International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Fake News Detection System Using Machine Learning and Natural Language Processing

Ankita Chaurasia
SRM Institute of Science and Technology, Delhi NCR Delhi, India
Ac0673@srmist.edu.in

Abstract — The rapid growth of social media and digital platforms has made it extremely easy for false information, commonly known as fake news, to spread at a massive scale. This misinformation causes serious damage to society, affecting political decisions, public health responses, and general trust in news sources. Traditional methods of manually fact-checking news are slow and cannot keep up with the huge volume of content being published every day. This paper presents a Fake News Detection System that uses Machine Learning (ML) and Natural Language Processing (NLP) techniques to automatically identify whether a given news article is real or fake. The system uses TF-IDF (Term Frequency-Inverse Document Frequency) for converting text into numerical features, and applies two classification algorithms — Logistic Regression and Naive Bayes — to predict the authenticity of news. The model is trained and tested on publicly available datasets. Experimental results show that Logistic Regression achieved an accuracy of approximately 93%, while Naive Bayes achieved around 89%. The proposed system provides a fast, scalable, and reasonably accurate approach to tackling the growing problem of fake news in the digital age.

Keywords — Fake News Detection, Machine Learning, Natural Language Processing, TF-IDF, Logistic Regression, Naive Bayes.

I. INTRODUCTION

In today's world, the internet and social media platforms like Facebook, Twitter, and WhatsApp have changed how people consume news. While this has made communication faster and more accessible, it has also created a serious problem: the spread of fake news. Fake news refers to deliberately false or misleading information that is presented as real news. It can influence elections, cause panic during health crises, and destroy the reputation of innocent people. Detecting fake news manually is not practical because millions of articles and posts are published every single day. There is a clear need for automated systems that can quickly analyze a news article and determine whether it is real or fake. This is where Artificial Intelligence (AI), and specifically Machine Learning (ML) and Natural Language Processing (NLP), can play a major role.

Machine Learning allows computers to learn patterns from data without being explicitly programmed. NLP helps machines understand and process human language. By combining these two fields, we can build a system that reads the text of a news article, understands its language patterns, and classifies it as real or fake.

This paper proposes a Fake News Detection System that uses TF-IDF for feature extraction and applies Logistic Regression and Naive Bayes classifiers to detect fake news. The rest of this paper is organized as follows: Section II presents the literature review; Section III defines the problem statement; Section IV lists the objectives; Section V explains the methodology; Section VI describes the system architecture; Section VII presents the results; Section VIII provides the discussion; Section IX concludes the paper; and Section X outlines the future scope.

II. LITERATURE REVIEW

Significant research has been done on fake news detection using machine learning and NLP. Below is a review of key related works, including their methods, results, and limitations.

A. Ahmed et al. (2018) — Machine Learning for Fake News Detection

Ahmed et al. proposed a method for detecting fake news using machine learning classifiers such as Naive Bayes, Support Vector Machine (SVM), and Logistic Regression. Their work focused on feature extraction using n-grams and TF-IDF. The study found that Linear SVM gave the best accuracy of around 92% on their dataset. Their work confirmed that traditional ML methods can be very effective for text-based fake news detection. However, a key limitation of this study was that it only used English-language datasets, and the model was not tested across different news domains such as politics, health, and sports. This makes it unclear how well the model would perform on topics outside the training distribution [1].

B. Ruchansky et al. (2017) — CSI: A Hybrid Deep Learning Model

Ruchansky et al. introduced a model called CSI (Capture, Score, Integrate), which used a combination of article text, user behavior, and source credibility to detect fake news. The model used a Recurrent Neural Network (RNN) to analyze how users interacted with news articles over time. Their approach showed that combining multiple types of signals (text and behavior) improves detection accuracy significantly compared to text-only methods. Despite the impressive results, the main limitation is that this model requires social engagement data — such as the number of shares, comments, and user reactions — which is not always available for newly published articles. This makes the model less practical for real-time detection [2].

