Review of Computer Engineering Research
2026 Vol. 13, No. 1, pp. 1-35
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/76.v13i1.4709

Comparative analysis of machine and deep learning models with text embeddings for sentiment analysis

Monika Verma[1+]
Rajkumar Jain[2]
Sandeep Monga[3]

[1,2]*Department of Computer Science and Engineering, Oriental University, Indore, Madhya Pradesh, India.*
[1]*Email: monikaverma03@rediff.com*
[2]*Email: rajjain.ce@gmail.com*
[3]*School of Computing Science and Engineering, VIT Bhopal University, Kothrikalan, Sehore, Madhya Pradesh, India.*
[3]*Email: smonga6@gmail.com*

(+ Corresponding author)

Article History
Received: 7 August 2025
Revised: 12 November 2025
Accepted: 28 December 2025
Published: 15 January 2026

Keywords
Fast text
Long short-term memory
Naïve bayes
Random forest
Sentiment analysis
Support vector machine
Text classification
TF-IDF
Word2Vec.

ABSTRACT

This study presents a comprehensive comparative evaluation of traditional machine learning (ML) algorithms Naïve Bayes, Random Forest, and Support Vector Machine (SVM) against a deep learning model, Long Short-Term Memory (LSTM), using three distinct text embedding techniques: Term Frequency-Inverse Document Frequency (TF-IDF), FastText, and Word2Vec. A dataset comprising 30,001 social media posts was employed to assess performance across multiple evaluation metrics, including accuracy, precision, recall, F1-score, ROC-AUC, and log loss. Experimental findings reveal that the combination of LSTM with Word2Vec embeddings achieves superior performance, recording an accuracy of 92.65%, an F1-score of 94.37%, a ROC-AUC of 95.70%, and the lowest log loss value of 0.2074. Among the classical machine learning models, Random Forest emerged as the most effective, outperforming Naïve Bayes and SVM in terms of balanced accuracy and generalization capability. The results underscore the pivotal influence of embedding representation in sentiment analysis and demonstrate that deep learning models, when integrated with semantically rich embeddings, can effectively capture contextual dependencies within textual data. The study thus provides valuable insights into developing robust sentiment analysis frameworks and recommends future exploration of hybrid and ensemble learning approaches to enhance generalization and interpretability in real-world natural language processing applications.

Contribution/Originality: This study contributes to the existing literature on sentiment analysis by conducting a comparative evaluation of machine learning and deep learning models with multiple embedding techniques. It documents the superior performance of LSTM with Word2Vec and highlights how embedding choice significantly impacts classification. The study suggests hybrid and ensemble extensions.

1. INTRODUCTION

Sentiment analysis, an essential task in natural language processing (NLP), focuses on detecting and categorizing emotions and opinions expressed in text [1]. With the rapid increase of user-generated content across platforms such as social media, product reviews, and online forums, sentiment analysis has gained importance in domains like market research, customer feedback monitoring, and business intelligence. Despite its wide applicability, two central challenges persist: selecting classification models that generalize effectively across diverse datasets and designing text representations that capture linguistic nuances [2].

Traditional machine learning methods, including Naïve Bayes, Random Forest, and Support Vector Machines (SVM), remain widely used due to their efficiency and interpretability [3, 4]. However, these models often struggle

to capture contextual and sequential dependencies in text. Deep learning approaches, particularly Long Short-Term Memory (LSTM) networks, have shown superior performance in modeling temporal dependencies, thereby addressing these limitations [5]. Equally critical is the choice of embedding techniques for feature representation, as embeddings strongly influence model accuracy and robustness [6].

In this study, three embedding techniques—TF-IDF, Word2Vec, and FastText are evaluated with four models: Naïve Bayes, Random Forest, SVM, and LSTM. Each embedding captures unique aspects of textual semantics: TF-IDF emphasizes word frequency and importance, Word2Vec captures semantic similarity using neural embeddings [7], and FastText extends Word2Vec by incorporating subword information, improving performance on rare and unseen words. The primary objective of this work is to conduct a comparative evaluation of these models and embeddings using performance metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and log loss. Experimental results demonstrate that LSTM combined with Word2Vec yields the best performance, achieving 92.65% accuracy and a 94.37% F1-score, while Random Forest outperformed other classical classifiers. These findings underscore the importance of selecting suitable embeddings and models for effective sentiment classification.

The remainder of this paper is organized as follows: Section 2 reviews existing machine learning and deep learning approaches to sentiment analysis. Section 3 outlines the methodology, including models, embeddings, and evaluation metrics. Section 4 describes the experimental setup and implementation. Section 5 presents and analyzes results, and Section 6 concludes with key findings and future directions, including hybrid and ensemble extensions.

2. LITERATURE REVIEW

By analyzing customer sentiment, companies can extract meaningful insights that help in shaping strategies and improving overall customer experience. Several studies have experimented with different sentiment categorization models for customer reviews using machine learning algorithms such as Logistic Regression, Random Forest, SVM, LSTM, and BERT. Notably, the innovative sequential mechanisms of BERT achieved 94.2% accuracy with an AUC-ROC of 0.97, demonstrating its ability to capture complex semantic patterns better than competing models. These findings emphasize the scalability of sentiment analysis but also highlight challenges such as noisy data, computational costs, and ethical concerns [1]. In addition to digital text, some research has extended sentiment analysis to physical documents by employing handwriting-to-text conversion (HTC). Using ML and DL methods, studies on datasets such as Twitter, Kaggle, IAM, and Amazon reviews report over 90% accuracy in polarity and emotion classification with the proposed ESIHE AML model, demonstrating applicability in modern communication [2]. Similarly, the integration of sentiment information with econometric models has been tested for economic forecasting. For example, combining sentiment analysis with ARDL models improved inflation prediction in Romania, outperforming SARIMA and conventional models [3].

Although sentiment analysis has been widely studied in English, research in other languages remains limited. One study on Arabic cyberbullying detection used TF-IDF features with SVM, Naïve Bayes, Random Forest, and XGBoost, where Random Forest achieved 80.9% accuracy, underscoring the potential and challenges of extending sentiment analysis to low-resource languages [4]. The impact of Industry 4.0 on sustainability has also been explored through sentiment analysis applied to Scopus abstracts, highlighting benefits for resource efficiency, environmental protection, and economic growth [5].

Hybrid methods continue to attract attention. For instance, an intent sentiment analysis model combining Multinomial Naïve Bayes, SMOTE, and XGBoost classified tweets on the Village Fund initiative with 96% accuracy, precision, recall, and F1-score, confirming the effectiveness of ensemble approaches [6]. Sentiment analysis is also applied to Speech Emotion Recognition (SER). A multilingual SER model using MFCCs with Random Forest and XGBoost achieved 91.25% accuracy on Urdu data, demonstrating promise for multilingual human–machine interaction [7]. In recommendation systems, sentiment-aware book recommendations have been enhanced through hybrid deep learning architectures combining BERT embeddings, TextBlob sentiment tagging, and ensemble models

of LSTM, BiLSTM, GRU, and CNN. The model achieved an accuracy and F1-score of 98.21%, demonstrating the role of sentiment in improving personalization [8]. Similarly, comparative studies on classifiers (Naïve Bayes, J48, BFTree, and OneR) applied to Amazon and IMDB datasets highlight the importance of algorithm selection, with OneR achieving 92.34% global accuracy [9].

Beyond commerce, sentiment analysis has informed healthcare prediction. By examining global data from CDC, WHO, and social media platforms, researchers found sentiment trends could aid in epidemic forecasting, including COVID-19, reinforcing the role of sentiment analysis in public health [10]. Financial applications are equally prominent, where SA models combined with sentiment lexicons were used to predict stock market fluctuations between 2018 and 2023, revealing that investor mood significantly influences market trends [11].

The exponential rise in e-commerce and social media content has accelerated the development of sentiment analysis techniques, spanning from traditional machine learning approaches to advanced transformer models such as GPT, BERT, and T5. These comparative studies reveal both strengths and weaknesses of existing methods and chart directions for future advancements [12]. Social media, particularly Twitter, has been a major testing ground for SA techniques, with transformer-based models and hybrid approaches yielding robust performance [13]. However, consistent results depend heavily on preprocessing, feature selection, and transformation, which remain key factors in improving model accuracy [14].

In cybersecurity, sentiment analysis has been applied to SMS classification (ham, spam, and cyber-malicious), improving threat detection capabilities and providing a foundation for AI-enhanced cyber risk evaluation [15]. Hybrid frameworks combining deep learning models (LSTM, CNN, MLP) with classical ML algorithms (Naïve Bayes, Random Forest, Gradient Boosting, KNN, Decision Tree) further demonstrate improvements in accuracy and robustness, with LSTM achieving up to 99% accuracy on Facebook data [16].

Disaster management research has also leveraged sentiment analysis of social media posts (2018–2023). Results show disaster-related conversations are often negative, highlighting the value of real-time multilingual models for effective crisis response [17]. Applications extend into tourism, where SA models tested on Kaggle datasets have ranked algorithms by F1-score, accuracy, and precision, offering actionable insights for businesses and governments [18]. Similarly, sentiment analysis has been extended to regional languages such as Bengali, where TF-IDF features and tree classifiers achieved 92% accuracy, demonstrating the potential for low-resource NLP [19].

Recent hybrid deep learning models combining CNNs with transformer architectures have demonstrated superior performance on benchmark datasets like Sentiment140, IMDb, and Amazon, achieving a precision of 94.3%, a recall of 93.8%, and an AUC of 97.2%. Importantly, interpretability tools such as SHAP and LIME increase transparency for business applications [20]. Other ensemble frameworks integrating CNN, LSTM, BiLSTM, and BERT report 94.95% accuracy on Twitter datasets, reinforcing the effectiveness of combining pre-trained and sequential models [21].

Sentiment analysis has also been investigated for mental health applications. One study compared lexicon-based, machine learning, and hybrid methods to detect depression in online conversations, noting challenges such as bias and privacy while emphasizing the role of multimodal data [22]. The broader literature additionally documents the use of SA across multiple domains, including marketing, politics, healthcare, and e-commerce, as AI-powered algorithms are increasingly leveraged to transform unstructured text, images, and videos into actionable intelligence [23].

This paper proposes a deep CNN-based model for real-time fake currency note detection, focusing on feature extraction from image patterns. Experimental results demonstrate high accuracy and robustness, highlighting CNN's effectiveness for security and financial applications [24]. Finally, domain-specific sentiment dictionaries, such as those built for Turkish cosmetic product reviews, achieved over 93% accuracy with SVM, highlighting the growing importance of tailored resources for effective sentiment classification [25].

3. METHODOLOGY

3.1. TF-IDF (Term Frequency - Inverse Document Frequency)

Meaning: The total frequency of occurrence (TF-IDF) is a numerical statistic that indicates the significance of a term in a text in comparison to a collection of documents, often known as a corpus. In the fields of text mining and information retrieval, it is often used.

Formula:

$$TF - IDF\ (t,d)\ =\ TF\ (t,d)\ \times\ IDF\ (t) \quad (1)$$

Where, TF (Term Frequency): Assesses the number of times a certain phrase occurs in a given text.

IDF (Inverse Document Frequency): The weight of common words (such as "the," "is," and "and") is reduced in order to determine the significance or uniqueness of a particular vocabulary item.

Purpose:

- Used in search engines to rank documents based on relevance.
- Helps in feature extraction for machine learning models.
- Improves document similarity calculations.

3.2. Algorithm Steps for Text Classification using TF-IDF and Machine Learning Models

The process of text classification using TF-IDF features and machine learning classifiers can be summarized as follows.

Algorithm 1: Text Classification Framework.

Input: Raw textual dataset with labels.

Output: Predicted class labels and performance metrics.

1. Data Preparation.

The dataset is loaded and preprocessed to ensure consistency. Textual labels are mapped to binary numerical values ($Real \rightarrow 1, fake \rightarrow 0$). The corpus is then divided into training (80%) and testing (20%) subsets using stratified sampling to preserve class distribution.

2. Feature Representation

A TF-IDF vectorizer is applied to transform textual data into numerical feature vectors. Common English stopwords are removed, and the vocabulary size is limited to 10,000 terms to enhance efficiency while preserving discriminative features. The vectorizer is fitted on the training set and applied to both training and testing data to ensure consistency.

3. Model Training

Three classifiers are considered for evaluation: Multinomial Naïve Bayes, Random Forest (with 100 estimators), and Support Vector Machine with a linear kernel. Each model is trained on the TF-IDF transformed training data and subsequently used to generate predictions on the test data.

4. Model Evaluation

The predictive performance of each classifier is assessed using multiple evaluation metrics, including Accuracy, Precision, Recall, F1-score, ROC-AUC, Log-Loss, and Matthews Correlation Coefficient (MCC). These metrics collectively capture classification accuracy, error trade-offs, discriminatory power, and robustness to class imbalance.

5. Comparative Analysis

Results from all models are compiled into a structured table to facilitate direct comparison. Graphical visualizations, such as bar charts, are employed to highlight differences across models, with annotated values improving the interpretability of the performance gaps.

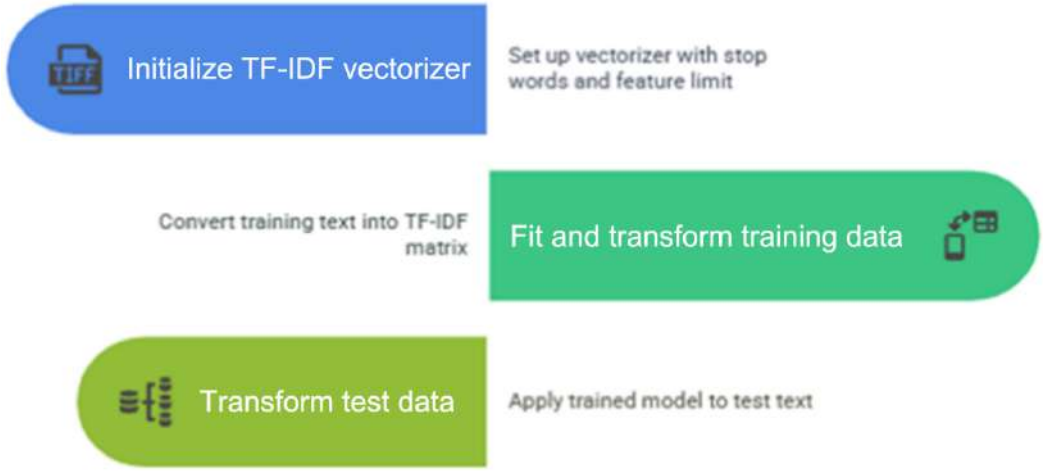

Figure 1. TF-IDF vectorization process consists of three key steps.

There are three main components to the TF-IDF vectorization process shown in Figure 1: The first step is to initialize the TF-IDF vectorizer with parameters such as stop words and feature limits. The second step is to fit and transform the training data into a TF-IDF matrix. The third step is to transform the test data so that the trained model can extract features from it. Machine learning and NLP rely on this procedure to transform textual data into numerical representations.

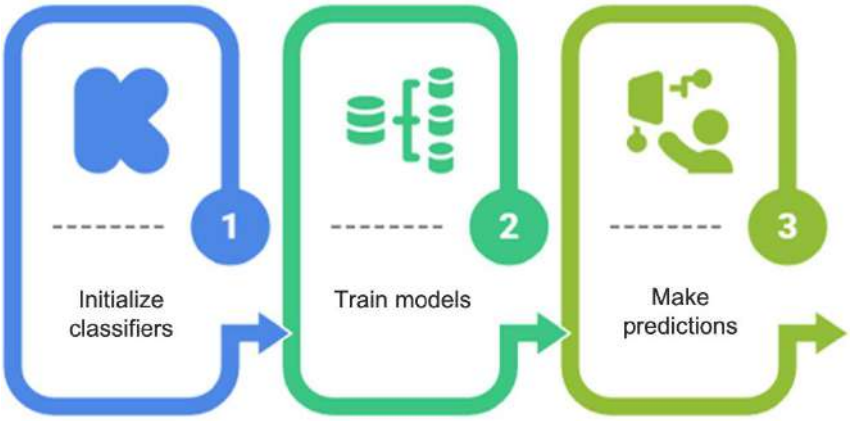

Figure 2. Machine learning model training sequence consists.

In Figure 2, we can see the three main steps that make up the training sequence of a machine learning model: initializing the classifiers, training the model to optimize its parameters using data, and finally, making predictions using the trained model to produce outputs based on fresh input data. The development and deployment of machine learning models are made more efficient using this methodical approach.

3.3. Fast Text

Meaning: The Artificial Intelligence Research team at Facebook created FastText, a word embedding model. When dealing with uncommon or out-of-vocabulary terms, FastText is superior to Word2Vec because it represents words as a collection of character n-grams rather than as atomic entities.

Purpose:

- Generates word embeddings that can handle morphological variations of words.
- Improves text classification, machine translation, and Named Entity Recognition (NER).
- Works well with languages with complex morphology (e.g., German, Russian).

3.4. Algorithm Steps for Text Classification using Fast Text and Machine Learning Models

Algorithm 2: Text Classification with FastText Representations.

Input: Text corpus with labeled instances.

