of TF-IDF + Stack Model for Dataset 1. The Stack model is the best classifier for Dataset 1 when using TF-IDF bigram, achieving an accuracy of 99.6%. The ROC curve indicates a mean ROC

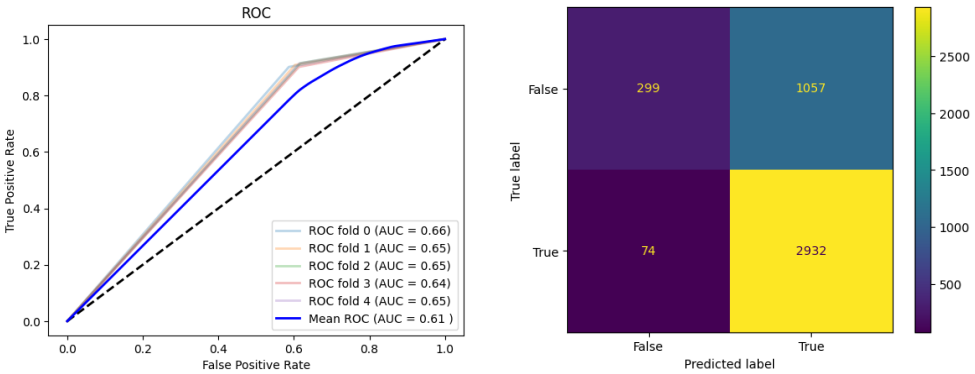


(a) ROC Curve (b) Confusion Matrix

Fig. 15 ROC Curve & Confusion Matrix of TF-IDF + Stack Model for Dataset-1

of 99%. The Confusion Matrix reveals high true negative (1560) and true positive (1429) values. In contrast, the false negative (7) and false positive (4) values are comparatively low. This indicates that the model performed well in this scenario.

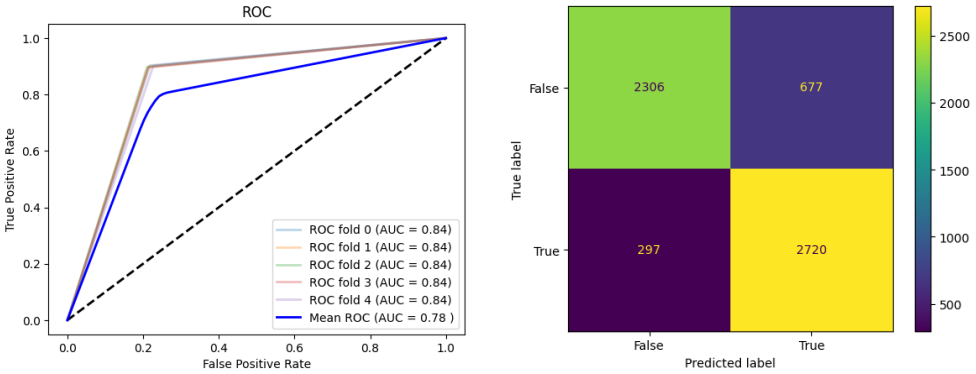
Fig. 16 presents the ROC Curve and Confusion Matrix for the TF-IDF combined with the Stack Model for Dataset-2 before using SMOTE to balance the dataset. The ROC curve showcases the model's performance across five folds, achieving an average AUC score of 0.61. The Confusion Matrix highlights the model's prediction outcomes, with 299 true negatives, 2932 true positives, 1057 false negatives, and 74 false positives. These results emphasize the model's strong predictive accuracy and robustness, particularly in identifying true positives.