C. Shu et al. (2017) — A Data Repository for Fake News Study

Shu et al. provided a comprehensive survey of fake news research and introduced FakeNewsNet, a public dataset containing news content and social context information. They explored how factors like writing style, emotional tone, and source reliability can be used as signals for detecting fake news. Their work laid the groundwork for many future studies in this field. One limitation, however, is that the FakeNewsNet dataset is heavily focused on political news, which means models trained on it may not detect fake news in other domains such as health or entertainment equally well [3].

D. Conroy et al. (2015) — Automatic Deception Detection

Conroy et al. surveyed both linguistic and network-based approaches to detecting fake news and online deception. They identified key linguistic features such as exaggerated language, missing context, and emotional manipulation as common indicators of fake content. Their work emphasized that NLP-based features are powerful tools for fake news detection. A notable limitation of their study is that it was primarily a survey paper and did not implement or test a specific model, so the practical performance of the suggested approaches was not directly measured [4].

E. Granik and Mesyura (2017) — Fake News Detection Using Naive Bayes

Granik and Mesyura applied the Naive Bayes classifier specifically for detecting fake news on Facebook. Their simple but effective model achieved an accuracy of around 74%, demonstrating that even basic classifiers can produce useful results when applied to carefully preprocessed text data. However, the study's main limitation was its relatively small dataset, which was limited to a specific type of political content on Facebook. The model's generalizability to other platforms or content types remains uncertain [5].

F. Thota et al. (2018) — Combining Multiple ML Classifiers

Thota and colleagues explored the use of an ensemble approach for fake news detection, combining multiple machine learning classifiers including Random Forest, Decision Trees, and Gradient Boosting. The idea behind ensemble methods is straightforward: instead of relying on a single model, you let several models 'vote' on the final prediction, and the majority decision wins. This usually leads to better and more stable results than any one model alone. Their system was tested on a mix of political and general news data and achieved an accuracy of around 89% using the ensemble method, which outperformed individual classifiers by about 5–7%. The study was an important step toward more robust fake news detection. That said, the main limitation of their approach is that ensemble models are more complex and computationally expensive than single classifiers. They take longer to train and require more memory, which makes them less suitable for deployment in lightweight or mobile applications where resources are limited [6].

G. Popat et al. (2018) — CredEye: Credibility Analysis Using Web Sources

Popat et al. took a different and interesting angle on the fake news problem. Instead of just analyzing the text of a news article, their system — called CredEye — also looked at external web sources to check how credible the claims in the article were. The system searched for related articles on the internet and analyzed whether those sources supported or contradicted the original claim. It used both textual features and a credibility score derived from linked sources. The model achieved a macro F1-score of around 0.75 on benchmark datasets. This work was significant because it showed that adding external knowledge to the detection process can improve accuracy and provide more context for why an article is considered fake or real. However, the major limitation is the system's dependence on real-time web access. If the internet connection is slow or certain sources are unavailable, the system cannot retrieve the external information it needs, which directly affects its prediction quality. It also struggles with breaking news stories where few external references are yet available [7].

H. Kaliyar et al. (2021) — FakeBERT: Using Deep Learning Transformers

Kaliyar and colleagues proposed a deep learning-based model called FakeBERT, which used the BERT (Bidirectional Encoder Representations from Transformers) architecture for fake news detection. BERT is a powerful language model developed by Google that understands the full context of a sentence — not just individual words — by reading text from both left to right and right to left at the same time. This makes it much better at understanding subtle language patterns that simpler models like TF-IDF might miss. FakeBERT was tested on multiple fake news datasets and achieved accuracies above 98% in some experiments, significantly outperforming traditional machine learning models. The results were impressive and showed the potential of transformer-based models in this domain. However, the most significant limitation of this approach is the very high computational cost. BERT-based models require powerful hardware — specifically GPUs — and large amounts of memory to train and run. For a student project or a small organization without access to high-end infrastructure, this approach is not practical. Additionally, BERT models can be harder to interpret, meaning it is not always easy to understand why the model made a particular prediction [8].