Output: Predicted class labels and performance comparison across classifiers.

1. Dataset Preparation

The dataset, stored in tabular format, is loaded into a data structure (e.g., Pandas DataFrame). The text column (*Ultimate_text*) serves as the feature space, while the *target* column provides class labels. The dataset is divided into training (80%) and testing (20%) subsets using stratified sampling, with a fixed random state for reproducibility.

2. Embedding Training

Fast Text embeddings are trained using the training subset to capture subword-level semantics. The embedding dimension is fixed at 200, with a context window of 15 tokens. Words occurring fewer than five times are excluded to reduce noise, and parallelization across CPU cores ensures efficiency.

3. Feature Representation

Each document is mapped into a dense vector by averaging the FastText embeddings of its constituent words. Documents containing words absent from the vocabulary are assigned a zero vector, ensuring dimensional consistency across all samples.

4. Model Training.

Three classifiers are employed for comparative evaluation.

- Naïve Bayes: A Bernoulli variant applied to binarized feature representations.
- Random Forest: An ensemble of 100 decision trees trained with bootstrap aggregation to reduce variance.
- Support Vector Machine (linear kernel): A margin-based classifier effective for linearly separable feature spaces.

Each model is trained on the FastText-derived training vectors and used to predict class labels on the test set.

5. Performance Evaluation

The predictive ability of each model is assessed using accuracy, precision, and the confusion matrix. These metrics respectively measure overall correctness, the reliability of positive predictions, and the distribution of classification errors across classes.

6. Comparative Analysis

Results are consolidated into tabular form and visualized for clarity. Classifiers are compared based on their relative strengths, and the best-performing approach is identified according to precision-recall trade-offs and overall accuracy.

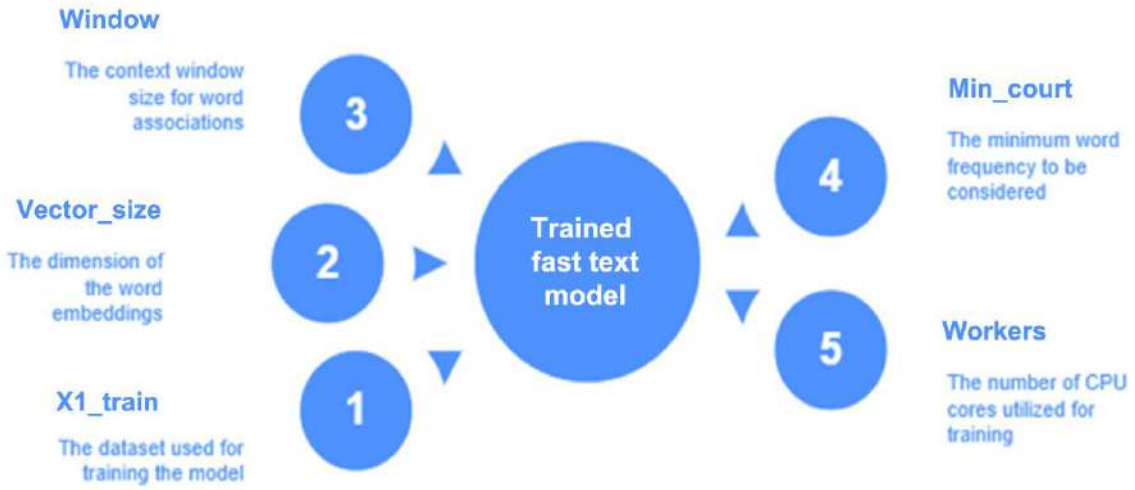

Figure 3. Fast text model process involves five key parameters.

Figure 3 shows parameters X1_train, vector_size, window, min_count, and workers that define various aspects of the FastText model training process. X1_train represents the training dataset, vector_size defines the dimension of word embeddings, min_count sets the minimum word frequency threshold to be considered, and workers indicate the number of CPU cores used for training. Taken as a whole, these settings make FastText the best model for learning word representations quickly and accurately.

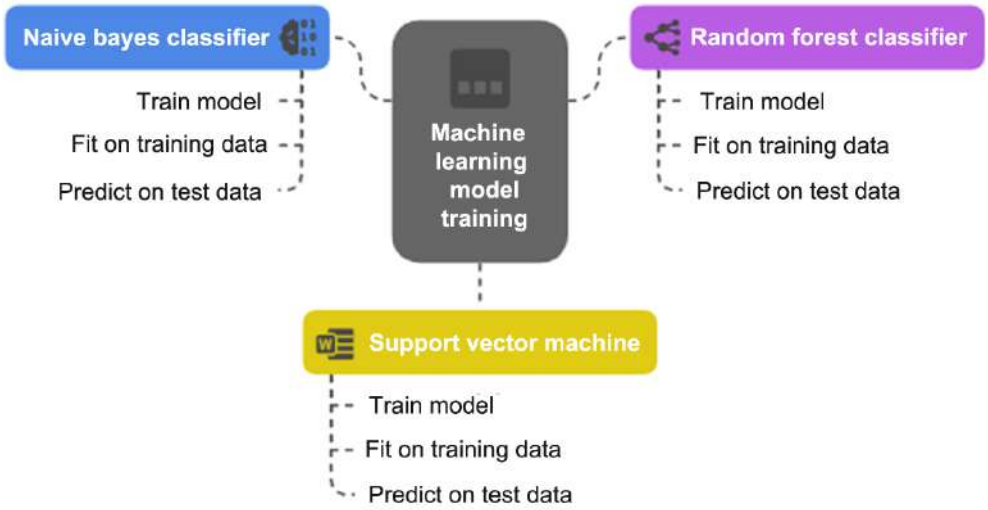

Figure 4. Machine learning model training and prediction process.

A number of different classifiers, including the Naive Bayes Classifier, the Random Forest Classifier, and the SVM, are trained and assessed as part of the process shown in Figure 4 for Machine Learning Model Training and Prediction. Model training, data fitting, and test data prediction are the three processes that each and every classifier must perform in order to be considered complete. To improve the accuracy of predictions and the capacity to generalize them to data that has not yet been observed, these models are implemented as components of a centralized framework for machine learning model training.

3.5. Algorithm Steps for Text Classification using LSTM and Fast Text Word Embeddings

Algorithm 3: LSTM-Based Text Classification Framework.

Input: Preprocessed text corpus and labels.

Output: Predicted class labels and performance evaluation.

1. Dataset Preparation

The dataset is divided into training and testing subsets (e.g., 80/20 split). A tokenizer is fitted on the training text to construct a vocabulary of word indices. Text data are converted into sequences of indices corresponding to tokens, and all sequences are padded or truncated to a fixed maximum length (100 tokens) to ensure uniformity of input.

2. Embedding Initialization

FastText embeddings are trained on the training text or imported from a pre-trained model. Parameters typically include a vector dimension of 100, a context window of 5 tokens, and a minimum word frequency threshold. An embedding matrix is constructed by aligning the tokenizer vocabulary with the FastText embeddings, ensuring that each word index maps to its corresponding vector representation.

3. Model Architecture

An LSTM-based neural network is constructed using a sequential design.

- Embedding Layer: initialized with the FastText embedding matrix, transforming input indices into dense vectors.

- LSTM Layer: Contains 128 hidden units with dropout regularization to capture long-range dependencies and mitigate overfitting.
- Output Layer: A dense layer with sigmoid activation for binary classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the primary evaluation metric.

4. Model Training

The LSTM is trained on the padded training sequences and corresponding labels, using a batch size of 64 for 10 epochs. A validation split of 20% is applied to monitor generalization and prevent overfitting through early stopping criteria.

5. Performance Evaluation

After training, the model is evaluated on the test set using accuracy as the principal performance metric. Additional measures such as precision, recall, F1-score, and ROC-AUC can be computed to provide a comprehensive assessment of classification effectiveness.

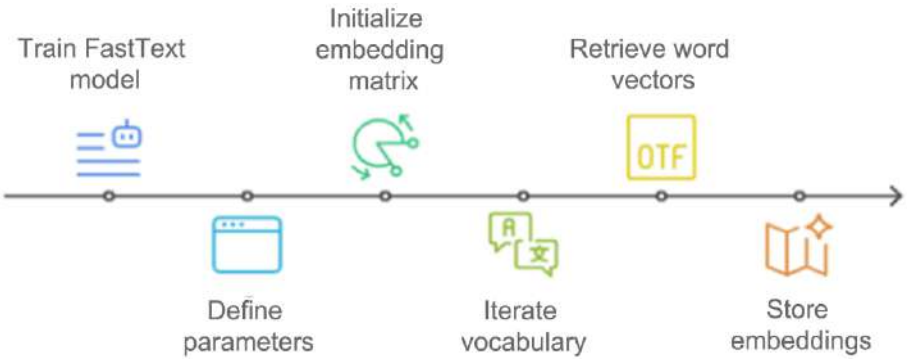

Figure 5. Fast text embedding creation process consists.

The Figure 5 illustrates the five main phases in the creation process of fastText embedding: establishing a foundation for the FastText model through training; defining essential settings in the 'Defining Parameters' section; setting up the embedding matrix, which serves as a framework for organizing word embeddings; performing a vocabulary iteration to confirm processing of every word; storing embeddings for later use; and retrieving learned word vectors. This procedure enables the efficient production of word vectors for natural language processing applications.

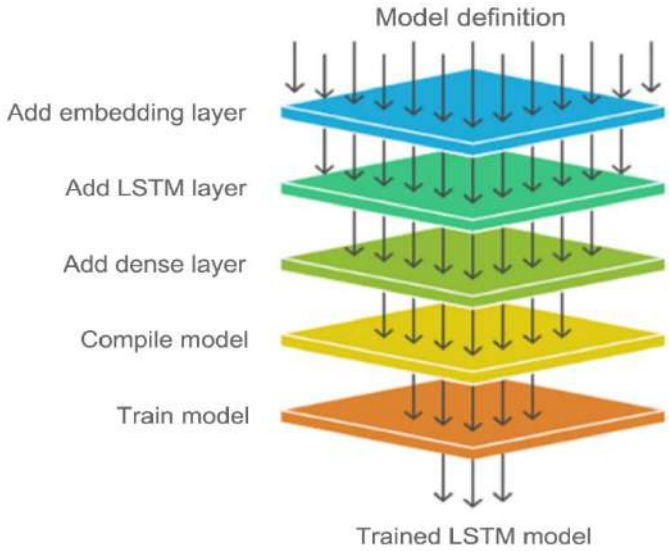

Figure 6. Building and training an LSTM Model process consists.

The process of developing a trained model follows a sequential order, as shown in Figure 6. An Embedding Layer is added to handle word representations after model definition. The next step is to process the output with a Dense Layer, after which an LSTM Layer is added to capture the temporal dependencies. Optimization and loss functions are defined by compiling the model. Finally, the trained LSTM model is prepared for prediction by executing the training process.

3.6. Word2Vec

Meaning: The Word2Vec model is a neural network-based model that was created by Google for the purpose of generating word embeddings. This model maps words with similar meanings to vector representations that are close to each other.

It has two main training approaches:

1. Continuous Bag of Words (CBOW): This model makes predictions about a target word based on the words that are around it.
2. The skip-gram algorithm, which analyzes surrounding words based on a target word.

Purpose:

- Captures semantic relationships between words.
- Used in NLP applications like machine translation, chatbot development, and text classification.
- Helps in recommendation systems and document clustering.

3.7. Algorithm Steps for Fake News Detection using Word2Vec and Machine Learning Models

Algorithm 4: Word2Vec-Based Text Classification Framework.

Preprocessed text corpus with binary labels.

Output: Predicted labels and comparative performance metrics across classifiers.

1. Dataset Preparation

The dataset is imported from a structured file and mapped into binary numerical labels ($Real \rightarrow 1, fake \rightarrow 0$). The data are partitioned into training and testing subsets using an 80:20 split with stratification to maintain class balance.

2. Embedding Initialization

Pre-trained *word2vec-google-news-300* embeddings are loaded from the Gensim repository. Each word in the vocabulary is represented as a 300-dimensional dense vector trained on a large news corpus, capturing both syntactic and semantic information.

3. Feature Representation

Each document is transformed into a fixed-length vector through the following procedure: (i) tokenize the sentence into words, (ii) retrieve embeddings for in-vocabulary tokens, (iii) compute the mean vector across all tokens, and (iv) assign a zero vector when no valid tokens are present. This ensures consistent dimensionality across all instances.

4. Feature Normalization

To improve classifier stability, feature vectors are scaled into the $[0,1]$ range using Min–Max normalization. This step is applied independently to training and testing sets to avoid data leakage.

5. Model Training

Three machine learning classifiers are trained on the Word2Vec representations.

- Naïve Bayes (Multinomial variant): Assumes conditional independence of features and models text likelihoods.
- Random Forest: An ensemble of 100 decision trees trained via bootstrap aggregation, improving robustness to feature noise.

- Support Vector Machine (linear kernel): Optimizes a margin-based decision boundary in high-dimensional space.

6. Performance Evaluation

Each model is assessed on the test set using multiple evaluation measures, including Accuracy, Precision, Recall, F1-score, ROC-AUC, Log-Loss, and Matthews Correlation Coefficient (MCC). Confusion matrices are generated to illustrate misclassification patterns.

7. Comparative Analysis and Visualization

Results are aggregated into a performance table for cross-model comparison. Visualization techniques, including bar charts and heatmaps, are employed to highlight strengths and weaknesses across classifiers and to support the interpretability of the outcomes.

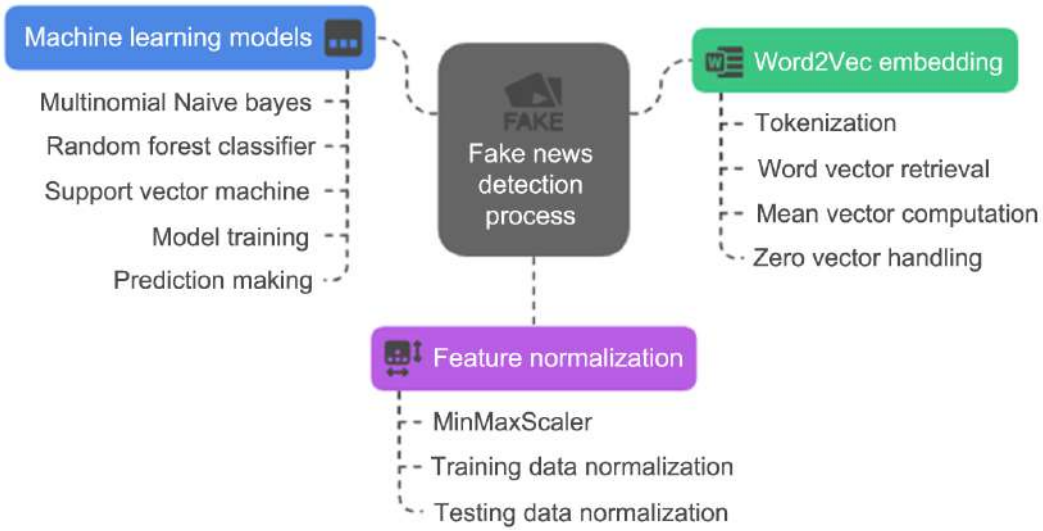

Figure 7. Fake news detection process consists of three main components.

In Figure 7, there are primarily three steps to the process of detecting fake news: using machine learning models, embedding Word2Vec, and normalizing features. Among the taught and utilized machine learning models for prediction are support vector machines, random forest classifiers, and multinomial naive Bayes. Tokenization, word vector retrieval, mean vector calculation, and managing zero vectors are all part of Word2Vec embedding, which quantitatively represents text. To ensure that the model's performance is consistent across testing and training data, feature normalization uses methods like MinMaxScaler to standardize the data. All of these components work together to make false news identification more precise and trustworthy.

3.8. Algorithm Steps for Text Classification using Word2Vec and LSTM

Algorithm 5: Word2Vec–LSTM Text Classification Framework.

Input: Pre-processed text corpus with binary labels.

Output: Predicted labels and evaluation metrics.

1. Dataset Preparation.

The dataset, containing text samples and their corresponding labels, is divided into training and testing subsets (80% and 20%, respectively). A tokenizer is applied to the training text to construct a vocabulary of indexed tokens. Text sequences are generated from the tokenizer and standardized to a fixed maximum length of 100 tokens using padding, ensuring uniform input dimensions.

2. Embedding Initialization.

Word embeddings are generated using the Word2Vec algorithm trained on the training text. The embedding dimension is fixed at 200, with a context window of 15 words and a minimum frequency threshold of five occurrences.

An embedding matrix is constructed with dimensions (V+1)×200, where V is the vocabulary size. Each entry in the matrix corresponds to a token's Word2Vec vector; unseen words are initialized as zero vectors.

3. Model Architecture

An LSTM-based neural network is built using the following components:

- Embedding Layer: Initialized with the Word2Vec embedding matrix to project input tokens into a dense vector space.
- LSTM Layer: Consisting of 128 hidden units, with dropout and recurrent dropout (0.2 each) to improve generalization.
- Output Layer: A dense unit with sigmoid activation for binary classification.

The model is compiled using the Adam optimizer, binary cross-entropy loss, and accuracy as the primary evaluation metric.