(b) ROC Curve (b) Confusion Matrix

Fig. 16 ROC Curve & Confusion Matrix of TF-IDF + Stack Model before balancing the Dataset-2 using SMOTE

Fig. 17 presents the ROC Curve and Confusion Matrix for the TF-IDF + Stack Model after balancing the data using SMOTE for Dataset-2. The ROC Curve demonstrates the model's strong classification capability, with a mean AUC of 0.78. The

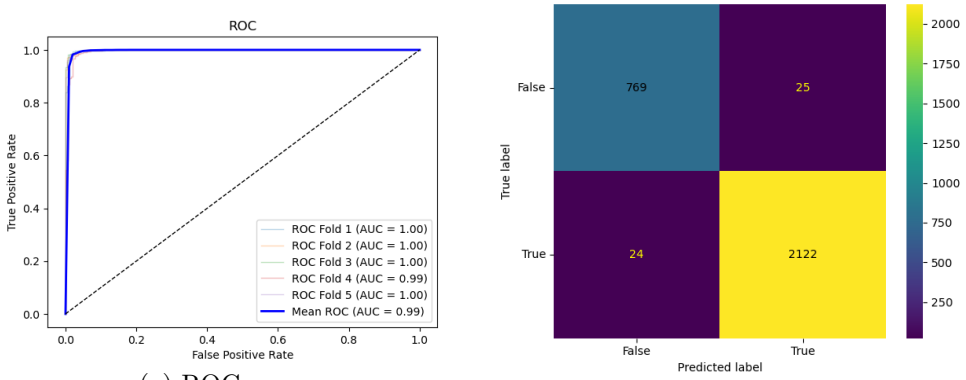


(a) ROC curve (b) Confusion Matrix

Fig. 17 ROC Curve & Confusion Matrix of TF-IDF + Stack Model after balancing the Dataset-2 using SMOTE

Confusion Matrix reveals that the model accurately predicted 2,306 true negatives and 2,720 true positives, while misclassifying 677 false negatives and 297 false positives. The integration of SMOTE effectively addressed data imbalance, resulting in a more balanced class distribution that enhanced the overall model performance and improved predictive accuracy.

Fig. 18 illustrates the ROC Curve and Confusion Matrix for Dataset 3, evaluated using the TF-IDF combined with the Stacked Model. This configuration achieved outstanding performance, reaching an accuracy of 98.4%. The ROC curve demonstrates a strong classification capability with a mean AUC of 99%. The Confusion Matrix highlights the model's effectiveness, with a high number of true negatives (769) and true positives (2122), while maintaining minimal false negatives (25) and false positives (24). These results demonstrate that the TF-IDF + Stacked Model configuration is highly effective and capable of delivering strong performance on Dataset 3.



(a) ROC curve (b) Confusion Matrix

Fig. 18 ROC Curve & Confusion Matrix of TF-IDF + Stack Model for Dataset-3

Based on the analysis of the ROC curves and Confusion Matrices, it is evident that the TF-IDF combined with the Stacked model is the best-performing approach across all three datasets: English (Dataset 1), Bangla (Dataset 2), and the BanFakeNews (Dataset 3) dataset for fake news detection.

4.4 Computational Cost & Space Requirement

Table 9 presents the computational cost and storage requirements of various machine learning models used for fake news detection. The table compares each model's training time (in seconds) and the required memory (in megabytes). This comparison is based only on TF-IDF-based classifiers, as TF-IDF produced the best results in our experiments. Among the models, Logistic Regression (LR) has the shortest training time (17s) with a moderate memory requirement of 662 MB, whereas the Voting model takes the longest time (2246s) with a memory usage of 1373 MB. The Stacked Model, which achieved the highest accuracy 99.6%, required 1821s for training and used 1223 MB of memory. Notably, SVM had the highest memory consumption (3816 MB), making it the most space-intensive model, while Decision Tree (DT) was the most efficient in terms of storage, requiring only 57 MB. Despite the stacked model's relatively high training time, it demonstrates a balanced trade-off between performance and memory efficiency, making it the optimal choice for fake news detection.