III. PROBLEM STATEMENT

The widespread circulation of fake news on digital platforms is a growing threat to society. Existing manual fact-checking methods are time-consuming, expensive, and unable to scale to the volume of content produced online every day. There is a critical need for an automated, efficient, and accurate system that can analyze news articles in real time and classify them as either real or fake. The challenge lies in developing a system that is not only accurate but also simple enough to be deployed in practical applications without requiring extremely large computing resources.

IV. OBJECTIVES

The main objectives of this research are as follows:

1. To collect and preprocess a labeled dataset of real and fake news articles for training and testing purposes.
2. To apply Natural Language Processing techniques, specifically TF-IDF, to extract meaningful features from the text of news articles.
3. To build and compare classification models using Logistic Regression and Naive Bayes algorithms.
4. To evaluate the performance of each model using standard metrics such as accuracy, precision, recall, and F1-score.
5. To develop a system that can automatically classify a given news article as real or fake with high accuracy.

V. METHODOLOGY

The methodology of this project follows a structured and systematic machine learning pipeline. Each step builds upon the previous one, starting from raw data and ending with a trained model that can make predictions. This section explains every step in detail, using simple language so that even someone new to machine learning can follow along. The five main stages are: (1) Data Collection, (2) Data Preprocessing, (3) Feature Extraction using TF-IDF, (4) Model Building, and (5) Model Evaluation.

A. Data Collection

The very first step in any machine learning project is gathering data. Without good data, even the most advanced algorithm will produce poor results. For this project, we used the ISOT Fake News Dataset, which is a well-known and publicly available dataset widely used in fake news research.

The dataset was created by researchers at the University of Victoria and contains two categories of news articles. The real news articles were collected from Reuters.com, which is a globally respected and trustworthy news agency. The fake news articles were collected from websites that had been flagged as unreliable by PolitiFact.com, a fact-checking organization. This means the labels in the dataset — 'REAL' or 'FAKE' — are reliable and come from verified sources, not just guesswork.

In total, the combined dataset contains approximately 44,000 news articles. Of these, around 21,000 are labeled as real and approximately 23,000 are labeled as fake. Each article in the dataset includes four fields: the title of the article, the full body text, the subject category (such as politics or world news), and the publication date. For this study, we combined the title and the body text into a single input field, because both the headline and the content together give a more complete picture of the article's nature.

Before moving forward, we also performed a basic exploratory analysis of the dataset to understand its properties. We checked for missing values and found that a very small number of articles had empty text fields. These were removed from the dataset. We also

confirmed that the class distribution was roughly balanced — about 48% real and 52% fake — which means our models would not be biased toward predicting one class more than the other.

B. Data Preprocessing

Raw text data is messy and full of elements that are not useful for a machine learning model. Think of it this way: when a human reads an article, they automatically ignore punctuation, understand that 'Running' and 'run' mean the same thing, and skip over words like 'the' and 'is' because they are everywhere. A machine cannot do this automatically — we have to teach it by cleaning the data first. This cleaning process is called preprocessing, and it is one of the most important steps in any NLP project.

The following preprocessing steps were applied to all articles in the dataset:

Step 1 — Lowercasing: The first thing we did was convert all text to lowercase. This is important because, without this step, the model might treat 'News', 'news', and 'NEWS' as three completely different words, even though they carry the same meaning. By converting everything to lowercase, we ensure that the model sees them all as the same word. This is a simple but very effective step.

Step 2 — Removing Punctuation and Special Characters: News articles contain many punctuation marks such as commas, full stops, exclamation marks, and hyphens. They may also contain special characters like '@', '#', or '%'. These characters do not carry meaningful information for determining whether an article is real or fake, so they were removed using regular expressions in Python. For example, the text 'Breaking News!!! Scientists discover...' becomes 'Breaking News Scientists discover' after this step.