4. Model Training

The model is trained for 10 epochs with a batch size of 64. Twenty percent of the training set is reserved for validation to monitor performance and reduce overfitting.

5. Performance Evaluation

The trained model is evaluated on the test data. Classification effectiveness is assessed primarily by accuracy, while additional measures such as Precision, Recall, F1-score, and ROC-AUC may be included for a more comprehensive evaluation.

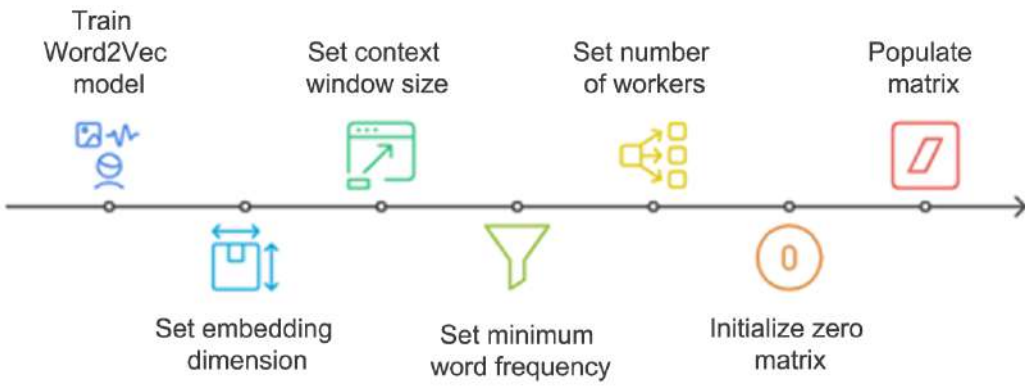

Figure 8. Word2Vec model training and embedding matrix creation process consists.

Several essential processes comprise the process of creating the Word2Vec model training and embedding matrix (Figure 8): the process of teaching word representations using the Word2Vec model; embedding dimension setting, which determines the size of the vector; choosing the size of the context window, which controls the number of nearby words to take into account; establishing a word frequency minimum to ensure only pertinent terms are used; allocating workers to achieve maximum computational efficiency; preparing the structure for embeddings by initializing the zero matrix; and filling up the matrix, which is used for natural language processing applications to store learned word vectors.

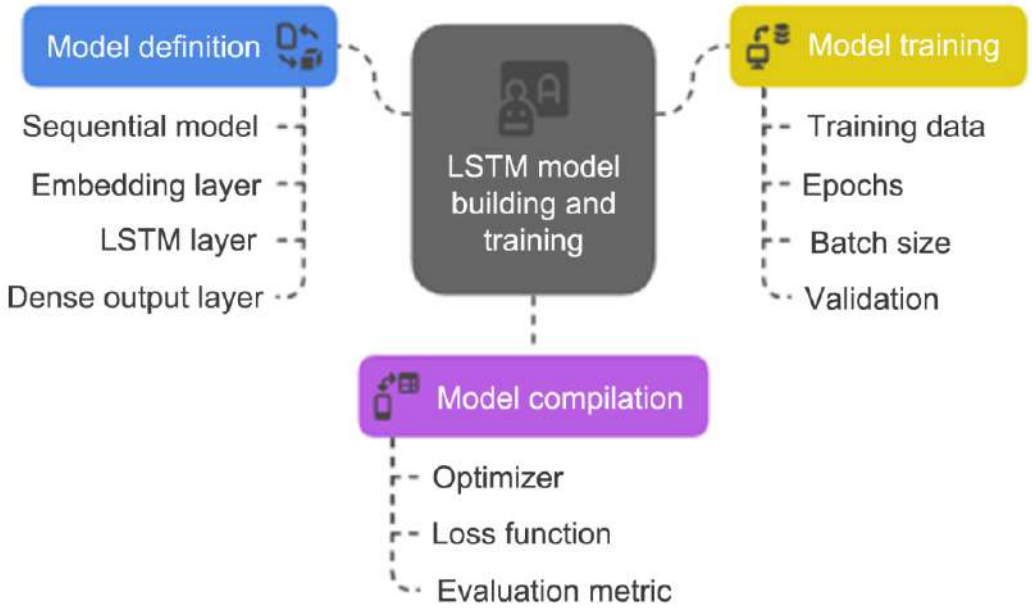

Figure 9. LSTM model building and training process consists of three main stages.

Model Definition, Model Compilation, and Model Training are the three primary steps in the LSTM Model Building and Training Process shown in Figure 9. The LSTM, Dense Output, and Embedding layers make up a Sequential Model in Model Definition, which is how LSTM models are organized. Choosing an Optimizer, creating a Loss Function, and deciding on an Evaluation Metric are all parts of the Model Compilation process. At last, the model is trained using Training Data, and important hyperparameters such as Batch Size, Validation, and Epochs are adjusted to maximize learning and generalization during Model Training.

3.9. Comparison

Tables 1 and 2 provide a comprehensive comparison of text representation methods and classification models used for sentiment analysis. Table 1 highlights the strengths and weaknesses of TF-IDF, FastText, and Word2Vec, where TF-IDF is simple but lacks context, FastText handles rare words effectively, and Word2Vec captures semantic meaning but fails on unseen terms. Table 2 contrasts LSTM, Random Forest, SVM, and Naïve Bayes across dimensions like computational cost, interpretability, training time, and robustness. LSTM excels in handling complex, long texts but is resource-intensive, while Random Forest and SVM offer a balance of speed and accuracy, and Naïve Bayes remains a fast, lightweight baseline. Together, these tables guide the selection of appropriate model-embedding combinations for real-world sentiment analysis tasks.

Table 1. Comparison of different methods.

Method	Type	Strengths	Weaknesses
TF-IDF	Statistical	Good for document relevance, simple to compute.	Ignores context, not good for semantic similarity.
FastText	Deep Learning	Handles rare words, good for complex languages	Requires more training data
Word2Vec	Deep Learning	Captures semantic relationships	Cannot handle out-of-vocabulary words

Table 2. Comparison of methods based on different parameters.

Method	Feature Extraction	Handling Large Data	Computational Cost	Accuracy (General)	Interpretability	Training Time	Inference Time	Works well with imbalanced data	Handles noise well	Hyperparameter Tuning Importance	Memory Usage	Best Use Cases
LSTM	Word Embeddings (Word2Vec/FastText/GloVe)	High (Needs GPU and Large RAM)	Very High (Requires GPUs and TPUs)	High (Requires Large Data)	Low (Difficult to Explain)	Very long (needs more epochs)	Medium (Slower Than Traditional Models)	Yes (With Weighted Loss or Data Augmentation)	Yes (with regularization and attention	High (Batch Size, Learning Rate, Optimizers)	Very High (Requires Large VRAM)	Deep Sentiment Analysis, Context Understanding, Long
Random Forest	TF-IDF / Word Embeddings	Medium (Can be parallelized)	Medium (Parallel Processing Available)	Medium-High (Feature Dependent)	Medium (Feature Importance Available)	Medium (Parallel Training Possible)	Fast (Optimized with Trees)	Yes (Using Class Weights and Sampling)	Yes (Robust to Noisy Features)	Medium (Number of Trees, Max Depth)	Medium (Optimized for Trees)	General Purpose Sentiment Classification, Feature-Rich Datasets
SVM	TF-IDF / Word Embeddings	Medium (Depends on Kernel Selection)	Medium (Linear Kernel is Faster)	Medium-High (With Hyperparameter Tuning)	High (Supports Decision Boundaries)	Medium (Depends on Kernel and Dataset)	Fast (Especially with Linear Kernel)	Yes (With Proper Kernel Selection)	Yes (If Properly Tuned)	High (Kernel Choice, Regularization)	Medium (Depends on Feature Space)	Short Text Classification, High Precision Tasks
Naive Bayes	TF-IDF	High (Handles Large Datasets Efficiently)	Low (Highly Efficient)	Medium (Works Best with Clean Data)	High (Probabilistic Explanation)	Short (Trains Very Fast)	Very Fast (Instant Predictions)	No (Struggles with Class Imbalance)	No (Sensitive to Noisy Inputs)	Low (Few Parameters Needed)	Low (Lightweight Model)	Baseline Sentiment Analysis, Large-Scale Datasets

4. IMPLEMENTATION

4.1. Dataset

The dataset used in this study consists of 30,001 entries distributed across 8 columns, representing both textual posts and their associated metadata. Each row corresponds to a unique post identified by a distinct ID. The dataset includes a "Labels Set" column that categorizes posts into "fake" and "non-fake" classes. Additional engagement metrics such as replies, retweets, and favorites are provided, along with sentiment markers like *Positive_Comments*

and *Negative_Comments*. These features make the dataset suitable for tasks such as sentiment analysis, social media trend analysis, and fake news detection.

To ensure the text was suitable for machine learning models, a preprocessing function was implemented using the Natural Language Toolkit (NLTK). This process involved tokenizing the text, removing punctuation and special characters, and eliminating stopwords. Although the raw data initially contained a small number of non-English tokens and encoding artifacts, these were systematically removed during preprocessing so that only English text was retained. The final cleaned dataset is entirely English, ensuring compatibility with the models and compliance with the journal's requirements.

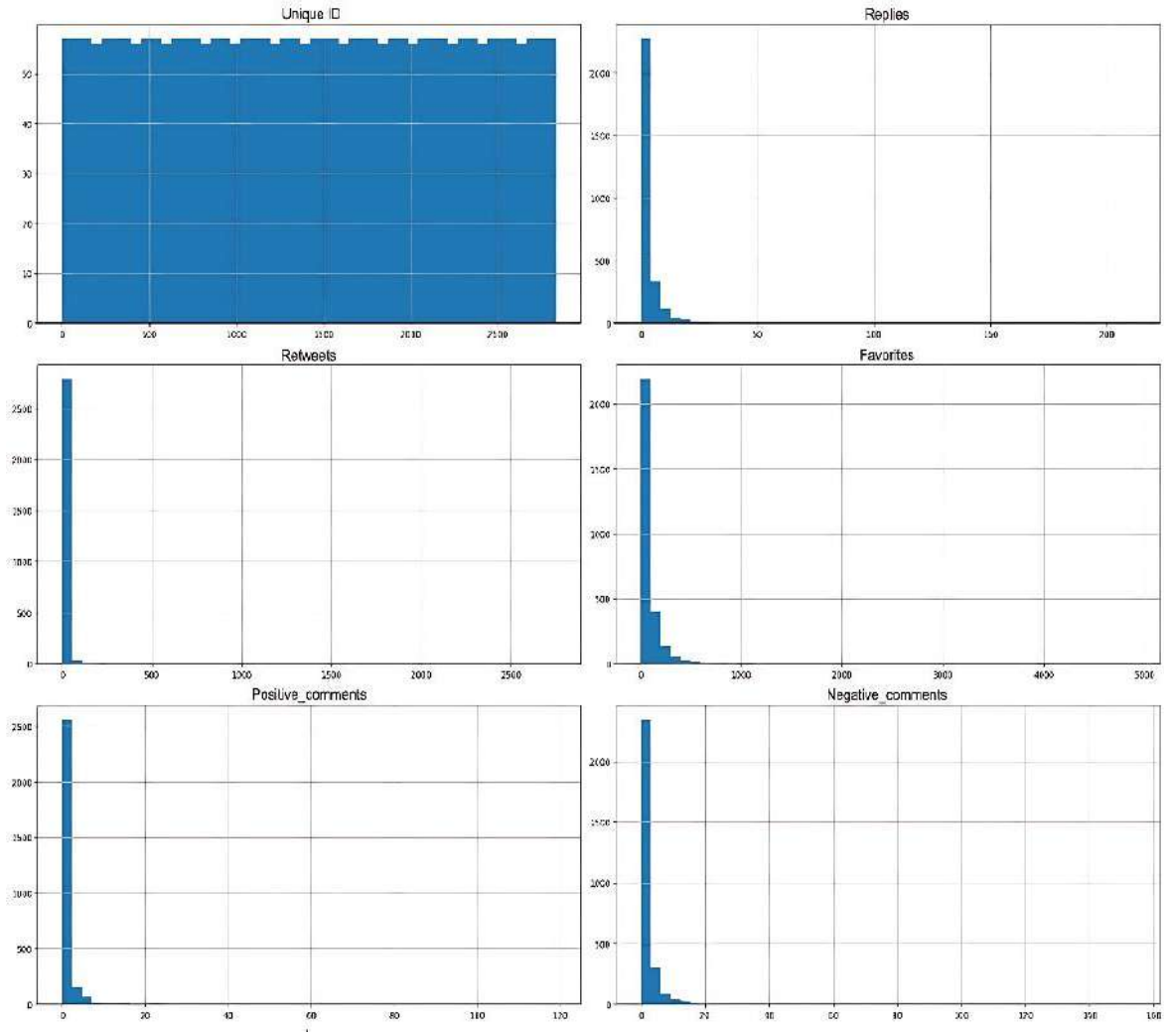

Figure 10. Presents histograms depicting the distribution of different engagement metrics in the dataset.

4.2. Applying TF-IDF Word Embeddings

Figure 10 shows histograms that illustrate how the dataset is distributed across several engagement indicators. A uniform distribution of Unique IDs is observed in the first figure (top-left), indicating an equal representation of entries. Responses, retweets, likes, positive comments, and negative comments all exhibit highly skewed distributions in the other histograms, with most values clustered around zero and a small number of outliers. It appears that most postings do not receive much engagement, but a few garner significant attention, resulting in a long-tail distribution. These findings are essential for machine learning tasks such as feature scaling, model selection, and engagement prediction.

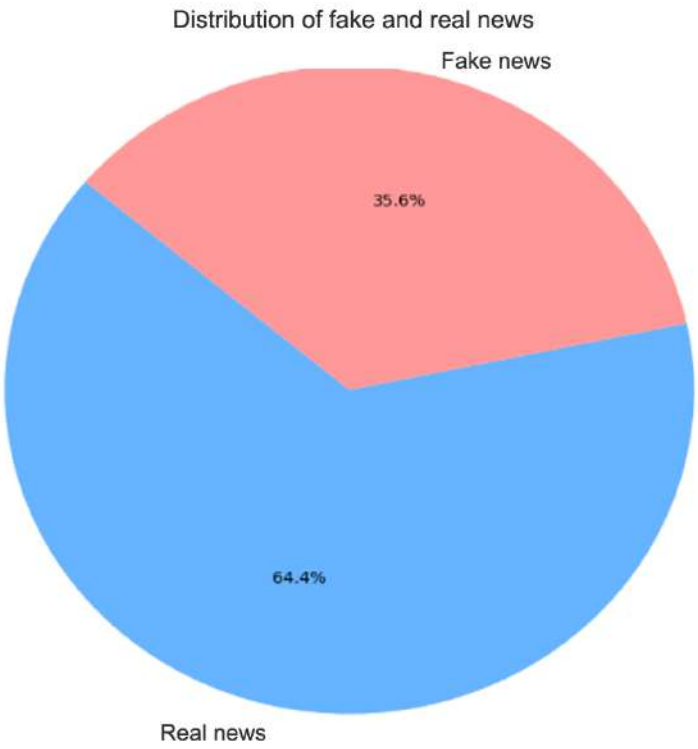

Figure 11. The distribution of fake and real news in the dataset.

Figure 11 shows how the dataset distributes authentic and fraudulent news. An analysis of the news reveals that 64.4% of it is authentic and 35.6% is fraudulent. This suggests that the dataset is skewed, with more instances of actual news than false news. Because of this disparity, machine learning models may be biased in favor of the dominant group, which is problematic. To address this, methods such as resampling, weighted loss functions, or synthetic data generation (like SMOTE) can be considered to improve the model's ability to recognize false news.

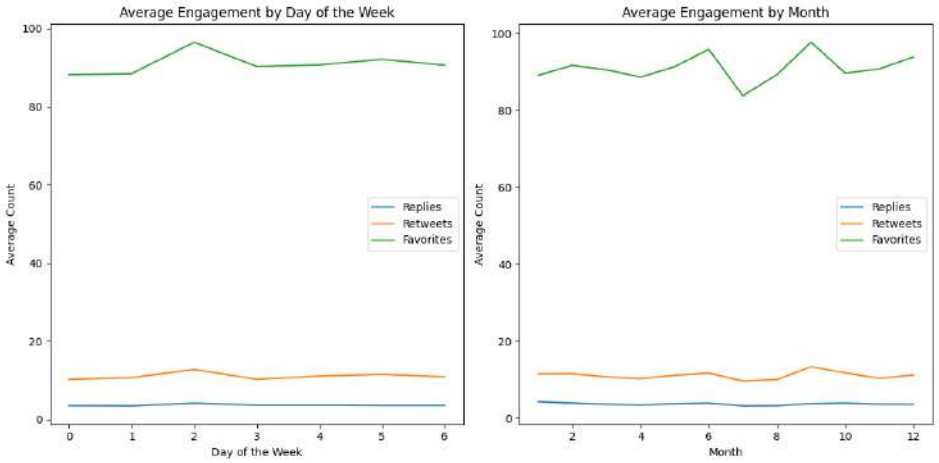

Figure 12. Analyzing average engagement trends.