Table 9 Computational Cost & Space Requirement for TF-IDF + ML Classifiers

Model	Training Time	Required Space
SVM	91s	3816MB
RF	116s	3835MB
DT	27s	57MB
LR	17s	66MB
GB	634s	1322MB
Gaussian NB	98s	2453MB
XGB	109s	3997MB
Adaboost	235s	1378MB
LightXGB	62s	1181MB
Voting	2246s	1373MB
Stack Model	1821s	1223MB

4.5 Discussion and Limitations

The whole world faces a serious problem with fake news, especially in countries like Bangladesh, India, and Pakistan. Numerous studies have been done before to eliminate or stop the spread of fake news, including the application of ML and DL. Studies on the subject of identifying fake news are now accessible and available in many journals. To prevent the spread of rumors, we developed this model to identify false news. In light of this study, an enhanced false news identification system utilizing word2vec, mBERT, TF-IDF, and eleven machine learning algorithms that were proposed. The models are trained using three publicly available datasets. In our observation, if we did not make any mistakes, no paper was published, and no work was done specifically before using the Bangla dataset (Dataset-2).

To determine the most effective model for fake news detection, we systematically compared eleven machine learning algorithms based on multiple performance

parameters, including accuracy, precision, recall, F1-score, and AUC. Additionally, we assessed each model using a confusion matrix and ROC curve analysis while also considering computational cost in terms of time and space complexity. Based on our extensive evaluation, we found that TF-IDF combined with the stacked model emerged as the best-performing classifier, achieving an impressive accuracy of 99.6%. Although its time complexity is higher than other models, making it the most time-consuming approach, the stacked model requires comparatively less memory. We also integrated mBERT as a feature extraction method and classified using the same Stacking model. While mBERT-based models showed competitive performance, the TF-IDF + Stacking configuration still outperformed all others across all evaluation metrics, making it the most effective choice for fake news detection. Given the strong performance of our stacked model, we did not pursue more complex AI models, such as deep learning architectures, as they would further increase computational costs in terms of both training time and memory usage without guaranteeing significant performance improvements.

In addition, we compared our best-performing stacked model with prior research in the field of fake news detection to assess its effectiveness. As discussed in the Related Work section, several studies have explored similar approaches. Our results are evaluated against existing methodologies from previous studies, as presented in Table 10. This comparison highlights the advantages of our proposed model over traditional classifiers, demonstrating its superior performance in identifying fake news.

Table 10 Comparison of Existing Works with Our Proposed Model

Ref.	Model	Algorithms/Techniques	Highest Accuracy
[23]	ML Model	RF, NB, LR	99.06% (RF)
[24]	ML Model	ID-3, NB, RF, AdaBoost	98.98% (AdaBoost)
[25]	ML Model	SVM, NB, RF, LR	97% (SVM)
[26]	ML + FV + Big Data	RF, DT, LR, Ensemble	93.45% (HashingTF-IDF_Ensembler)
[27]	ML Model	SVM, RF, KNN, NB, LR, Bagging, Boosting	97.31% (RF)
[28]	ML Model	DT, LR, KNN, SVM, LSVM, SGD	93.5% (LSVM)
[29]	ML Model	Gaussian NB, SVM, LR, RF, Multinomial NB, AdaBoost, GB, Voting	87% (Gaussian NB)
[30]	ML Model	Multinomial NB, SVM	96.64% (SVM), 93.32% (Multinomial NB)
Proposed Model	Feature Vectorization + ML	SVM, RF, DT, LR, GB, NB, XGB, AdaBoost, LightGBM, Voting, and Stack + (Word2Vec, mBERT, TF-IDF)	99.6% for Dataset 1, 85.2% for Dataset 2, and 98.4% for dataset-3 (Stack + TF-IDF)

When we compared our English fake news detection method to the classification methods in related works, our proposed classifier produced significantly better results. Among the papers listed in Table 10, *V. Asha et al.* [23] achieved the highest accuracy of 99%. In contrast, [29] used Gaussian Naive Bayes (NB) to detect fake news in

Bengali, achieving 87% accuracy, which was the best among their techniques. Similarly, [30] used SVM and MNBs to identify fake news in Bangla, with SVM achieving a 96.64% accuracy, the highest in their study.