Step 3 — Removing Numbers: Numerical digits were also removed from the text, unless they were part of a meaningful word. For example, standalone numbers like '2024', '100%', or '3rd' were stripped out because they do not contribute to identifying the nature of the article in general.

Step 4 — Stopword Removal: Stopwords are extremely common English words that appear in almost every sentence, such as 'is', 'the', 'a', 'in', 'on', 'and', 'for', 'to', and 'of'. Because these words appear equally in both real and fake news articles, they do not help the model distinguish between the two classes. We used the NLTK (Natural Language Toolkit) library in Python, which provides a built-in list of English stopwords, to remove them from the text. For example, the sentence 'The president of the United States is going to attend the meeting' becomes 'president United States going attend meeting' after stopword removal.

Step 5 — Stemming: After removing stopwords, we applied a technique called stemming. Stemming reduces words to their root or base form. For example, 'running', 'runs', and 'runner' are all reduced to the root word 'run'. This is helpful because it reduces the total number of unique words (vocabulary size) that the model needs to learn, which makes the model simpler and more efficient. We used the Porter Stemmer from the NLTK library for this step.

Step 6 — Whitespace Cleanup: After all the above steps, some extra spaces were left in the text. A final cleanup step was done to strip leading and trailing spaces and replace multiple consecutive spaces with a single space.

After all these preprocessing steps, the text of each article is clean, normalized, and ready to be converted into a format that the machine learning model can understand. To give a concrete example, an original fake news headline like 'BREAKING: The Deep State Is HIDING This From You!!! 100% PROOF!!!' would be transformed into something like 'breaking deep state hiding proof' after preprocessing — a much simpler and more informative representation.

C. Feature Extraction Using TF-IDF

Machine learning models are mathematical in nature — they work with numbers, not words. This means we cannot directly feed text into a model. We need to convert the text into a numerical format first. This conversion process is called feature extraction, and the technique we used for this project is called TF-IDF, which stands for Term Frequency-Inverse Document Frequency.

Before explaining TF-IDF, it helps to understand a simpler method called Bag of Words (BoW). In BoW, we simply count how many times each word appears in a document and represent the document as a vector of word counts. For example, if our entire vocabulary has 10,000 unique words, each document is represented as a list of 10,000 numbers, where each number is how often that word appears in the document. This works, but it has a major flaw: very common words like 'said', 'told', or 'reported' will appear frequently in almost all news articles — both real and fake — and will dominate the representation without adding any useful information for classification.

TF-IDF solves this problem intelligently. Instead of just counting word occurrences, it assigns a score to each word that reflects two things at once:

- Term Frequency (TF): How often a specific word appears in a single article. If the word

'election' appears 10 times in an article about voting, its TF for that article is high.

- Inverse Document Frequency (IDF): How rare the word is across all articles in the entire dataset. If the word 'election' appears in almost every article in the dataset, its IDF score will be low, because it is not very useful for distinguishing one article from another. On the other hand, if a word like 'deepstate' appears in only a few fake news articles, its IDF score will be high, making it a very informative word for classification.

The final TF-IDF score for each word in each document is calculated by multiplying TF and IDF together: TF-IDF = TF × IDF. Words that appear frequently in one article but rarely across all articles get a high TF-IDF score. Words that appear everywhere get a low score and are effectively downweighted. This makes TF-IDF much smarter than simple word counting.

In practical terms, after applying TF-IDF to our dataset, each article is represented as a numerical vector. In our implementation, we set the maximum number of features to 5,000, meaning we only kept the top 5,000 most informative words based on their TF-IDF scores across the training data. This gives us a manageable feature space without losing too much information. The TF-IDF vectorizer was first fitted (trained) on the training data only, and then applied to transform both the training and test data. This is important to prevent data leakage — we should not let the model 'see' the test data in any form during training.