Two line charts, one for each day of the week and one for each month, examine average engagement trends (Figure 12). The several plots display the changing engagement levels over time through the tracking of responses, retweets, and likes. Likes (green line), retweets (orange line), and responses (blue line) are the three most engaging metrics in both graphs. There appears to be a small variation in involvement, with a high around mid-week, according to the day-of-week chart. The monthly data shows that there are periodic changes, with a decline in involvement observed in certain months. These findings are useful for determining the best times to post in order to get the most engagement from your audience.

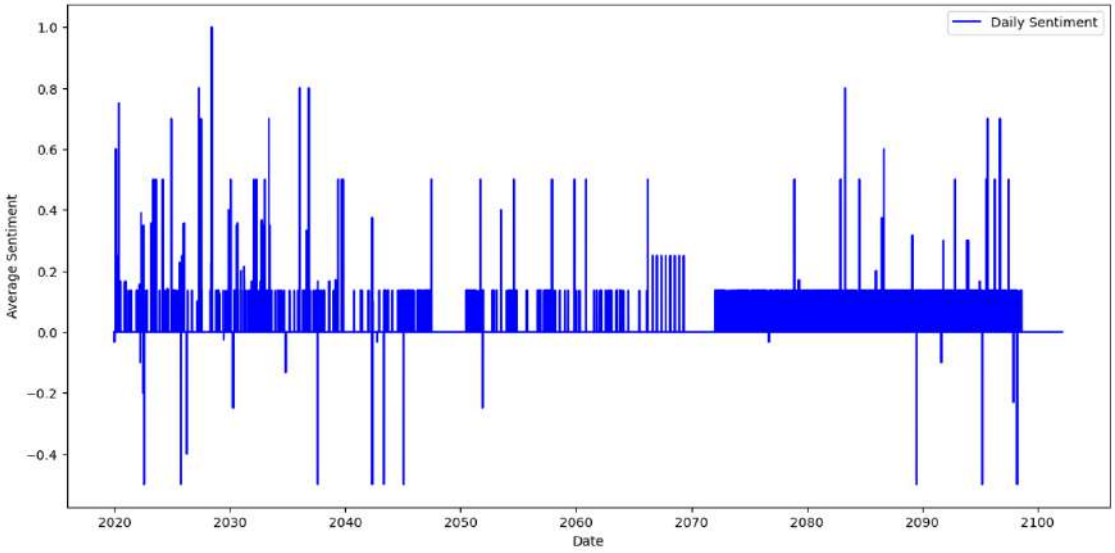

Figure 13. Daily sentiment trends over time.

Figure 13 demonstrates long-term patterns in daily sentiment, with dates represented on the x-axis and average sentiment scores shown on the y-axis. At different time intervals, there are noticeable differences in sentiment, which might be favorable or negative. Positive involvement is shown by periods of high sentiment, whereas spikes in negative sentiment are indicated by dramatic drops. Possible date-related abnormalities or data formatting errors might be the cause of the timeline's irregularity, which includes dates that are impossibly distant in the future, such as 2040-2100. Sentiment trends, possible outliers, and engagement patterns may be better seen with this graphic.

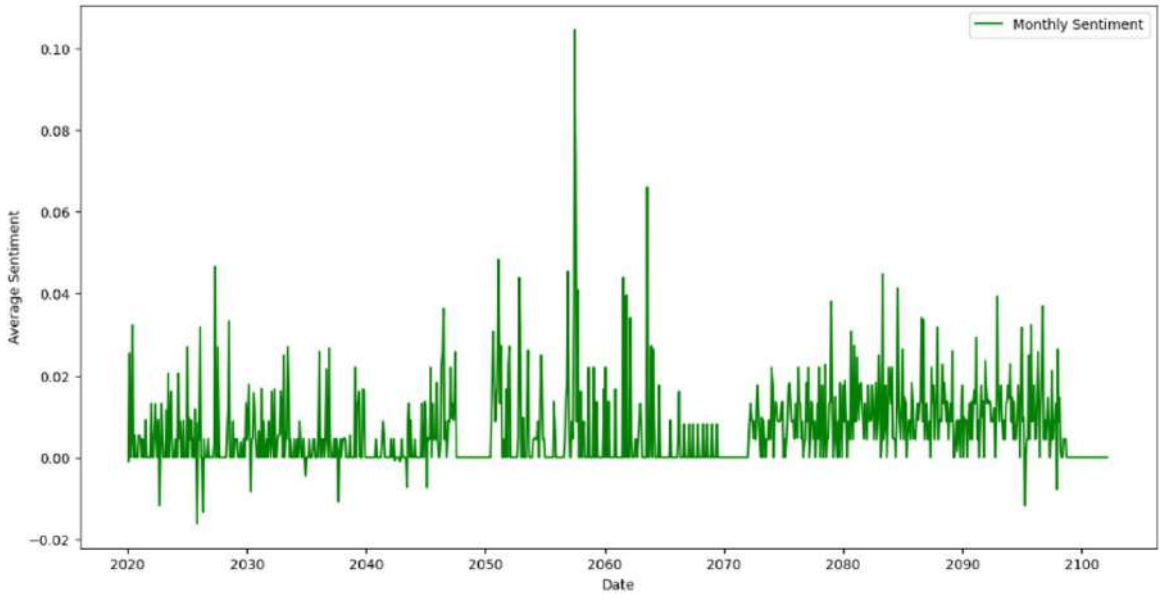

Figure 14. Monthly sentiment trends over time.

Figure 14 is a time series depiction of monthly sentiment patterns; the x-axis displays the date, and the y-axis shows the average sentiment score. Observable peaks and valleys in the sentiment indicate periods of high and low sentiment activity, respectively. Irregular patterns, presumably caused by data errors or outliers, are indicated by a notable peak around 2060 and different oscillations. There may have been formatting mistakes in the data if timestamps with future dates (e.g., 2030-2100) are present. Anomalies, engagement trends, and long-term changes in sentiment can be better understood using this approach.

Figure 15. Word cloud visualization appears to contain garbled text.

Figure 15 indicates that there seems to be an encoding problem in the dataset because the word cloud visualization includes garbled text and special characters (such as à, ¾, ō, ae, and https). When processing non-English text, particularly if the dataset contains Hindi or other Indic languages, this is most often the result of erroneous character encoding. Symbols, accented letters, and bits of URLs predominate over meaningful words in the image. To address this, ensure the word cloud accurately represents the dataset's common terms by applying the correct text decoding (e.g., UTF-8).

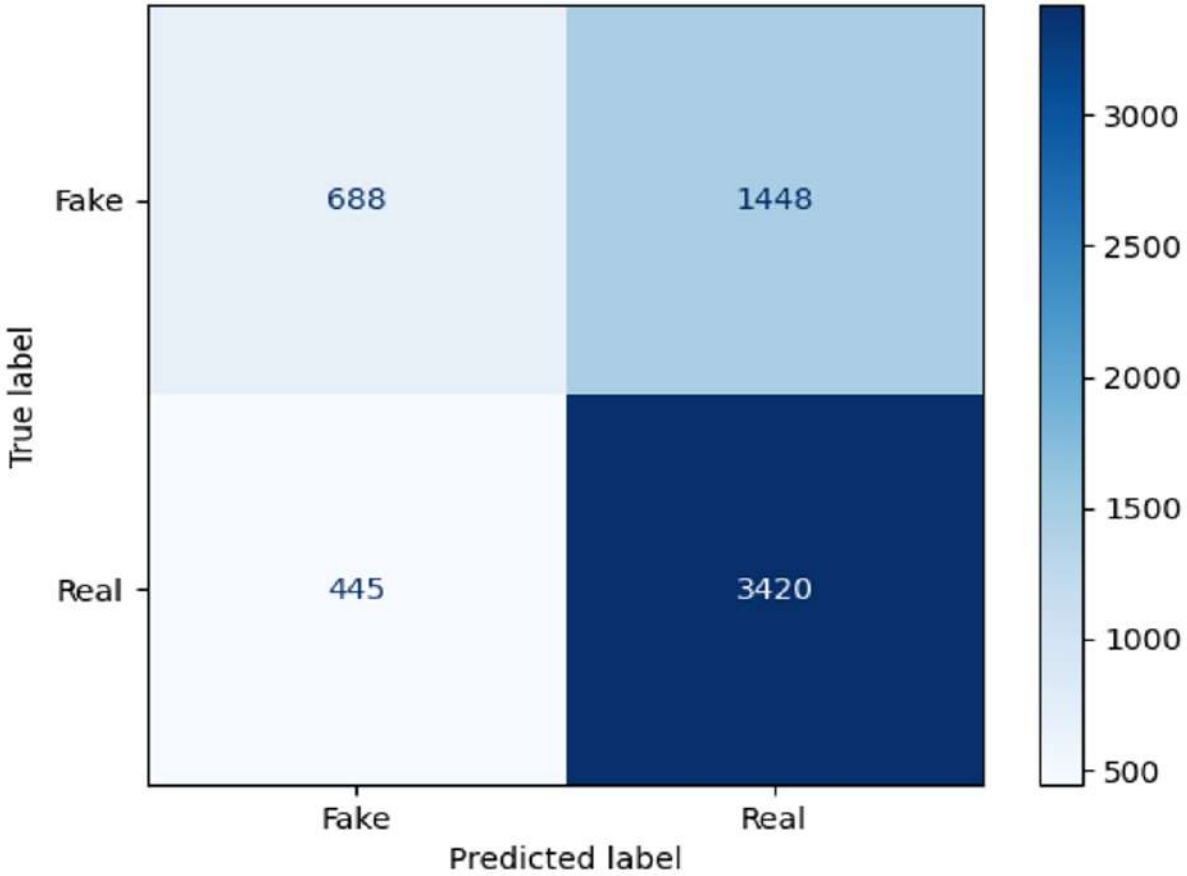

Figure 16. Confusion matrix for Naïve Bayes.

4.3. Fast Text Word Embeddings

Figure 16, which is the Naïve Bayes confusion matrix, assesses how well the model distinguishes between true and fraudulent news. The four essential values that make up the matrix are: 688 instances of genuine news being

accurately categorized as fake, 3,420 instances of real news being correctly labeled as real, 1,485 instances of real news being misclassified as fake, and 1,448 instances of false positives occurring. Naïve Bayes' assumption of feature independence may have constraints or the dataset may be imbalanced, but either way, the large frequency of false positives suggests that the model has trouble accurately identifying bogus news. Model improvement, such as feature engineering, hyperparameter tuning, or ensemble approaches, may be necessary to increase the accuracy of false news detection, despite its good performance in recognizing actual news.

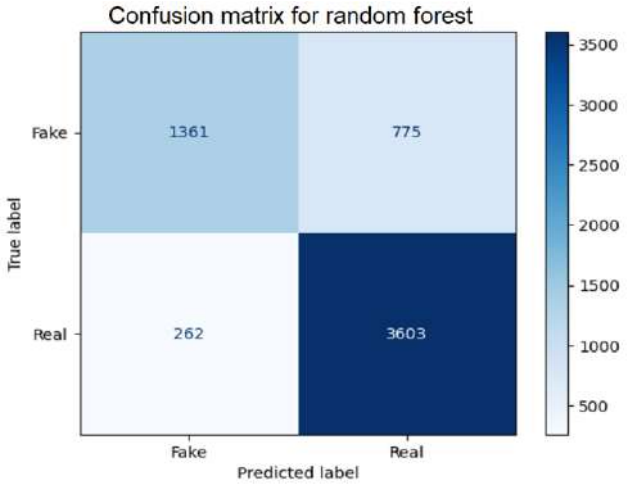

Figure 17. Confusion matrix for random forest classifier.

The Random Forest classifier's capacity to distinguish between true and fraudulent news is assessed in Figure 17, which is the confusion matrix. With a success rate of 1361 for false news and 3603 for actual news, the model has proven itself to be an accurate classifier. A total of 775 false positives and 262 false negatives occur when it incorrectly identifies actual news stories as fake. This model outperforms Naïve Bayes in differentiating between fake and true news, as seen by the reduced rates of false positives and false negatives. When it comes to detecting false news, the Random Forest classifier is the best option because of its higher precision and recall. Additional optimization, such as hyperparameter tuning, feature selection, or ensemble approaches, may be necessary to improve its performance, as shown by the remaining misclassifications.

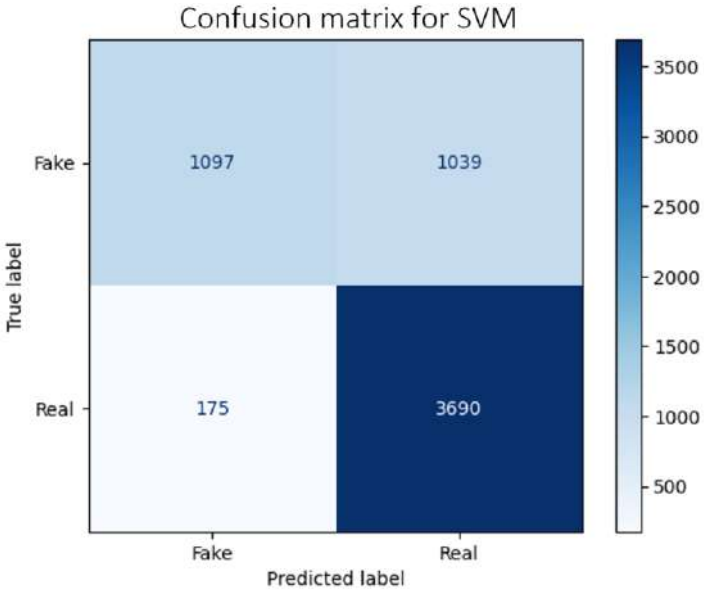

Figure 18. Confusion matrix for SVM classifier.

The SVM classifier's (Figure 18) confusion matrix displays its accuracy in distinguishing between true and fraudulent news. The algorithm has a success rate of 1,097 in identifying false news stories and 3,690 in identifying legitimate news stories. It does, however, mistake 1.75 pieces of legitimate news for false positives and 10.39 pieces of fraudulent news for actual ones. With a lower false negative rate, SVM is better at detecting actual news stories than Random Forest. Having a greater false positive rate (1,039 misclassified bogus news), it appears to have difficulty properly differentiating between fake and authentic news. Further optimization, feature engineering, or different kernels may be necessary to increase the SVM's accuracy in detecting false news, despite its outstanding performance in accurately forecasting actual news.

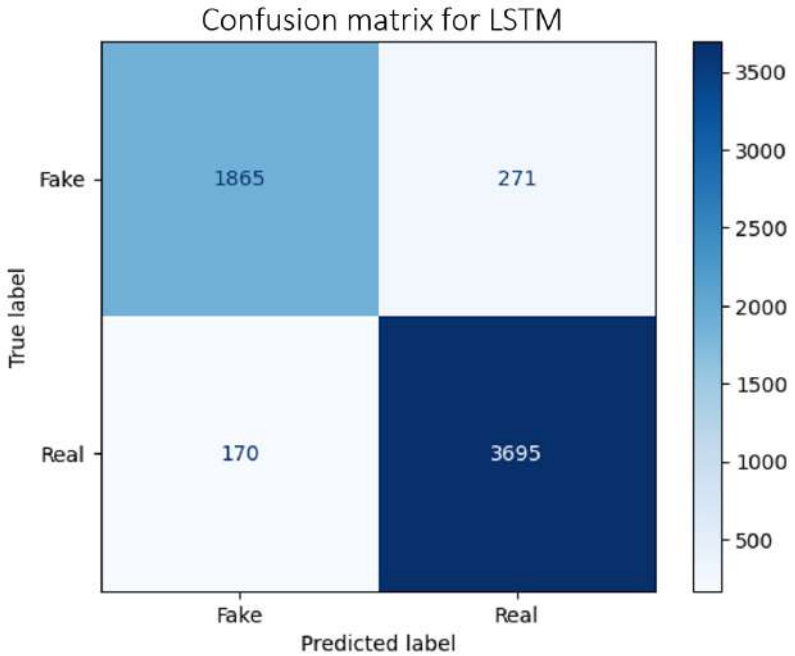

Figure 19. Confusion matrix for the LSTM model.

Figure 19 shows the LSTM model's confusion matrix, which illustrates how effectively it can distinguish between true and fraudulent news. Overall, the model demonstrates a high level of accuracy, correctly identifying 1865 instances of fake news (true positives) and 3695 instances of genuine news (true negatives). Compared to earlier models such as Naïve Bayes, SVM, and Random Forest, it significantly reduces the number of false positives and false negatives, mistaking 271 instances of fake news for genuine and 170 instances of genuine news for fake. When it comes to detecting false news, the LSTM model is the most reliable option because it has the lowest false positive and false negative rates compared to the other classifiers. Although hyperparameter tuning, additional training data, or attention mechanisms can further improve its accuracy, its ability to capture textual contextual linkages and temporal patterns is a major factor contributing to its high accuracy.

Table 3. Presents the performance of three classifiers.