When comparing our proposed Bangla fake news detection method to previous studies, the initial improvement may seem modest. This was largely due to the limited size and quality of the initial Bangla dataset, as well as the lack of essential language tools. Notably, our initial evaluation was conducted on a newly introduced dataset that had not been used in prior research. However, when we later tested our model on the well-established BanFakeNews dataset, it not only performed well but also outperformed existing methods, demonstrating its effectiveness on more established and structured data. While our main objective was to develop a robust model for under-resourced languages rather than to chase state-of-the-art accuracy, these results validate the strength of our approach. We aim to continue improving Bangla fake news detection by incorporating richer datasets and more advanced features in future work.

This study offers a thorough literature analysis, which is valuable to both students and researchers. By identifying the most frequently discussed topics in the fake news domain, it helps journals better understand how fake news spreads, supporting governments, funding organizations, and society in making more informed decisions about future efforts.

5 Conclusion and Future Work

This study combined eleven machine learning algorithms with feature extraction methods such as Word2Vec and TF-IDF to present an improved fake news detection system. The most successful model for detecting fake news is the Stack Model, in conjunction with TF-IDF. It achieved 99.6% accuracy and an F1-score of 99.8% for English fake news identification and 85.16% accuracy for Bangla fake news identification. Additionally, it achieved 98.4% accuracy on the BanFakeNews dataset, demonstrating strong performance even in a low-resource language. The system's accuracy on the Bangla dataset (85.2%) was lower than its performance on the English dataset, highlighting the challenges of fake news detection in low-resource languages, where high-quality annotated datasets are limited. Models like TF-IDF and Word2Vec primarily rely on word-level representations, which often fail to capture deeper contextual nuances such as sarcasm, implied meaning, or cultural references. This can lead to misclassifications, particularly when dealing with ambiguous or misleading content. Additionally, models trained on static datasets may struggle to generalize to emerging patterns of misinformation, as fake news content evolves rapidly. Future research could explore advanced language models like BERT, LSTM, GPT, BanglaBERT, AlBERT, DistilBERT, and RoBERTa, which excel in understanding context and semantics and may enhance fake news detection in low-resource languages like Bangla. Additionally, we plan to apply deep learning and transfer learning techniques to improve prediction accuracy across various languages, including Bangla. Developing models that can identify and classify fake news in real time as it spreads on social media is essential. This requires algorithms capable of processing streaming data efficiently and adaptively. Integrating explainability techniques would enhance the system's transparency, making it more valuable for journalists and fact-checkers. These techniques would help users understand why certain news is flagged as fake, fostering trust in the system. Furthermore, in the future we will focus on exploring cross-lingual transfer learning, conducting a detailed error analysis to categorize challenging news types, and performing ablation studies to quantify the unique contribution of our stacking ensemble.

Ethical Consideration

While our study aims to mitigate the spread of fake news, it is essential to consider the ethical implications of automated fake news detection. Misclassifications can lead to severe consequences, including unjust censorship or the suppression of legitimate information. Furthermore, biases in datasets and models may disproportionately impact certain groups or viewpoints. Since we used publicly available datasets, we acknowledge the importance of transparency and fairness in both data collection and model evaluation. Ensuring the responsible use of AI in combating misinformation remains a critical direction for future work.

Acknowledgments

The study presented here is the result of a collaborative effort by the authors. No external support was received for this study.

Authors Contributions

Conceptualization: Md. Sabbir Hossen, Pabon Shaha, Anichur Rahman; Methodology: Md. Sabbir Hossen, Pabon Shaha, Anichur Rahman; Formal analysis and investigation: Fahjimatus Sabah, K. M. Mursalin Billah Rezwan; Writing - original draft preparation: Md. Sabbir Hossen, Pabon Shaha, Md. Mowahibur Rahman Twake, and Abu Saleh Musa Miah; Writing - review and editing: Md. Sabbir Hossen, Pabon Shaha, Anichur Rahman, Fahim Al Farid, Hezerul Abdul Karim; Funding acquisition: Fahim Al Farid, Abu Saleh Musa Miah, Hezerul Abdul Karim; Resources: Fahim Al Farid, Abu Saleh Musa Miah, Fahjimatus Sabah; Supervision: Pabon Shaha, and Anichur Rahman.