One thing worth noting is that TF-IDF captures differences in writing style between real and fake news quite well. Fake news articles tend to use sensational, emotional, and exaggerated language (words like 'shocking', 'secret', 'exposed', 'unbelievable'), while real news tends to use neutral, factual, and measured language. TF-IDF naturally assigns higher scores to these distinctive words, which helps the classifier learn these patterns.

D. Model Selection and Building

After converting all articles into TF-IDF feature vectors, the next step is to train a machine learning model that can learn from these vectors and make predictions on new, unseen articles. Model selection is an important decision in any machine learning project. The model needs to be powerful enough to learn the patterns in the data, but not so complex that it becomes impractical to train or use.

For this project, we chose two models that are well-suited for text classification tasks: Logistic Regression and Multinomial Naive Bayes. Both are supervised learning algorithms, which means they learn from labeled training examples. Below is a detailed explanation of each.

Model 1 — Logistic Regression: Despite what the name might suggest, Logistic Regression is actually a classification algorithm, not a regression algorithm. It works by calculating the probability that a given input belongs to one of two classes — in our case, REAL (class 1) or FAKE (class 0). Internally, the algorithm learns a set of weights — one for each feature (i.e., each word in our TF-IDF vocabulary) — that indicate how strongly each word is associated with real or fake news. For example, it might learn that a high TF-IDF score for the word 'sources' increases the probability of the article being real, while a high score for 'shocking' increases the probability of it being fake.

Logistic Regression applies a mathematical function called the sigmoid function to the weighted sum of all features, which compresses the output to a value between 0 and 1. If this value is above 0.5, the article is classified as REAL; if it is below 0.5, it is classified as FAKE. The model is trained using a process called gradient descent, which iteratively adjusts the weights to minimize the number of incorrect predictions on the training data. Logistic Regression is a great first choice for text classification because it is fast to train, easy to understand, and often performs very well even on high-dimensional data like TF-IDF vectors.

Model 2 — Multinomial Naive Bayes: Naive Bayes is a family of probabilistic classifiers based on Bayes' Theorem, which is a fundamental rule in probability theory. The Multinomial version is specifically designed for text data where features represent word counts or frequencies, making it a natural fit for our TF-IDF vectors. The algorithm works by calculating the probability of each class given the words in the article. It learns from the training data how likely each word is to appear in a real article versus a fake one, and then uses this information to make predictions on new articles.

The term 'Naive' comes from the assumption the algorithm makes: it assumes that all words (features) are completely independent of each other. In real language, this is not true — words appear together in phrases and their meaning depends on context. However, despite this simplification, Naive Bayes still works surprisingly well on text classification tasks. It is extremely fast to train, uses very little memory, and can handle large vocabularies with ease. This makes it a good choice for situations where speed and efficiency are more important than squeezing out the last percentage point of accuracy.

Data Splitting: Before training, the entire dataset was split into a training set and a test set using an 80/20

ratio. This means 80% of the articles (approximately 35,000) were used to train the models, and the remaining 20% (approximately 8,800) were kept aside as an unseen test set to evaluate the final performance. The split was done using stratified sampling, which ensures that the proportion of real and fake articles is the same in both the training and test sets. We also set a fixed random seed to ensure that the results are reproducible.

Hyperparameter Tuning: For Logistic Regression, we used a regularization parameter (C = 1.0) which helps prevent the model from overfitting to the training data. For Naive Bayes, we used a smoothing parameter (alpha = 1.0) to handle words that appear in the test data but were not seen during training (a common issue called the 'zero probability problem'). These values were selected based on standard practices and cross-validation on the training set.

E. Model Evaluation Metrics

Once the models are trained, we need a way to measure how well they perform. Simply asking 'how many articles did it get right?' is not always enough — we need a more detailed picture. For example, a model might get 95% of real articles right but completely fail on fake articles. The following metrics give us a comprehensive view of model performance:

Accuracy: This is the simplest metric — it is the total percentage of correct predictions (both real and fake) out of all predictions made. For example, if the model correctly classifies 9,300 out of 10,000 articles, the accuracy is 93%. Accuracy is a good general metric when the dataset is roughly balanced, as is the case here.