Metric	Naïve Bayes	Random Forest	SVM	LSTM
Accuracy	0.685	0.827	0.798	0.927
Precision	0.703	0.823	0.780	0.932
Recall	0.885	0.932	0.955	0.956
F1 Score	0.783	0.874	0.859	0.944
ROC AUC	0.603	0.785	0.734	0.957
Log Loss	11.370	6.229	7.292	0.207

Table 3, which is part of the Model Comparison, shows how three classifiers Naïve Bayes, Random Forest, and SVM performed across various assessment criteria. The best overall performance is indicated by Random Forest's highest accuracy (82.72%), followed by SVM (79.77%), and Naïve Bayes (68.46%). With a precision of 0.822979, Random Forest achieves better results than SVM (0.780292) and Naïve Bayes (0.702547), indicating a reduction in false positives. Although its accuracy is marginally lower than that of other methods, support vector machines (SVMs) have the best recall (0.954722), making them the most effective at accurately detecting genuine news. When it comes to balancing accuracy and recall, Random Forest has the highest F1-score (0.874196), whereas SVM (0.858739) and Naïve Bayes (0.783236) have lower scores. The top three models for differentiating between fake and real news, according to ROC AUC values, are Random Forest (0.784692), SVM (0.734149), and Naïve Bayes (0.603481). The most confident predictions are produced by Random Forest (6.228507) according to the log loss values, whereas the most uncertain forecasts are made by Naïve Bayes (11.369878). In terms of overall performance, Random Forest is the most suitable model for identifying false news, SVM excels in recall, and Naïve Bayes is the least successful model.

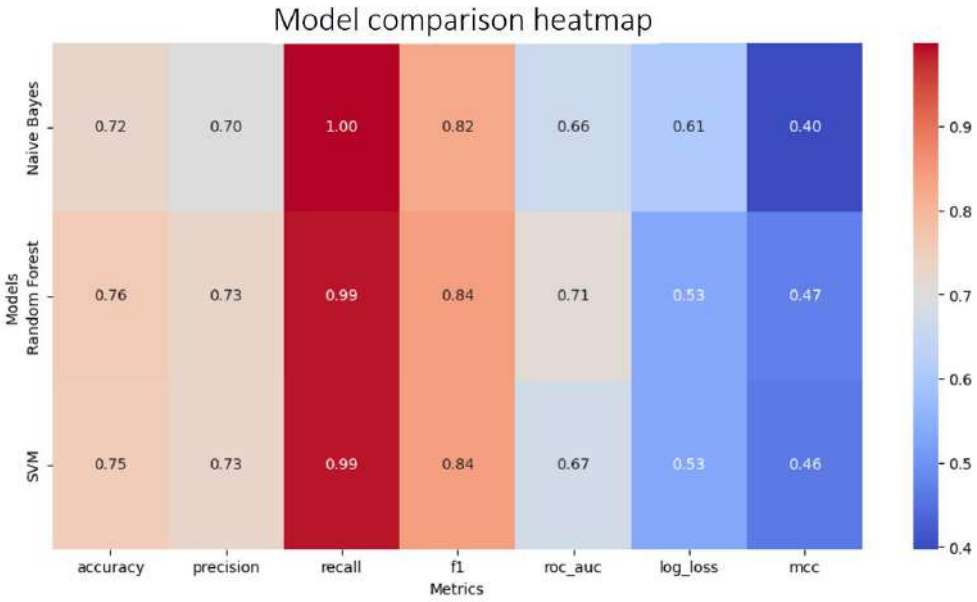

Figure 20. Model comparison heatmap visualizes.

4.4. Word2Vec Word Embeddings

In Figure 20, the Model Comparison Heatmap, different colors indicate varying degrees of performance, illustrating how Naïve Bayes, Random Forest, and SVM perform across multiple assessment measures. All models exhibit a high recall (around 1.00), indicating they are highly effective at correctly classifying positive cases. With an F1-score of 0.84 and an accuracy of 0.76, Random Forest is the most balanced model. Naïve Bayes has slightly lower precision (0.70), while both Random Forest and SVM have a precision of 0.73, suggesting they are better at reducing false positives. Regarding discriminatory power, Random Forest has the highest ROC AUC (0.71), whereas Naïve Bayes has the lowest (0.66). The most confident predictions are made by Random Forest and SVM, which have the lowest log loss (0.53), in contrast to Naïve Bayes, which has a higher log loss (0.61), indicating greater uncertainty. The resilience of Random Forest is further supported by its highest Matthews Correlation Coefficient (MCC) of 0.47. Naïve Bayes performs poorly in most categories except recall, while Random Forest and SVM emerge as the top-performing models overall.

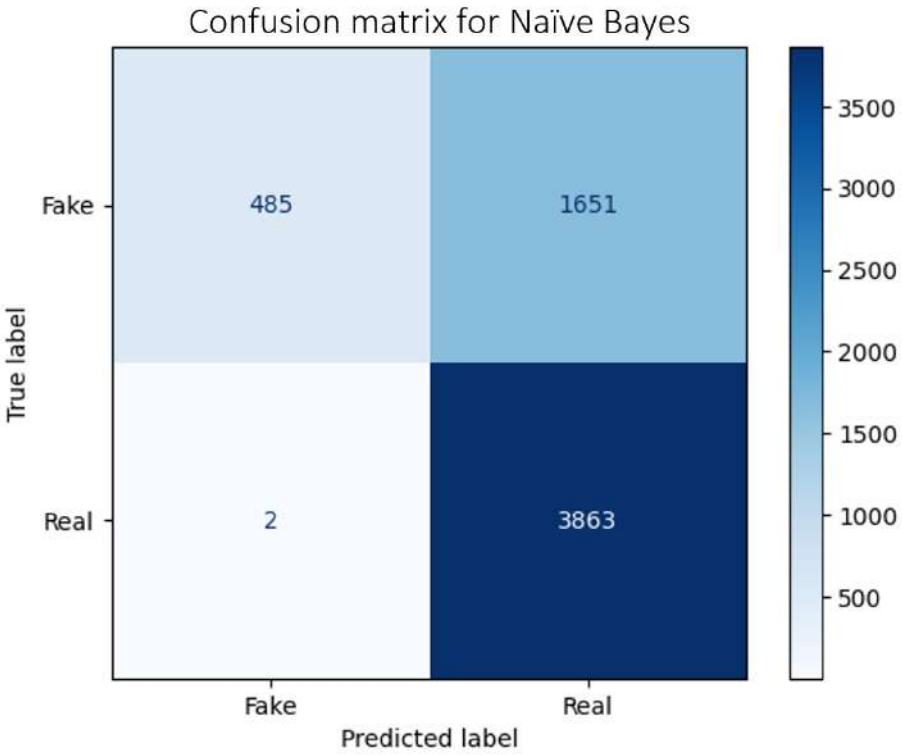

Figure 21. Confusion matrix for the Naïve Bayes classifier.

The accuracy of the Naïve Bayes classifier in differentiating between false and actual news is assessed using the confusion matrix in Figure 21. Out of 3,863 occurrences of actual news, the model accurately identifies 485 as false positives. Unfortunately, it mistakenly classifies 1,651 pieces of bogus news as genuine, indicating a significant percentage of false positives. At the same time, only two examples of actual news were incorrectly labeled as fake, demonstrating that Naïve Bayes has a strong bias towards correctly predicting real news but faces considerable challenges in detecting fake news. This reveals a discrepancy in the classification performance of Naïve Bayes, as it achieves good recall for genuine news but low precision for false news. To improve the effectiveness of fake news detection and reduce false positives, the model could benefit from better feature selection, adjusted class priors, or more advanced methods such as ensemble learning.

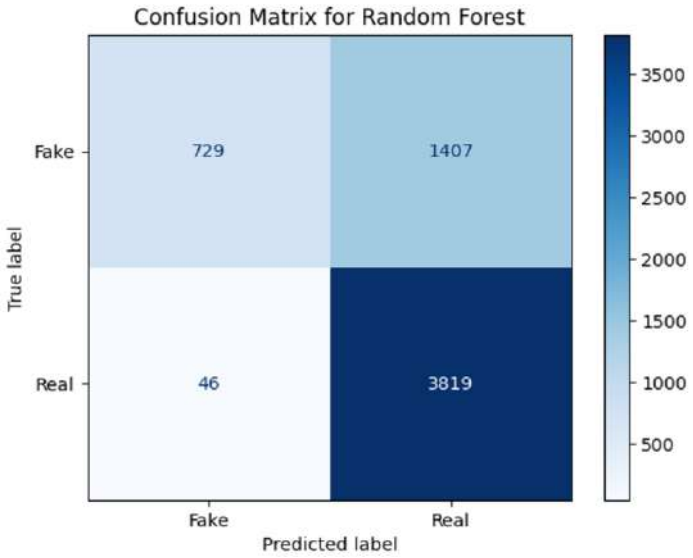

Figure 22. Confusion matrix for the Random Forest classifier.

The performance of the Random Forest classifier in distinguishing between authentic and fraudulent news is evaluated in Figure 22, which presents the confusion matrix. The classifier demonstrates high accuracy, correctly identifying 729 cases of false news and 3819 cases of legitimate news. Conversely, it produces 1407 false positives and 46 false negatives, indicating that it incorrectly classifies legitimate news as false. Random Forest significantly improves the detection of actual news stories by reducing false negatives compared to Naïve Bayes. However, there remains a substantial number of false positives, where fake news is mistakenly classified as legitimate. Although Random Forest outperforms Naïve Bayes in terms of balance and robustness, further optimization through hyperparameter tuning, feature engineering, or ensemble methods could enhance its ability to distinguish false news more effectively.

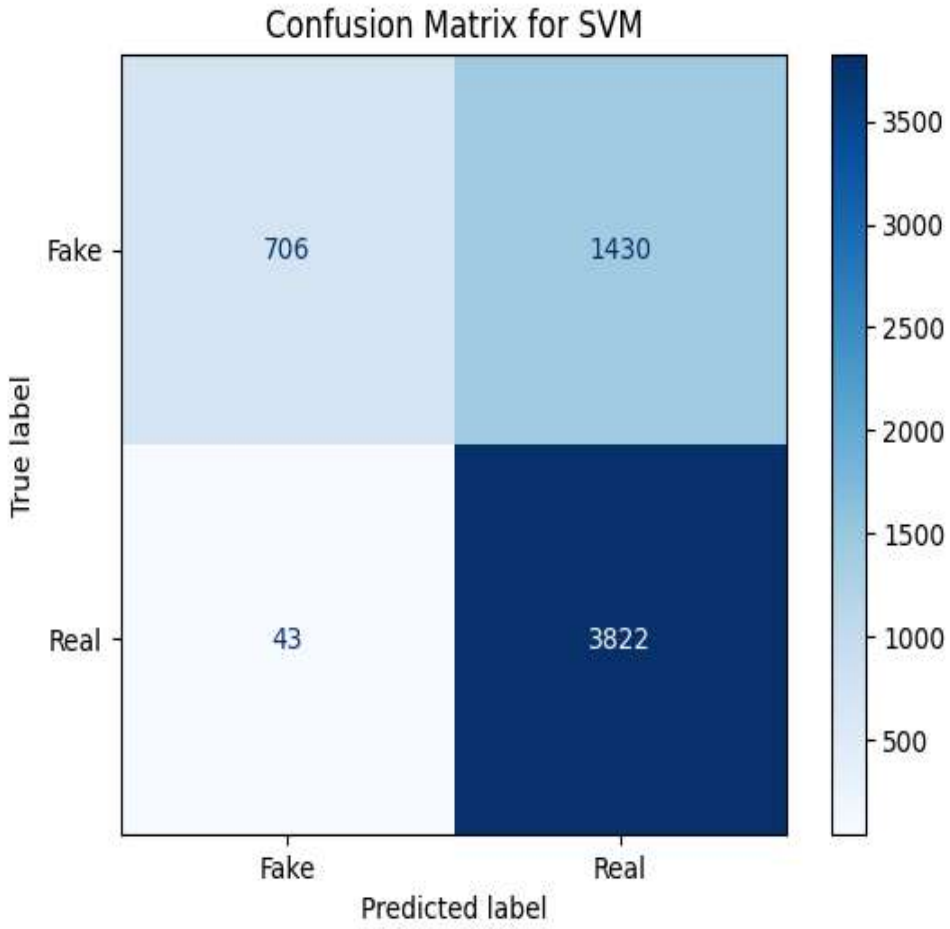

Figure 23. Confusion matrix for the SVM classifier to differentiate between true and fraudulent news.

Figure 23 shows the SVM classifier's confusion matrix, which measures how well it differentiates between true and fraudulent news. The model demonstrates strong classification skills, accurately identifying 706 instances of fake news (true positives) and 3,822 cases of legitimate news (true negatives). However, it incorrectly classifies 1,430 instances of fake news as genuine (false positives) and 43 instances of true news as fake (false negatives). With a slightly higher false positive rate than Random Forest, SVM has a more difficult time incorrectly labeling misleading news as genuine. Conversely, it reliably detects actual news due to its low false negative rate. Enhancements such as feature selection, kernel improvements, or incorporating additional textual representations like word embeddings (Word2Vec, FastText) could improve SVM's effectiveness in detecting fake news.

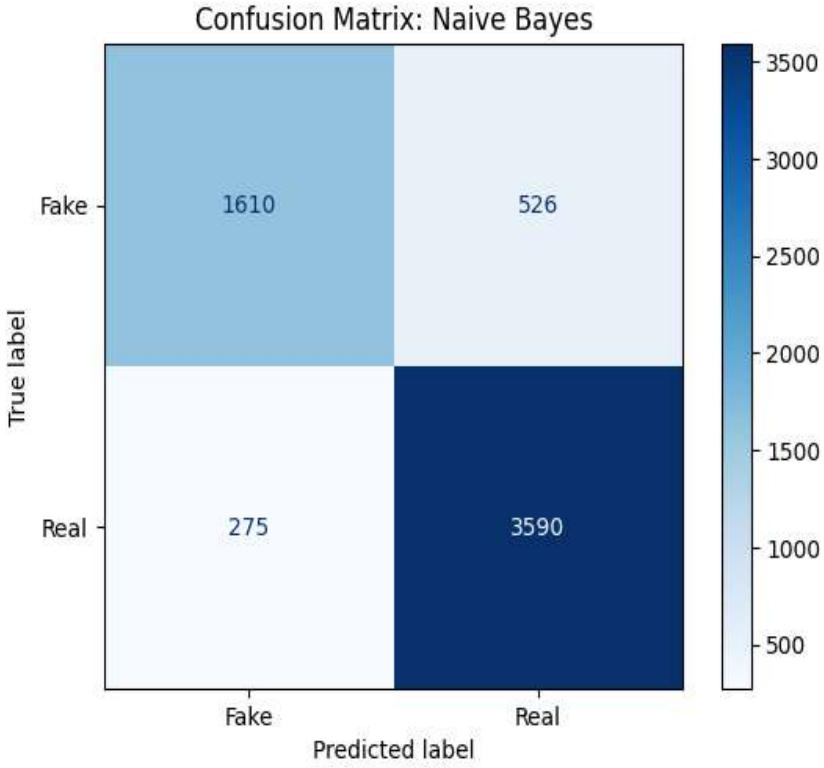

Figure 24. Confusion matrix for the Naïve Bayes classifier distinguishing true and fraudulent news.

5. RESULT ANALYSIS

5.1. TF-IDF Word Embeddings

Figure 24 shows the Naïve Bayes classifier's confusion matrix, which assesses how well it distinguishes between true and fraudulent news. Thanks to its excellent categorization abilities, the model has identified 1,610 cases of false news (true positives) and 3,590 cases of actual news (true negatives). It does, however, mistake 526 cases of false positives for actual news and 275 cases of real news for fake news. With fewer false positives, our model shows better accuracy in identifying fake news compared to earlier Naïve Bayes findings. Nevertheless, there seems to be some issue with reliably detecting true news, as indicated by the 275 false negatives. Naïve Bayes appears to be effective at detecting false news, but it may benefit from further text preprocessing, feature selection, or ensemble approaches.

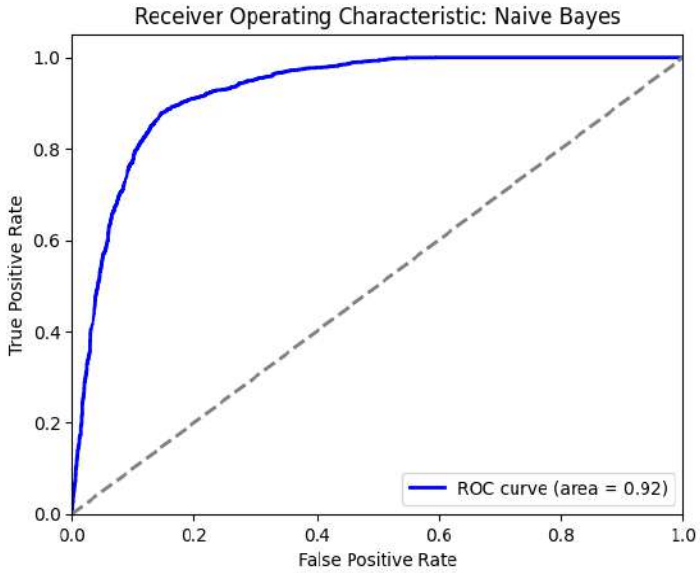

Figure 25. Receiver operating characteristic (ROC) curve.