Declaration of Fake News Detection in the Writing Process

The authors used Quilbot and Grammarly to enhance the language and readability of the work during its preparation. After utilizing these tools, the authors assumed full responsibility for the content, thoroughly reviewing and making any necessary revisions before publication.

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interest.

Data Availability

The source code is available via the GitHub link. The proposed approach and schematic diagram are shared with the scientific community for public access: https://github.com/PabonSC/Fake-News-Detection.

References

[1] Alghamdi, J., Luo, S., Lin, Y. (2024). A comprehensive survey on machine learning approaches for fake news detection. *Multimedia Tools and Applications*.

[2] Rahman, A., Khan, M. S. I., Montieri, A., Islam, M. J., Karim, M. R., et al. (2024). Blocksd-5gnet: Enhancing security of 5g network through blockchain-sdn with ml-based bandwidth prediction. *Transactions on Emerging Telecommunications Technologies*, *35*(*4*), e4965.

[3] Smaïli, K., Anissa, H., David, L., Djegdjiga, A. (2024). Boutef: Bolstering our understanding through an elaborated fake news corpus//*The 8th International Conference on Arabic Language Processing*.

[4] Marchal, N., Hoes, E., Klüser, K. J., Hamborg, F., Alizadeh, M., et al. (2024). How negative media coverage impacts platform governance: Evidence from facebook, twitter, and youtube. *Political Communication*, 1–19.

[5] Kumar, S., Kumar, S., Singh, S. R. (2024). A Headline-Centric Graph-Based Dual Context Matching Approach for Incongruent News Detection. *IEEE Transactions on Computational Social Systems*, *11*(5), 5913–5924.

[6] Kumar, S., Jaiswal, R., Sharma, M. R., Singh, S. R. (2023). Multiset Dual Summarization for Incongruent News Article Detection. *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*, 779–790.

[7] Pandey, S., Prabhakaran, S., Reddy, N. S., Acharya, D. (2022). Fake news detection from online media using machine learning classifiers//*Journal of Physics: Conference Series*. volume 2161. IOP Publishing.

[8] Amin, N. (2020). Badda lynching: Renu was attacked by mob hearing riya's screams. *The Business Standard*.

[9] Udoy, A. I., Rahaman, M. A., Islam, M. J., Rahman, A., Ali, Z., et al. (2023). 4sqr-code: A 4-state qr code generation model for increasing data storing capacity in the digital twin framework. *Journal of Advanced Research*.

[10] Allen, J., Watts, D. J., Rand, D. G. (2024). Quantifying the impact of misinformation and vaccine-skeptical content on facebook. *Science*, *384*, eadk3451.

[11] Shaha, P., Khan, M. S. I., Rahman, A., Hossain, M. M., Mammun, G. M., et al. (2024). A prevalent model-based on machine learning for identifying drdos attacks through features optimization technique. *Statistics, Optimization & Information Computing*.

[12] Lahby, M., Aqil, S., Yafooz, W. M., Abakarim, Y. (2022). Online fake news detection using machine learning techniques: A systematic mapping study, 3–37.

[13] Madani, M., Motameni, H., Roshani, R. (2024). Fake news detection using feature extraction, natural language processing, curriculum learning, and deep learning. *International Journal of Information Technology & Decision Making*, *23*, 1063–1098.

[14] Abualigah, L., Al-Ajlouni, Y. Y., Daoud, M. S., Altalhi, M., Migdady, H. (2024). Fake news detection using recurrent neural network based on bidirectional lstm and glove. *Social Network Analysis and Mining*, *14*, 40.

[15] Dev, D. G., Bhatnagar, V., Bhati, B. S., Gupta, M., Nanthaamornphong, A. (2024). Lstmcnn: A hybrid machine learning model to unmask fake news. *Heliyon*, *10*.