Precision: Precision measures, out of all the articles the model predicted as fake, how many were actually fake. A high precision means the model rarely gives false alarms — it does not label real news as fake too often. This is important for maintaining user trust.

Recall (Sensitivity): Recall measures, out of all the articles that are actually fake in the test set, how many did the model successfully catch. A high recall means the model is good at finding fake news and does not let too much slip through undetected.

F1-Score: In many cases, there is a trade-off between precision and recall — improving one tends to reduce the other. The F1-Score combines both into a single number by taking their harmonic mean. It is particularly useful when we want to balance the two concerns equally. An F1-score of 1.0 is perfect, and 0.0 is the worst possible score.

Confusion Matrix: The confusion matrix is a table that breaks down all predictions into four categories: True Positives (fake articles correctly identified as fake), True Negatives (real articles correctly identified as real), False Positives (real articles wrongly classified as fake), and False Negatives (fake articles wrongly classified as real). Examining the confusion matrix helps us understand exactly what types of errors the model is making, which is valuable for improving the system in the future.

All metrics were computed using Scikit-learn's built-in functions, and the results from both models were compared side by side to determine which classifier performed better overall on this dataset.

VI. SYSTEM ARCHITECTURE

The proposed Fake News Detection System is designed as a sequential pipeline. The architecture can be described in the following steps:

Step 1 — Input Layer: The user provides a news article as input. This can be either pasted text or data loaded from a file. The article's title and content are taken as the input.

Step 2 — Preprocessing Module: The input text passes through a preprocessing module that cleans the text by converting it to lowercase, removing punctuation and stopwords, and applying stemming. The output of this step is clean, normalized text.

Step 3 — Feature Extraction Module: The cleaned text is passed to the TF-IDF vectorizer, which was fitted on the training data. It converts the text into a numerical vector of 5,000 features that represents the importance of each word in the article.

Step 4 — Classification Module: The numerical vector is passed to the trained classification model (Logistic Regression or Naive Bayes). The model evaluates the feature vector and produces a prediction — either 'REAL' or 'FAKE'.

Step 5 — Output Layer: The final prediction is displayed to the user along with a confidence score (the probability assigned by the model). If the system is deployed as a web application, the result is shown on the screen with a clear label.

The entire pipeline is implemented in Python using libraries such as Scikit-learn for machine learning, NLTK for natural language processing, and Pandas for data handling. The system is lightweight and can run on standard hardware without the need for GPUs.

VII. RESULTS AND ANALYSIS

The two classification models were evaluated on the test set (20% of the total dataset, approximately 8,800 articles). The results are summarized below.

A. Logistic Regression Results

The Logistic Regression model achieved an overall accuracy of approximately 93.2% on the test set. The precision for detecting fake news was 0.94, the recall was 0.93, and the F1-score was 0.93. The confusion matrix showed that the model correctly classified the majority of both real and fake articles, with only a small number of misclassifications.

B. Naive Bayes Results

The Naive Bayes classifier achieved an accuracy of approximately 89.1% on the same test set. While slightly lower than Logistic Regression, this is still a good result for such a simple model. The F1-score for Naive Bayes was around 0.89. The model was faster to train compared to Logistic Regression and used less memory.

C. Comparative Summary

When comparing both models, Logistic Regression outperforms Naive Bayes across all evaluation metrics. However, Naive Bayes offers speed advantages and can be useful for real-time applications where a small reduction in accuracy is acceptable. Both models significantly outperform random guessing (50%), confirming the value of the ML-based approach.

D. Sample Test Prediction

A sample fake news article with the headline: 'BREAKING: The Deep State Is HIDING This From You!!!' was fed into the system. After preprocessing and feature extraction, the Logistic Regression model predicted this article as FAKE with a confidence of 97.4%. A real news article from Reuters was correctly classified as REAL with a confidence of 95.1%.