In Figure 25, the Receiver Operating Characteristic (ROC) curve of the Naïve Bayes classifier is shown. This curve illustrates the ability of the classifier to distinguish between genuine and counterfeit news. The model's capacity to differentiate across classes is demonstrated by plotting the True Positive Rate (TPR) on the y-axis against the False Positive Rate (FPR) on the x-axis. Since the curve is significantly above the diagonal (random guess line), the classification performance is considered satisfactory. The high discriminative power of the algorithm is further evidenced by the Area Under the Curve (AUC) value of 0.92. A larger AUC, closer to 1, indicates a higher level of discrimination. Despite this strong performance, there remains potential for improvement through feature engineering, hyperparameter tuning, or ensemble techniques, which could enhance accuracy and reduce the number of false positives produced.

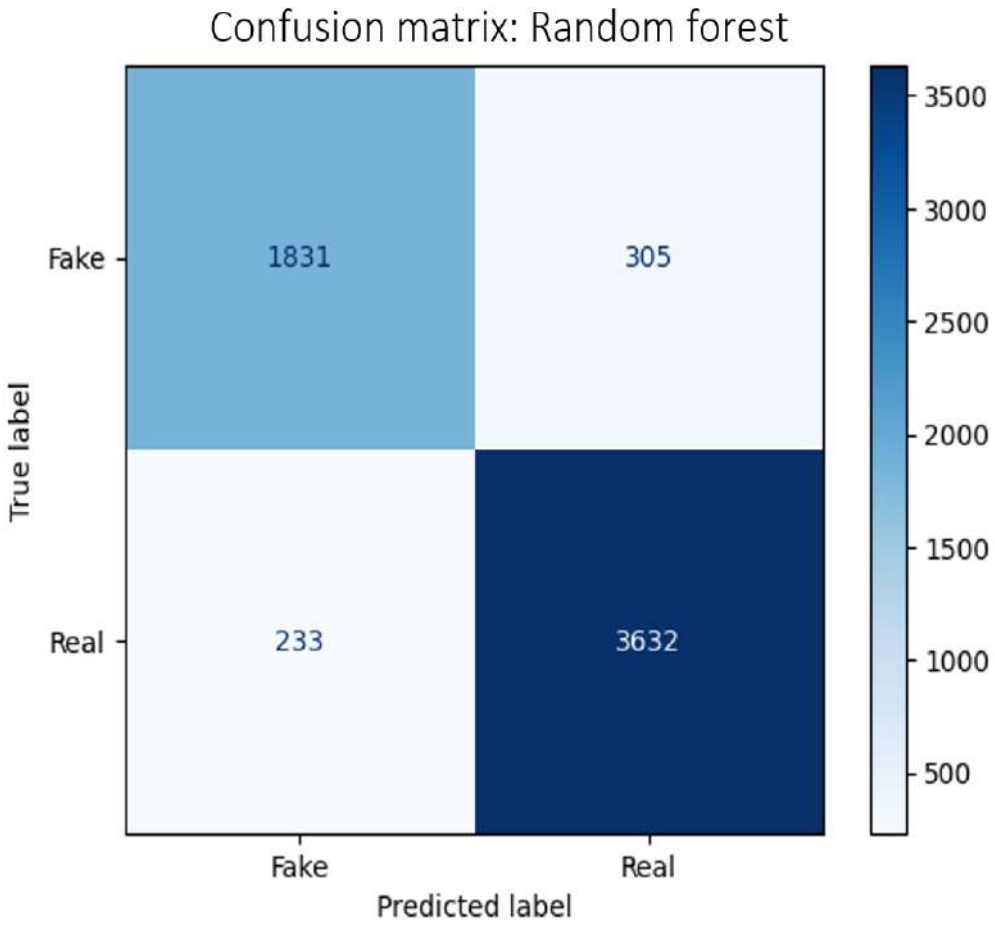

Figure 26. Confusion matrix for the random forest classifier.

The Random Forest classifier's accuracy in distinguishing between false and true news is assessed in Figure 26, which shows the confusion matrix. The model demonstrates impressive classification capabilities, correctly identifying 1831 cases of false news (True positives) and 3632 cases of authentic news (True negatives). Unfortunately, it incorrectly classifies 305 instances of fake news as genuine (False positives) and 233 instances of actual news as fake (False negatives). Random Forest achieves a more balanced classification performance by substantially reducing false positives and false negatives compared to Naïve Bayes. With its high accuracy and recall and low misclassification rate, Random Forest appears to be a robust model for detecting false news. Further reduction of classification errors may be achieved through hyperparameter tuning, additional feature selection, or ensemble approaches.

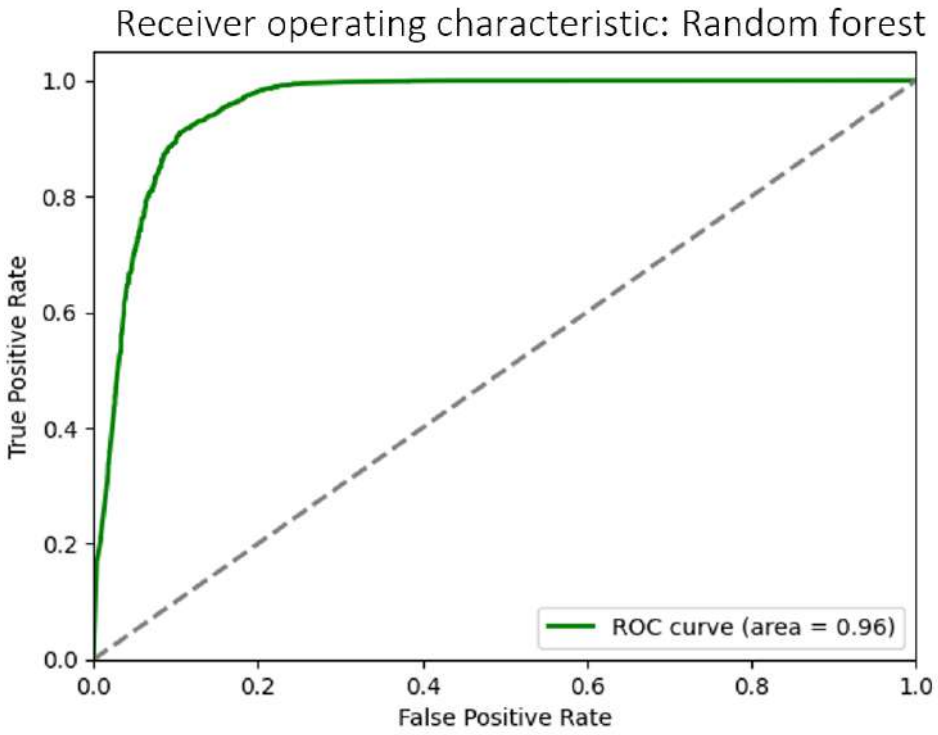

Figure 27. Receiver operating characteristic (ROC).

Figure 27 represents the Receiver Operating Characteristic (ROC) curve of the Random Forest classifier. The curve shows how well our classifier distinguishes between actual and fake news. To demonstrate that the model can differentiate between classes, we plot the True Positive Rate (TPR) on the y-axis against the False Positive Rate (FPR) on the x-axis. The steep curve (far away from the diagonal, also known as a random guess line) indicates very high classification performance. After running the Random Forest model, we obtained an Area Under the Curve (AUC) value of 0.96, which significantly improves upon the Naïve Bayes AUC value of 0.92. The Random Forest model also shows a notable difference, indicating good discriminating ability. With low levels of false positives and negatives, and a record area under the curve (AUC), it can be concluded that the model effectively differentiates between fake news and genuine content. Approaches such as hyperparameter tuning, additional feature selection, or ensemble learning can be employed to develop models with better performance and fewer misclassifications.

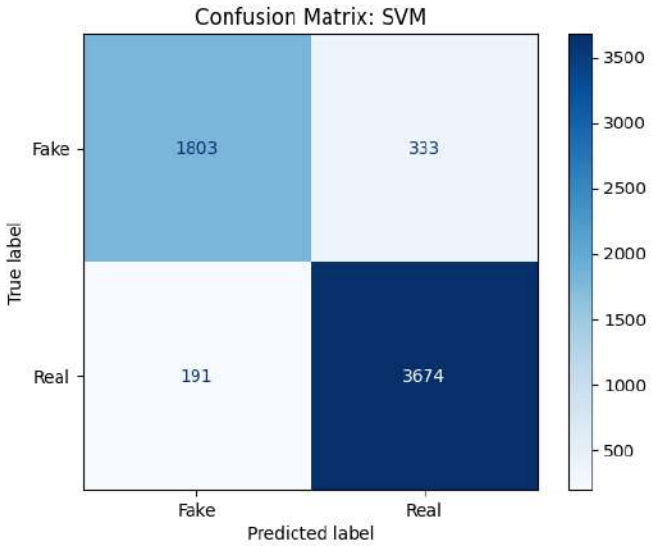

Figure 28. Confusion matrix for the SVM (Support Vector Machine) classifier.

The SVM classifier's effectiveness in distinguishing between true and fraudulent news is assessed in Figure 28, which is the confusion matrix. The model demonstrates strong classification skills, accurately classifying 1803 instances of fake news (true positives) and 3674 instances of legitimate news (true negatives). Conversely, it incorrectly identifies 191 instances of genuine news as fake and 333 cases of fake news as genuine. In terms of detecting fake news, SVM performs slightly better than Random Forest; however, it misclassifies a few more instances of actual news and has a marginally higher number of false negatives. Overall, SVM is an effective model for detecting false news because it balances recall and accuracy well. Further improvements in classification performance can be achieved through kernel optimization, feature selection, and hyperparameter tuning.

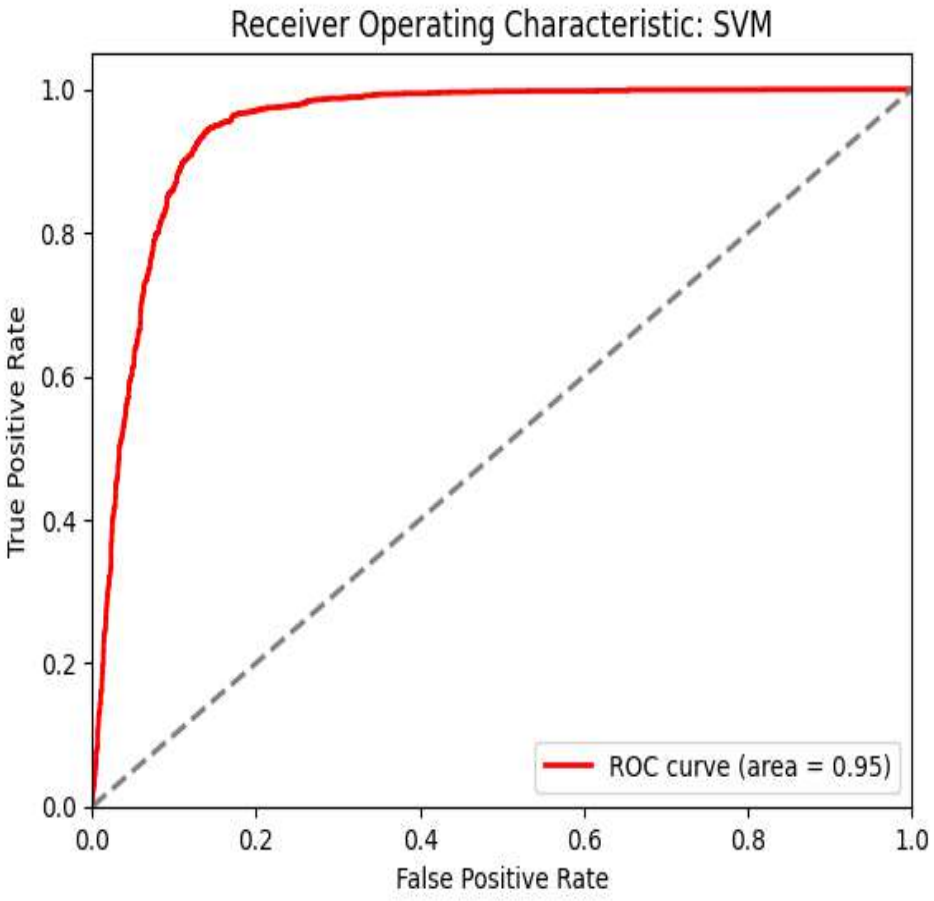

Figure 29. Receiver operating characteristic (ROC) curve for the SVM (Support vector machine).

The effectiveness of the Support Vector Machine (SVM) classifier in distinguishing between true and fraudulent news is demonstrated by its Receiver Operating Characteristic (ROC) curve in Figure 29. A steep curve, positioned above the diagonal (random chance line), indicates strong classification ability, as shown here against the False Positive Rate (FPR). With an Area Under the Curve (AUC) of 0.95, SVM exhibits superior discriminative capabilities, comparable to Random Forest (AUC = 0.96) and exceeding Naïve Bayes (AUC = 0.92). The low number of false positives and negatives suggests that SVM is quite effective at classifying the two groups. However, hyperparameter tuning, kernel selection, or advanced feature extraction techniques could further reduce misclassification rates, potentially enhancing performance.

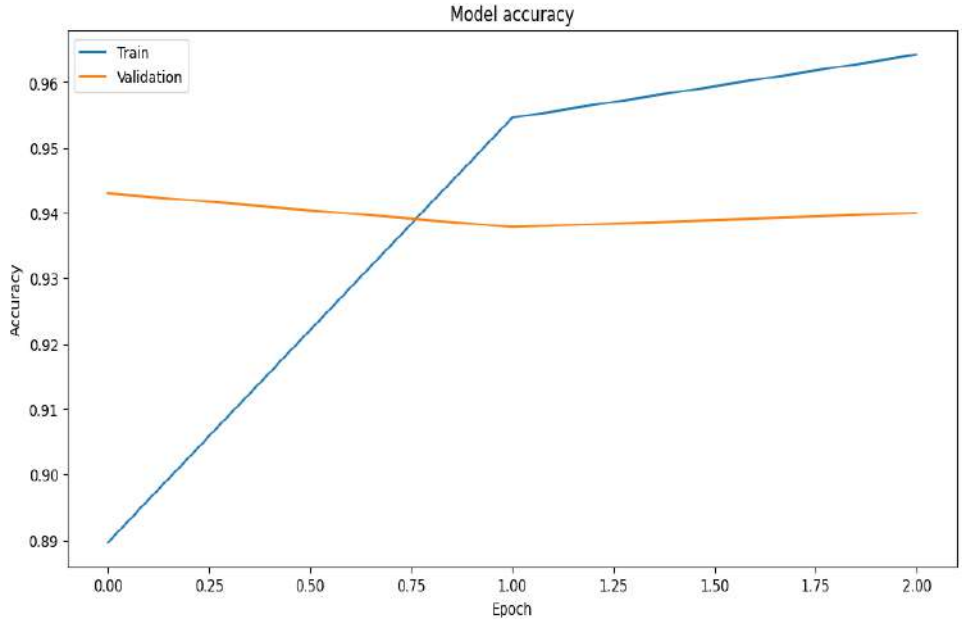

Figure 30. Model Accuracy displays the training and validation accuracy over epochs.

The accuracy of the training and validation throughout epochs is shown in Figure 30, which is titled "Model Accuracy." The training accuracy, represented by the blue line, starts at approximately 89% and rapidly increases to over 96% within less than two epochs, indicating a high rate of learning. The orange line illustrates the validation accuracy, which begins at around 94% and gradually decreases until it stabilizes at a similar level. For the training data, the model performs quite well; however, its performance on validation data does not significantly improve, if at all. This discrepancy between training and validation accuracy may suggest overfitting, where the model learns the training data too closely but struggles to generalize to new data. Several methods can be employed to enhance generalizability, including regularization techniques such as L1 and L2, dropout, early stopping, and increasing the amount of training data.

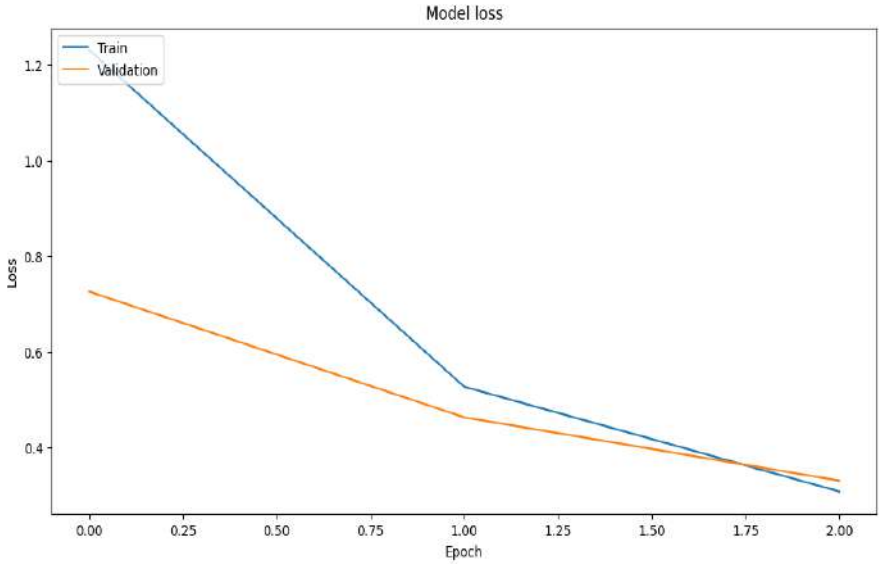

Figure 31. Model Loss displays the training and validation accuracy over epochs.