[16] M. S. H. Rabbi, M. M. Bari, T. Debnath, A. Rahman, A. K. Das, M. P. Hossain, and G. Muhammad, "Performance evaluation of optimal ensemble learning approaches with PCA and LDA-based feature extraction for heart disease prediction," *Biomedical Signal Processing and Control*, vol. 101, p. 107138, 2025.

[17] Mallik, A., Kumar, S. (2024). Word2vec and lstm based deep learning technique for context-free fake news detection. *Multimedia Tools and Applications*, *83*, 919–940.

[18] Mahmud, T., Hasan, I., Aziz, M. T., Rahman, T., Hossain, M. S., et al. (2024). Enhanced fake news detection through the fusion of deep learning and repeat vector representations//*2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE.

[19] S. H. Mahir, M. T. A. Tashrif, M. A. Karim, D. Kundu, A. Rahman, M. A. Hamza, F. Al Farid, A. S. M. Miah, and S. Mansor, "Advanced Hydro-Informatic Modeling through Feedforward Neural Network, Federated Learning, and Explainable AI for Enhancing Flood Prediction," *IEEE Open Journal of the Computer Society*, 2025.

[20] Ghamdi, M. A. A., Bhatti, M. S., Saeed, A., Gillani, Z., Almotiri, S. H. (2024). A fusion of bert, machine learning and manual approach for fake news detection. *Multimedia Tools and Applications*, *83*, 30095–30112.

[21] Kumar, S., Kumar, G., Singh, S. R. (2022). Text_Minor at CheckThat!-2022: Fake News Article Detection Using RoBERT. *CLEF (Working Notes)*.

[22] Kumar, S., Kumar, G., Singh, S. R. (2022). Detecting incongruent news articles using multi-head attention dual summarization. *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

[23] Asha, V., Nirmala, A., Raj, A. A., Yadav, A., Sen, A., et al. (2024). An effective assessment of machine learning approaches for fake news detection//*2024 International Conference on Inventive Computation Technologies (ICICT)*. IEEE.

[24] Holla, L., Kavitha, K. (2024). An improved fake news detection model using hybrid time frequency-inverse document frequency for feature extraction and adaboost ensemble model as a classifier. *Journal of Advances in Information Technology*, *15*, 202–211.

[25] Choudhury, D., Acharjee, T. (2023). A novel approach to fake news detection in social networks using genetic algorithm applying machine learning classifiers. *Multimedia Tools and Applications*, *82*, 9029–9045.

[26] Altheneyan, A., Alhadlaq, A. (2023). Big data ml-based fake news detection using distributed learning. *IEEE Access*, *11*, 29447–29463.

[27] Jain, M. K., Gopalani, D., Meena, Y. K. (2024). Confake: Fake news identification using content based features. *Multimedia Tools and Applications*, *83*, 8729–8755.

[28] Asha, J., Meenakowshalya, A. (2021). Fake news detection using n-gram analysis and machine learning algorithms. *Journal of Mobile Computing, Communications & Mobile Networks*, *8*, 33–43.

[29] Mugdha, S. B. S., Ferdous, S. M., Fahmin, A. (2020). Evaluating machine learning algorithms for bengali fake news detection//*2020 23rd International*