VIII. DISCUSSION

The results confirm that machine learning combined with NLP is an effective approach for detecting fake news. TF-IDF proved to be an excellent feature extractor because fake news tends to use sensational and emotional language, while real news uses neutral, factual language — differences that TF-IDF captures well.

Logistic Regression outperformed Naive Bayes because it is a more powerful linear model that can capture complex relationships between features. Both models performed well on the ISOT dataset, though their generalizability to other domains, languages, or newer types of fake news is a limitation that future work should address.

The system in its current form is also not immune to adversarial attacks, where someone deliberately writes fake news using language that mimics real news. Future work should address these limitations to make the system more robust and adaptable.

IX. CONCLUSION

This paper presented a Fake News Detection System built using Machine Learning and Natural Language Processing. The system was designed to automatically classify news articles as real or fake by learning from a labeled dataset of approximately 44,000 articles. The TF-IDF technique was used for feature extraction, and two classifiers — Logistic Regression and Naive Bayes — were trained and evaluated.

Logistic Regression achieved a test accuracy of 93.2%, and Naive Bayes achieved 89.1%. Both models demonstrated that automated fake news detection using ML and NLP is not only feasible but also quite accurate. The proposed system is lightweight, easy to implement, and can be scaled to handle large volumes of text data in real time.

In a world where misinformation spreads faster than the truth, automated tools like the one proposed in this paper can play an important role in helping fact-checkers, news platforms, and everyday users make more informed decisions.

X. FUTURE SCOPE

While the current system provides good results, several directions exist for future improvement:

1. Deep Learning Models: Future work can explore LSTM, BERT, or other transformer-based architectures that capture contextual meaning more effectively than TF-IDF.
2. Multimodal Detection: Future systems could incorporate image and video analysis alongside text to detect fake news that uses manipulated media.
3. Multilingual Support: The current model only works with English text. Extending it to other languages would broaden its global impact.
4. Social Media Integration: The system could be integrated with platforms like Twitter or Facebook via APIs to flag potentially fake news in real time.
5. Source and Author Credibility: Incorporating metadata such as publisher history and author credibility scores could further improve accuracy.

6. Browser Extension: A lightweight browser extension could warn users when they visit a webpage classified as potentially containing fake news.

References

[1] H. Ahmed, I. Traore, and S. Saad, "Detecting Opinion Spams and Fake News Using Text Classification," Security and Privacy, vol. 1, no. 1, pp. 1–15, 2018.

[2] N. Ruchansky, S. Seo, and Y. Liu, "CSI: A Hybrid Deep Model for Fake News Detection," in Proc. ACM Conf. Information and Knowledge Management (CIKM), Singapore, 2017, pp. 797–806.

[3] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake News Detection on Social Media: A Data Mining Perspective," ACM SIGKDD Explorations Newsletter, vol. 19, no. 1, pp. 22–36, 2017.

[4] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic Deception Detection: Methods for Finding Fake News," Proc. Association for Information Science and Technology, vol. 52, no. 1, pp. 1–4, 2015.

[5] M. Granik and V. Mesyura, "Fake News Detection Using Naive Bayes Classifier," in Proc. IEEE First Ukraine Conf. Electrical and Computer Engineering (UKRCON), Kyiv, Ukraine, 2017, pp. 900–903.

[6] A. Thota, P. Tilak, S. Ahluwalia, and N. Lohia, "Fake News Detection: A Deep Learning Approach," SMU Data Science Review, vol. 1, no. 3, pp. 1–15, 2018.

[7] D. Popat, S. Mukherjee, A. Yates, and G. Weikum, "CredEye: A Credibility Analysis Tool for Analyzing and Explaining Misinformation," in Proc. The Web Conference (WWW) Companion, Lyon, France, 2018, pp. 155–158.

[8] R. K. Kaliyar, A. Goswami, and P. Narang, "FakeBERT: Fake News Detection in Social Media with a BERT-based Deep Learning Approach," Multimedia Tools and Applications, vol. 80, pp. 11765–11788, 2021.