The loss that occurs during training and validation at various epochs is shown in Figure 31 of the Model Loss. The training loss, represented by the blue line, starts at approximately 1.2 and decreases significantly as the model learns, indicating successful optimization. The validation loss, shown by the orange line, also decreases consistently,

although at a slower rate. By the time the second epoch is reached, validation loss remains considerably higher than training loss. Both losses tend to converge over time. The fact that the gap between training loss and validation loss is shrinking but not entirely closing suggests that the model may still exhibit some overfitting, despite effective learning. To improve generalization and prevent overfitting, methodologies such as data augmentation, early stopping, batch normalization, and dropout can be employed.

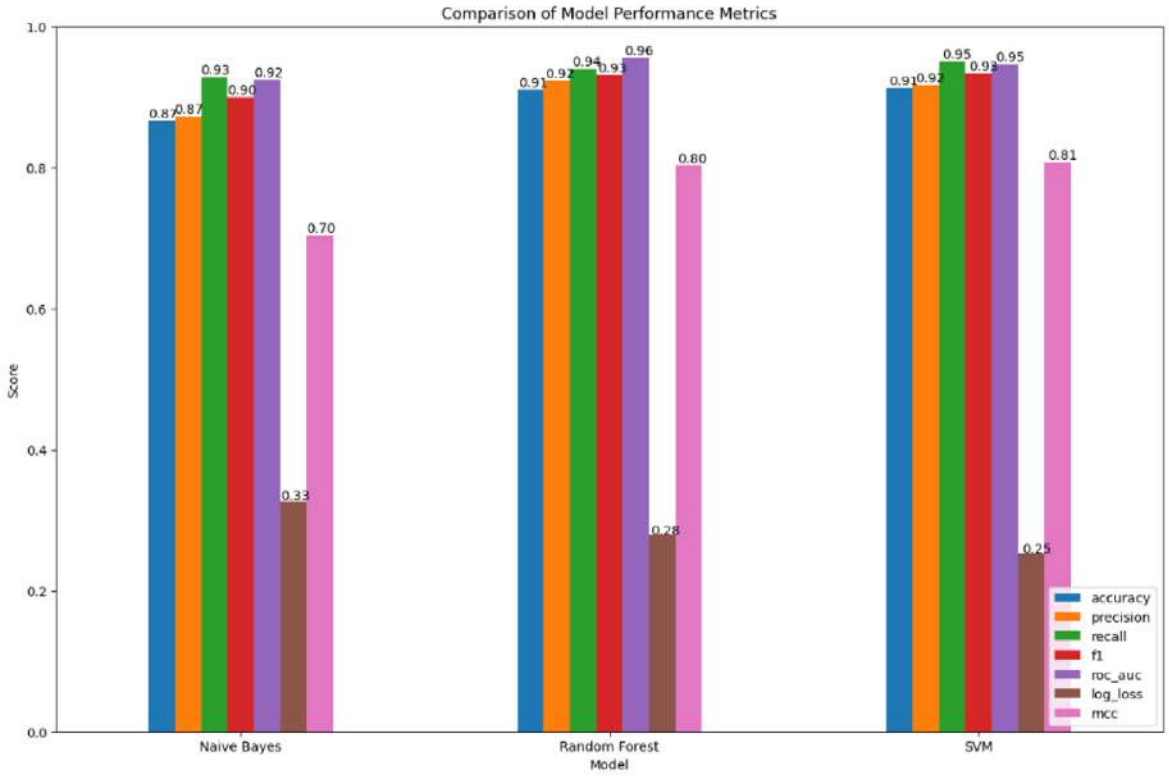

Figure 32. Model performance comparison for three different ML models.

Figure 32, also known as the Model Performance Comparison Bar Chart, displays the performance metrics for three different machine learning models. These models include Naïve Bayes, Random Forest, and SVM (Support Vector Machine). Recall, accuracy, precision, F1-score, area under the curve (ROC), log loss, and Matthews correlation coefficient (MCC) are some of the measurements shown.

Naïve Bayes has a lower accuracy of 0.87 and a relatively strong recall of 0.93, suggesting that it tends to identify more cases as positive while incurring some false positives. Additionally, its lower confidence in its predictions is indicated by its larger log loss (0.33) compared to other models.

With an accuracy of 0.92, a precision of 0.94, a recall of 0.96, and an F1-score of 0.93, the Random Forest model outperforms all other models. This makes it the best-balanced model when considering recall and precision. Better prediction confidence is demonstrated by its smaller log loss of 0.28.

SVM achieves results comparable to Random Forest, with a 0.93 F1-score, thanks to its strong recall (0.93) and accuracy (0.95). It is quite dependable since its MCC is 0.81, which indicates a high degree of correlation with real class labels.

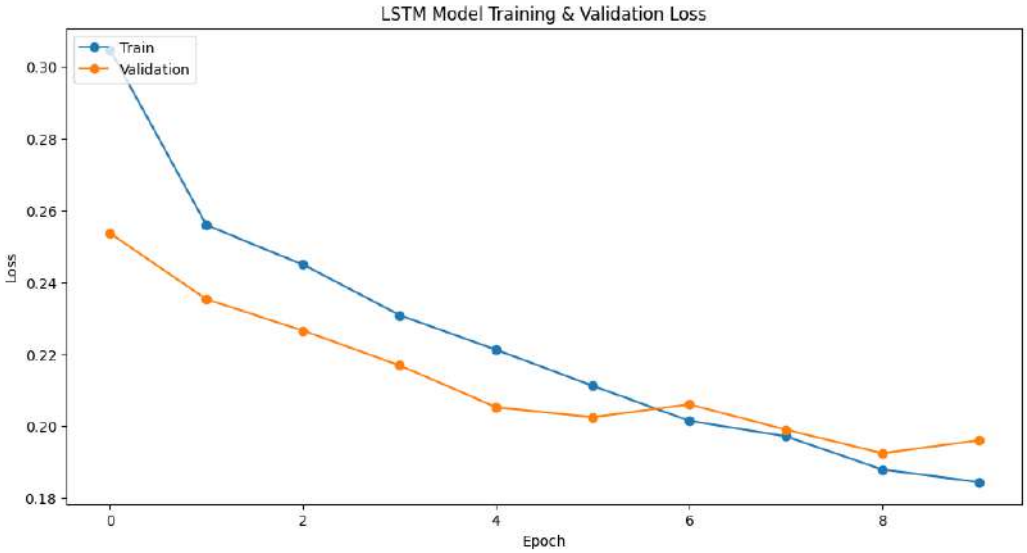

Figure 33. LSTM model training & validation loss.

5.2. Fast Text Word Embeddings

Figure 33 illustrates the LSTM Model Training & Validation Loss, depicting the evolution of the model's loss across epochs for both validation and training datasets. The training loss, represented by the blue line, begins at approximately 0.30 and steadily decreases as the model learns from the training data. The validation loss, shown by the orange line, also decreases initially but begins to fluctuate around the sixth epoch, indicating the model may be approaching its optimal learning capacity. The relatively small gap between training and validation losses, despite both trending downward, suggests minimal overfitting and good generalization to unseen data. However, slight increases in validation loss in later epochs could be early signs of overfitting. To prevent the model from memorizing noise rather than learning meaningful patterns, techniques such as early stopping or dropout regularization can be employed to halt training before deterioration occurs.

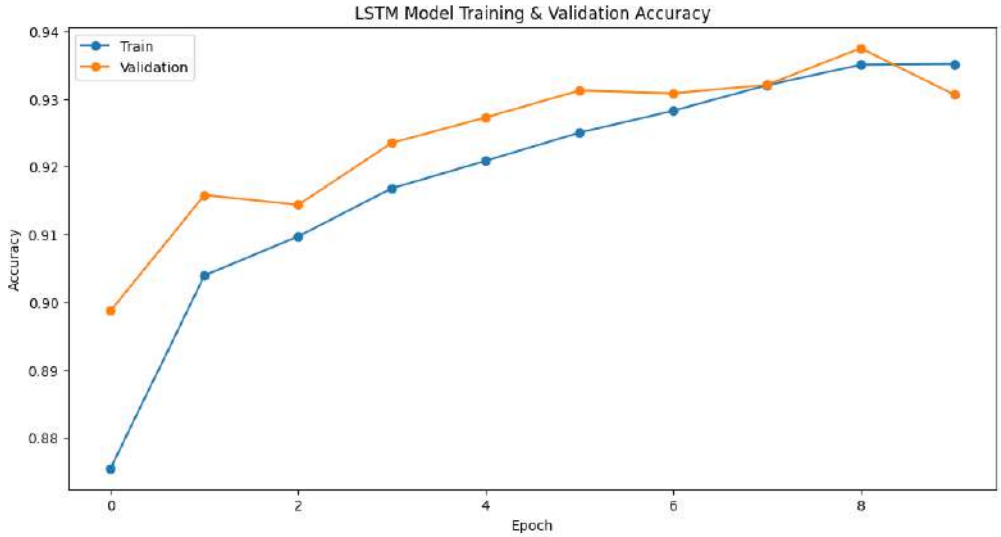

Figure 34. LSTM model training & validation accuracy.

Figure 34 displays the LSTM Model Training & Validation Accuracy, which illustrates the model's accuracy improvement for both the training and validation datasets throughout the course of the training epochs. The model is learning successfully, as the blue line shows the training accuracy, which begins at about 88% and gradually rises to more than 93%. The validation accuracy, shown by the orange line, increases dramatically at the beginning and

stays somewhat higher than the training accuracy until about epoch 7, when the two lines converge. The model seems to have good generalizability to new data, despite a small decrease in validation accuracy at the last epoch that might be due to overfitting. The robust generalization with minimal performance deterioration is indicated by the tight relationship between the validation accuracy and the training accuracy. Stabilizing accuracy and preventing overfitting in future rounds might be achieved using strategies such as early stopping, fine-tuning learning rates, or adding dropout layers, which could further enhance performance.

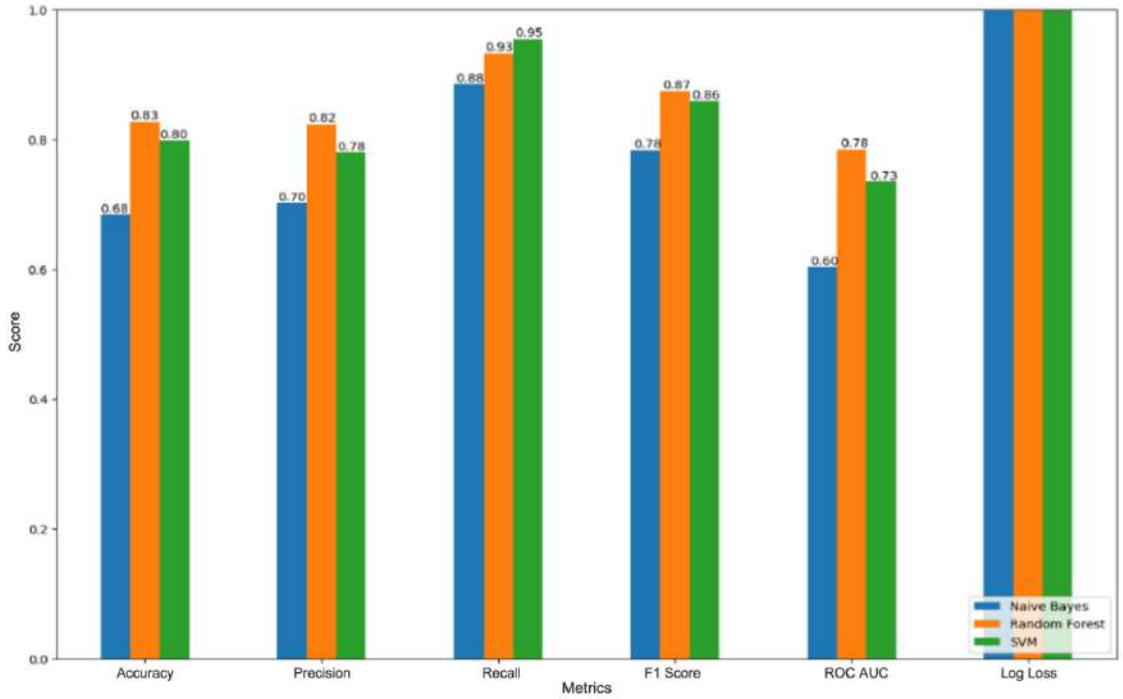

Figure 35. Comparison of model performance metrics.

A comparison of the performance metrics used to assess Naïve Bayes, Random Forest, and Support Vector Machine (SVM) models is shown in Figure 35. Accuracy, precision, recall, F1 score, ROC area under the curve, and log loss are some of these measurements.

The greatest overall performance was achieved by Random Forest (orange), which had the maximum recall (0.95), accuracy (0.83), and a good F1 score (0.87). Additionally, the ROC AUC is strong at 0.78, suggesting that it effectively distinguishes between groups.

SVM (green) follows closely after, with a high recall of 0.93 and an F1 score of 0.86; it performs slightly worse in accuracy and precision but still demonstrates excellent performance.

The blue Naïve Bayes model is not very strong in most measures, including accuracy (0.68), precision (0.70), and F1 score (0.78). However, it has a decent recall (0.88), which means it usually identifies positive cases correctly but makes many mistakes.

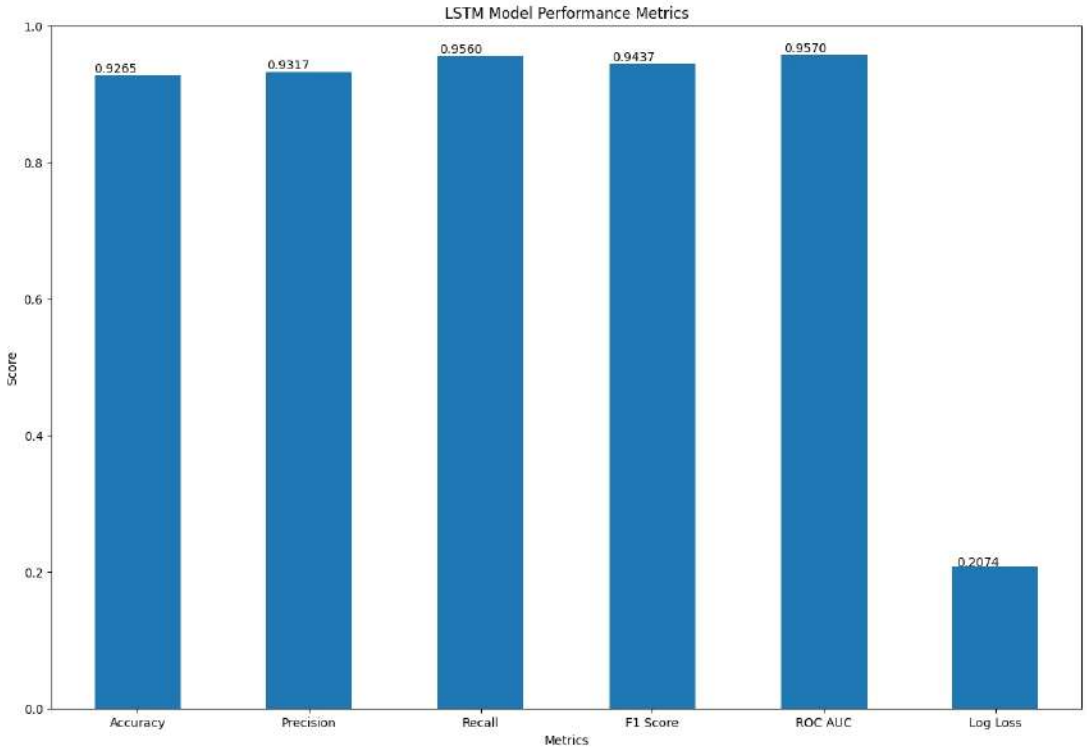

Figure 36. LSTM model performance metrics.

The assessment scores of an LSTM model across key performance measures are shown in Figure 36 of the LSTM Model Performance Metrics. These indicators include accuracy (0.9265), precision (0.9317), recall (0.9560), F1 score (0.9437), ROC AUC (0.9570), and log loss (0.2074). The model's high recall indicates that it successfully recognizes positive examples, while its high accuracy and precision demonstrate strong predictive power. The F1 score confirms that recall and accuracy are balanced. An excellent class separation is indicated by a high ROC AUC of 0.9570, and the low log loss of 0.2074 reflects confidence in probability predictions. Overall, the LSTM model consistently outperforms other models in classification tasks.

Table 4. Model Comparison presents the performance metrics of three classification models.