Conference on Computer and Information Technology (ICCIT). IEEE.
[30] Hussain, M. G., Hasan, M. R., Rahman, M., Protim, J., Hasan, S. A. (2020). Detection of bangla fake news using mnb and svm classifier//*2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*. IEEE.
[31] Coelho, S., Hegde, A., Kavya, G., Shashirekha, H. L. (2023). Mucs@dravidianlangtech2023: Malayalam fake news detection using machine learning approach//*Proceedings of the Third Workshop on Speech and Language Technologies for Dravidian Languages*.
[32] Farooq, M. S., Naseem, A., Rustam, F., Ashraf, I. (2023). Fake news detection in urdu language using machine learning. *PeerJ Computer Science*, *9*, e1353.
[33] Balshetwar, S. V., Rs, A. (2023). Fake news detection in social media based on sentiment analysis using classifier techniques. *Multimedia Tools and Applications*, *82*, 35781–35811.
[34] Pal, A., Pradhan, M. (2023). Survey of fake news detection using machine intelligence approach. *Data & Knowledge Engineering*, *144*, 102118.
[35] Alghamdi, J., Lin, Y., Luo, S. (2023). Towards covid-19 fake news detection using transformer-based models. *Knowledge-Based Systems*, *274*, 110642.
[36] Salh, D. A., Nabi, R. M. (2023). Kurdish fake news detection based on machine learning approaches. *Passer Journal of Basic and Applied Sciences*, *5*, 262–271.
[37] Azzeh, M., Qusef, A., Alabboushi, O. (2025). Arabic fake news detection in social media context using word embeddings and pre-trained transformers. *Arabian Journal for Science and Engineering*, *50*(*2*), 923–936.
[38] Khalil, M., Zhang, C., Ye, Z., Zhang, P. (2025). Pegasosqsvm: A quantum machine learning approach for accurate fake news detection. *Applied Artificial Intelligence*, *39*(*1*), 2457207.
[39] Hossain, M. Z., Rahman, M. A., Islam, M. S., & Kar, S. (2020). BanFakeNews: A Dataset for Detecting Fake News in Bangla. In *Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 2862–2871). European Language Resources Association, Marseille, France.
[40] Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242. Citeseer.
[41] Sharma, U., Saran, S., Patil, S. M. (2020). Fake news detection using machine learning algorithms. *International Journal of Creative Research Thoughts (IJCRT)*, *8*, 509–518.
[42] Muhammad, P. F., Kusumaningrum, R., Wibowo, A. (2021). Sentiment analysis using word2vec and long short-term memory (lstm) for indonesian hotel reviews//*Procedia Computer Science*. volume 179.
[43] Kapusta, J., Držík, D., Šteflovič, K., Nagy, K. S. (2024). Text data augmentation techniques for word embeddings in fake news classification. *IEEE Access*, *12*, 31538–31550.
[44] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
[45] Devlin, J. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, arXiv:1810.04805. https://arxiv.org/abs/1810.04805
[46] Hossen, M. S., Shaha, P., Saiduzzaman, M. (2025). Social Media Sentiments Analysis on the July Revolution in Bangladesh: A Hybrid Transformer Based Machine Learning Approach//*17th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE.

[47] Suthaharan, S. (2016). Support vector machine. *//Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. 207–235.
[48] Adib, Q. A. R., Mehedi, M. H. K., et al., M. S. S. (2021). A deep hybrid learning approach to detect bangla fake news//*2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE.
[49] Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.
[50] Alhussan, A. A., Khafaga, D. S., El-Kenawy, E. M., Ibrahim, A., Eid, M. M., Abdelhamid, A. A. (2022). Pothole and plain road classification using adaptive mutation dipper throated optimization and transfer learning for self-driving cars. *IEEE Access*, *10*, 84188–84211. doi:10.1109/ACCESS.2022.3196660
[51] Song, Y.-Y., Ying, L. (2015). Decision tree methods: Applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*, 130.
[52] Kaur, S., Kumar, P., Kumaraguru, P. (2020). Automating fake news detection system using multi-level voting model. *Soft Computing*, *24*, 9049–9069.
[53] Jr, D. W. H., Lemeshow, S., Sturdivant, R. X. (2013). *Applied Logistic Regression*. John Wiley & Sons.
[54] Mohamed, A., Mahmood, S. (2025). K-Nearest Neighbors Approach to Analyze and Predict Air Quality in Delhi. *Journal of Artificial Intelligence and Metaheuristics*, *9*(1), 34–43. doi:10.54216/JAIM.090104
[55] Jain, P., Sharma, S., et al., P. K. A. (2022). Classifying fake news detection using svm, naive bayes and lstm//*2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE.
[56] Shaikh, J., Patil, R. (2020). Fake news detection using machine learning//*2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*. IEEE.
[57] Akhter, M. P., Zheng, J., et al., F. A. (2021). Supervised ensemble learning methods towards automatically filtering urdu fake news within social media. *PeerJ Computer Science*, *7*, e425.
[58] Hossain, M. M., Chowdhury, Z. R., et al., S. M. R. H. A. (2023). Crime text classification and drug modeling from bengali news articles: A transformer network-based deep learning approach//*2023 26th International Conference on Computer and Information Technology (ICCIT)*. IEEE.
[59] Akhter, M. P., Zheng, J., Afzal, F., Lin, H., Riaz, S., et al. (2021). Supervised ensemble learning methods towards automatically filtering urdu fake news within social media. *PeerJ Computer Science*, *7*, e425.
[60] El-kenawy, E. M. (2025). A Review of Machine Learning Models for Predicting Air Quality in Urban Areas. *Metaheuristic Optimization Review*, *3*(2), 33–46.
[61] Hossain, M. M., Chowdhury, Z. R., Akib, S. M. R. H., Ahmed, M. S., Hossain, M. M., et al. (2023). Crime text classification and drug modeling from bengali news articles: A transformer network-based deep learning approach//*2023 26th International Conference on Computer and Information Technology (ICCIT)*. IEEE.
[62] Rahman, A., Hossain, M. S., Muhammad, G., Kundu, D., Debnath, T., et al. (2023). Federated learning-based ai approaches in smart healthcare: concepts, taxonomies, challenges and open issues. *Cluster Computing*, *26*(*4*), 2271–2311.
[63] Sraboni, T., Uddin, M. R., Shahriar, F., Rizon, R. A., Polock, S. I. S. (2021). *FakeDetect: Bangla fake news detection model based on different machine learning classifiers*. Ph.D. thesis, Brac University.
[64] Kumar, S., Kumar, D., Singh, S. R. (2023). Gated recursive and sequential deep hierarchical encoding for detecting incongruent news articles. *IEEE*

Transactions on Computational Social Systems, *11*(1), 1023–1034.

[65] Tasnim, A., Saiduzzaman, M., Rahman, M. A., Akhter, J., Rahaman, A. S. M. M. (2022). Performance evaluation of multiple classifiers for predicting fake news. *Journal of Computer and Communications*, *10*, 1–21.

[66] Kaliyar, R. K., Goswami, A., Narang, P. (2019). Multiclass fake news detection using ensemble machine learning//*2019 IEEE 9th International Conference on Advanced Computing (IACC)*. IEEE.

[67] Kumar, S., Agrahari, S., Soni, P., Sachdeva, A., Singh, S. R. (2025). Fake news detection using Hashtag context. *Pattern Recognition Letters*.

[68] Chen, T. (2015). *XGBoost: Extreme Gradient Boosting*. R Package Version 0.4-2.

[69] Freund, Y., Schapire, R. E. (1996). Experiments with a new boosting algorithm//*Proceedings of the 13th International Conference on Machine Learning (ICML)*. Citeseer.

[70] Salamai, A. A., El-kenawy, E. M., Ibrahim, A. (2021). Dynamic Voting Classifier for Risk Identification in Supply Chain 4.0. *Computers, Materials & Continua*, *69*(3), 3749–3766. doi:10.32604/cmc.2021.018179

[71] Kumar, S., Kumar, M., Singh, S. R. (2023). Language-Independent Fake News Detection over Social Media Networks Using the Centrality-Aware Graph Convolution Network. *International Conference on Mathematics and Computing*, 89–97, Singapore: Springer Nature Singapore.

[72] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*.

[73] Elsaeed, E., Ouda, O., Elmogy, M. M., Atwan, A., El-Daydamony, E. (2021). Detecting fake news in social media using voting classifier. *IEEE Access*, *9*, 161909–161925.