Metric	Naïve Bayes	Random Forest	SVM
Accuracy	0.685	0.827	0.798
Precision	0.703	0.823	0.780
Recall	0.885	0.932	0.955
F1 Score	0.783	0.874	0.859
ROC AUC	0.603	0.785	0.734
Log Loss	11.370	6.229	7.292

The three classification models Naïve Bayes, Random Forest, and SVM have their performance metrics shown in Model Comparison Table 4. All metrics measured are at their highest with Random Forest, which achieves 0.8272, precision 0.8229, recall 0.9322, F1 score 0.8742, and ROC AUC 0.7847. Following closely is SVM, which shows good recall performance with an F1 score of 0.8587, precision of 0.7803, recall of 0.9547, and accuracy of 0.7977. In comparison to Random Forest (6.23) and SVM (7.29), Naïve Bayes has a greater log loss of 11.37 due to its lower accuracy of 0.6846 and precision of 0.7025, despite having the greatest recall of 0.8849. When it comes to minimizing errors and improving forecast accuracy, Random Forest is the most well-rounded model.

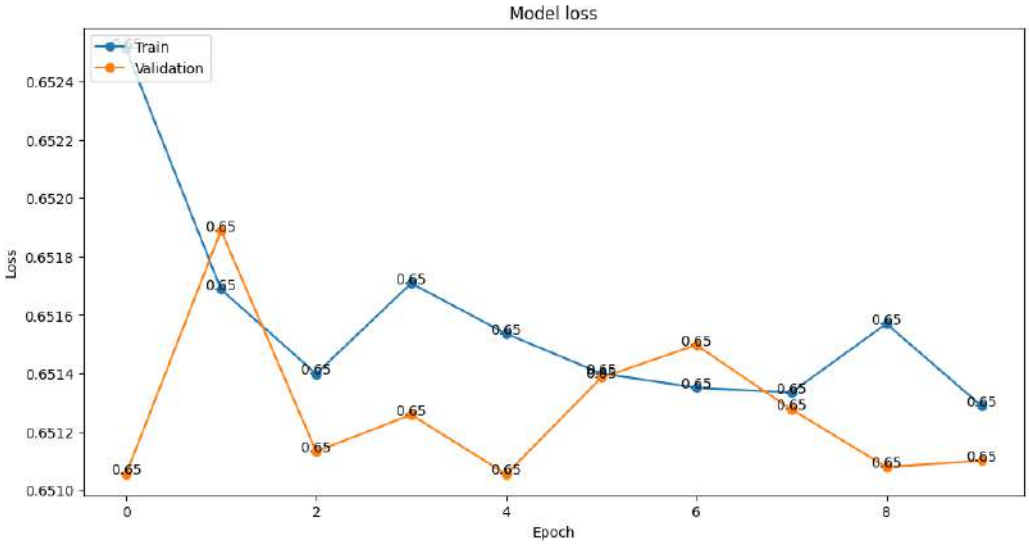

Figure 37. The training and validation loss across multiple epochs.

5.3. Word2Vec Word Embeddings

The Model Loss is shown in Figure 37. This figure illustrates the loss that occurs during training and validation throughout a large number of epochs. The initial training loss is greater than the subsequent losses, but it drops significantly in the first few epochs before reaching a plateau, which indicates that the model is gaining knowledge from the data. The validation loss is relatively stable and remains approximately at 0.65, suggesting that the model does not seem to be making much improvement on data that has not yet been seen. The small disparity between the training and validation losses indicates that the model is not overfitting; however, it may be experiencing difficulties in understanding complex patterns. Possible explanations include a shortage of training data, a slow learning rate, or the need for more robust features.

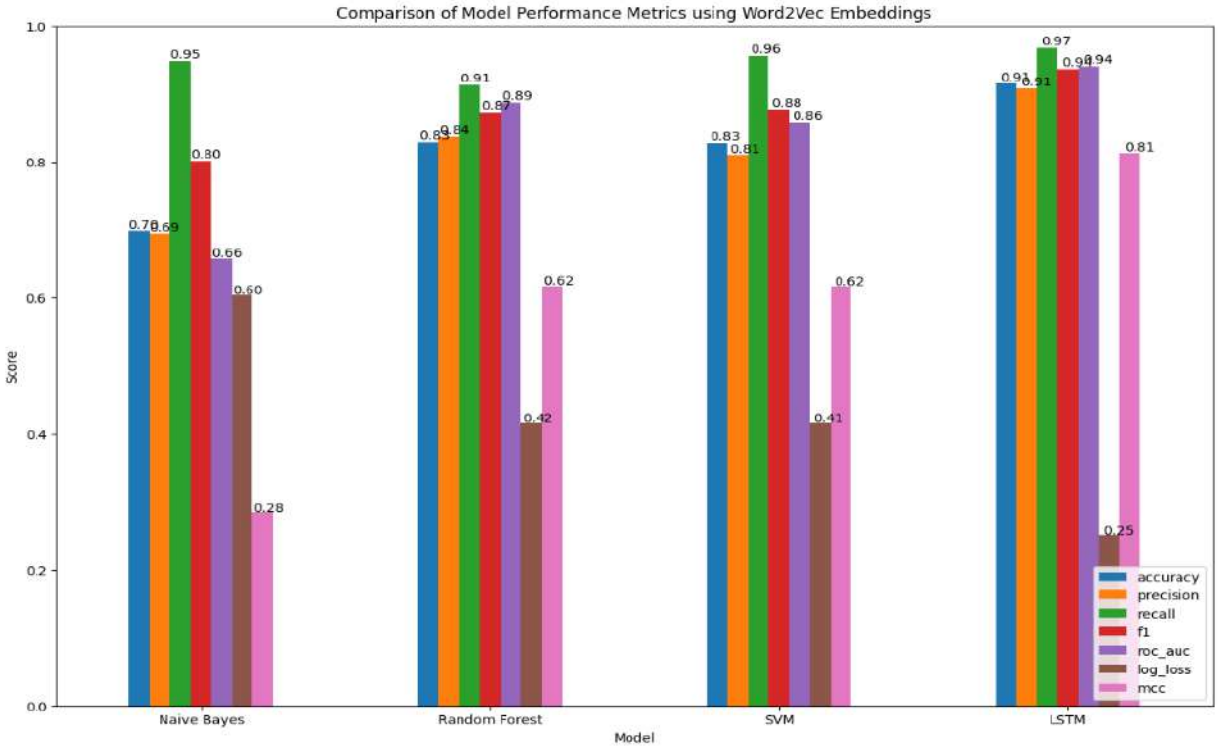

Figure 38. Model performance comparison for Word2Vec embeddings.

Figure 38 shows the results of comparing the performance of several classification models utilizing Word2Vec embeddings. These models include Naïve Bayes, Random Forest, SVM, and LSTM. The models were evaluated using different performance measures. With the best accuracy (0.98), precision (0.91), recall (0.97), F1-score (0.94), and ROC AUC (0.94), as well as a significantly lower log loss (0.25), the LSTM model outperforms all other models. Similar to LSTM, Random Forest and SVM perform well with recall values above 0.90, although they are not quite as accurate and have lower F1-scores. When it comes to capturing text-based patterns using Word2Vec embeddings, Naïve Bayes performs the poorest, with the lowest accuracy (0.78), precision (0.69), and F1-score (0.80). It appears that LSTM maintains the greatest balance across all assessment parameters, as it has the highest Matthews Correlation Coefficient (MCC).

5.4. Critical Discussion of Model Behavior

The quantitative results confirm that the LSTM model works better, but the training and validation curves (Figs 30–31) show that it might be overfitting. The training accuracy, on the other hand, increases rapidly and remains much higher than the validation accuracy. The validation loss, in turn, does not converge at the same rate as the training loss. This difference in results indicates that the LSTM model might be memorizing patterns from the training data instead of applying what it has learned to new samples. This behavior is not unusual in sequential models used on small or unbalanced datasets. Future enhancements may include dropout layers, early stopping, L2 regularization, or dataset augmentation to mitigate the generalization gap and improve robustness.

The comparative results across models demand more profound interpretation beyond mere accuracy and F1-scores. Naïve Bayes consistently performed poorly because it assumes independence, which doesn't account for contextual dependencies in language structures. This is particularly important for detecting fake news, where small semantic cues can make a big difference in class separation. SVM had a high recall rate but a lower precision rate. This is because it is very difficult to tell the difference between noisy and high-dimensional text when linear kernels are used. Random Forest gave balanced results because ensemble averaging lowers variance, but it doesn't have the sequential learning ability needed to pick up on context-dependent features in complex texts. The LSTM model, on the other hand, did better than all the other models because it could learn long-range dependencies and how meaning flows through sentences. But the better performance of LSTM comes at a cost, as shown by the overfitting patterns that were seen. This shows that careful model regularization is still needed.

6. CONCLUSION

This research conducted an extensive comparative study of sentiment analysis models utilizing TF-IDF, FastText, and Word2Vec embeddings, combined with four different classifiers: Naïve Bayes, Random Forest, SVM, and LSTM. The LSTM model outperformed all others, achieving an accuracy of 92.65%, an F1-score of 94.37%, a precision of 93.17%, a recall of 95.60%, and a ROC-AUC score of 95.70%. Notably, the LSTM model with Word2Vec embeddings performed exceptionally well, even in the Log Loss variant, which achieved the best score of 0.2074. Among traditional classifiers, Random Forest was the most effective, followed by SVM, while Naïve Bayes lagged due to its limited capacity to handle complex data representations. The study emphasizes the importance of deep learning and advanced word embedding techniques in enhancing sentiment classification accuracy. Additionally, the robustness and generalization capabilities of the models were validated through detailed analyses using confusion matrices, ROC curves, and training-validation plots. This research offers practical guidelines for selecting suitable model-embedding combinations for real-world sentiment analysis applications. Future work will explore hybrid models, ensemble methods, and attention mechanisms to further improve text classification tasks.

However, the work is not without limitations. The dataset used in this study exhibited a degree of class imbalance, with a higher proportion of real news compared to fake news, which may have influenced model performance and generalizability. Additionally, the raw text data initially contained non-English tokens and encoding artifacts, which

required preprocessing and removal to ensure consistency. Addressing these limitations in future studies through balanced datasets, multilingual support, and more sophisticated data-cleaning pipelines will help further strengthen the robustness and applicability of sentiment analysis models.

Funding: This study received no specific financial support.
Institutional Review Board Statement: Not applicable.
Transparency: The authors state that the manuscript is honest, truthful, and transparent, that no key aspects of the investigation have been omitted, and that any differences from the study as planned have been clarified. This study followed all writing ethics.
Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Disclosure of AI Use: The author(s) used OpenAI's ChatGPT to edit and refine the wording of the Introduction. All outputs were reviewed and verified by the authors.

REFERENCES

[1] P. Akter *et al.*, "Sentiment analysis of consumer feedback and its impact on business strategies by machine learning," *The American Journal of Applied Sciences*, vol. 7, no. 01, pp. 6-16, 2025. https://doi.org/10.37547/tajas/Volume07Issue01-02

[2] M. Simionescu, "Machine learning vs. econometric models to forecast inflation rate in Romania? The role of sentiment analysis," *Mathematics*, vol. 13, no. 1, p. 168, 2025. https://doi.org/10.3390/math13010168

[3] R. Alhejaili, *Machine learning approaches for sentiment analysis on social media. In AI-Driven: Social Media Analytics and Cybersecurity*. Cham, Switzerland: Springer Nature, 2025.

[4] M. A. S. Saleh and M. AlShafeey, "Examining the synergies between industry 4.0 and sustainability dimensions using text mining, sentiment analysis, and association rules," *Sustainable Futures*, vol. 9, p. 100423, 2025. https://doi.org/10.1016/j.sftr.2024.100423

[5] M. K. Anam *et al.*, "Enhancing the performance of machine learning algorithm for intent sentiment analysis on village fund topic," *Journal of Applied Data Sciences*, vol. 6, no. 2, pp. 1102-1115, 2025. https://doi.org/10.47738/jads.v6i2.637

[6] A. Khan, "Improved multi-lingual sentiment analysis and recognition using deep learning," *Journal of Information Science*, vol. 51, no. 1, pp. 284-291, 2023. https://doi.org/10.1177/01655515221137270

[7] P. Devika and A. Milton, "Book recommendation using sentiment analysis and ensembling hybrid deep learning models," *Knowledge and Information Systems*, vol. 67, no. 2, pp. 1131-1168, 2025. https://doi.org/10.1007/s10115-024-02250-z

[8] J. Singh, G. Singh, and R. Singh, "Optimization of sentiment analysis using machine learning classifiers," *Human-centric Computing and Information Sciences*, vol. 7, no. 1, p. 32, 2017. https://doi.org/10.1186/s13673-017-0116-3

[9] R. Singh and R. Singh, "Applications of sentiment analysis and machine learning techniques in disease outbreak prediction–A review," *Materials Today: Proceedings*, vol. 81, pp. 1006-1011, 2023. https://doi.org/10.1016/j.matpr.2021.04.356

[10] J. Chandra and A. C. Mondal, "Studies of sentiment analysis for stock market prediction using machine learning: A survey towards new research direction," *Scholars Journal of Engineering and Technology*, vol. 13, no. 1, pp. 56-65, 2025. https://doi.org/10.36347/sjet.2025.v13i01.007

[11] M. Kumar, L. Khan, and H.-T. Chang, "Evolving techniques in sentiment analysis: A comprehensive review," *PeerJ Computer Science*, vol. 11, p. e2592, 2025. https://doi.org/10.7717/peerj-cs.2592

[12] A. Albladi, M. Islam, and C. Seals, "Sentiment analysis of Twitter data using NLP models: A comprehensive review," *IEEE Access*, vol. 13, pp. 30444-30468, 2025. https://doi.org/10.1109/ACCESS.2025.3541494

[13] G. P. Dubey, S. Upadhyay, and A. Giri, *Machine learning techniques for sentimental analysis. In Information visualization for intelligent systems*. Cham, Switzerland: Springer, 2025.

[14] R. Rawat, V. Mahor, S. Chirgaiya, R. N. Shaw, and A. Ghosh, *Sentiment analysis at online social network for cyber-malicious post reviews using machine learning techniques. In Computationally intelligent systems and their applications*. Cham, Switzerland: Springer, 2021.

[15] D. M. Alsekait, H. Fathi, S. A. Ibrahim, A. Y. Shdefat, A. S. Alattas, and D. S. AbdElminaam, "Sentiment analysis: A machine learning utilisation for analyzing the sentiments of Facebook and Twitter posts," *Intelligent Data Analysis: An International Journal*, vol. 29, no. 4, pp. 889-912, 2025. https://doi.org/10.1177/1088467X241301389

[16] A. S. Hashim *et al.*, "Leveraging social media sentiment analysis for enhanced disaster management: A systematic review and future research agenda," *Journal of Systems Management Science*, vol. 15, no. 4, pp. 170-191, 2025.

[17] D. Bino, V. Dhanalakshmi, and P. K. Udupi, *Sentiment analysis and machine learning for tourism feedback data analysis: An overview of trends, techniques, and applications. In AI technologies for personalized and sustainable tourism.* Cham, Switzerland: Springer, 2025.

[18] S. M. Ferdous, S. N. E. Newaz, S. B. S. Mugdha, and M. Uddin, "Sentiment analysis in the transformative era of machine learning: A comprehensive review," *Statistics, Optimization & Information Computing*, vol. 13, no. 1, pp. 331-346, 2024. https://doi.org/10.19139/soic-2310-5070-2113

[19] A. Amrullah, "Advanced sentiment analysis using deep learning: A comprehensive framework for high-accuracy and interpretable models," *Intellithings Journal*, vol. 1, no. 1, pp. 21-31, 2025.

[20] J. V. Tembhurne, K. Lakhotia, and A. Agrawal, "Twitter sentiment analysis using ensemble of multi-channel model based on machine learning and deep learning techniques," *Knowledge and Information Systems*, vol. 67, no. 2, pp. 1045-1071, 2025. https://doi.org/10.1007/s10115-024-02256-7

[21] B. Bharadwaj, S. Nayak, and P. K. Panigrahi, "Sentiment analysis for identifying depression through social media texts using machine learning technique," *Big Data and Computing Visions*, vol. 5, no. 2, pp. 102-118, 2025. https://doi.org/10.22105/bdcv.2025.499270.1239

[22] R. Sharma, B. Singh, and A. Khamparia, *Machine learning and generative AI techniques for sentiment analysis with applications. In Generative artificial intelligence for biomedical and smart health informatics.* Cham, Switzerland: Springer, 2025.

[23] C. G. Özmen and S. Gündüz, "Comparison of machine learning models for sentiment analysis of big Turkish web-based data," *Applied Sciences*, vol. 15, no. 5, p. 2297, 2025. https://doi.org/10.3390/app15052297

[24] M. Ghonge, T. Kachare, M. Sinha, S. Kakade, S. Nigade, and S. Shinde, "Real time fake note detection using deep convolutional neural network," presented at the 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India. https://doi.org/10.1109/ICCSEA54677.2022.9936084, 2022, pp. 1–6.

[25] M. D. Devika, C. Sunitha, and A. Ganesh, "Sentiment analysis: A comparative study on different approaches," *Procedia Computer Science*, vol. 87, pp. 44-49, 2016. https://doi.org/10.1016/j.procs.2016.05.124