

AN INTELLIGENT MACHINE LEARNING FRAMEWORK FOR FAKE NEWS DETECTION USING TF-IDF FEATURE EXTRACTION AND ENSEMBLE LEARNING

Faisal Shahzad^{*1}, Laiba Urooj², Ayesha Maryam³

^{*1,3}Department of Computer Science, Bahauddin Zakariya University, Multan 60800, Pakistan.

²Department of Computer Science, University College of Management & Sciences, Khanewal 58150, Pakistan

^{*1}faisal.shahzad.research@gmail.com

DOI: <https://doi.org/10.5281/zenodo.21868077>

Keywords

Fake News Detection; Machine Learning; TF-IDF; XGBoost; Logistic Regression; Random Forest; Text Classification;

Article History

Received: 01 January 2026

Accepted: 16 March 2026

Published: 31 March 2026

Copyright @Author

Corresponding Author: *

Faisal Shahzad

Abstract

News moves quickly these days thanks to websites and social platforms. That rush? It brings trouble—made-up reports spreading like they're true. Telling fact from fiction isn't always easy anymore. Earlier efforts leaned on machines trained to tell real stories apart from fake ones. These systems often cleaned up text first, then turned words into numbers using methods such as TF-IDF before sorting them. This new research builds on earlier findings. Machine learning tools spot false stories by pulling information directly from files named Fake.csv and True.csv. Out of 44,898 entries, slightly more than half were fake—23,481 tagged as 0. Real articles made up the rest, marked with a 1. Instead of keeping them separate, the title and body text were joined into one field. Cleaning came next, stripping away messy bits from that combined content. After tidying things up, each word was transformed through TF-IDF weighting. We began with familiar steps. First, prep work. Then, the transformation of words to numbers. A learner was selected. The learner was then taught using the prepared data. Finally, we measured how well the learner had absorbed the lessons. In this version, four-fifths of the data is dedicated to the learning process, while the remaining one-fifth is reserved; it will be used for later verification checks. Fifty thousand features shaped the input space. Words alone mattered, along with pairs sitting side by side. Common fillers like "the" or "of" got dropped out early. Terms appearing too often across sources disappeared when shared by over 80 percent. Rare ones seen fewer than twice vanished too. Frequencies were carefully adjusted with logarithmic scaling to temper any overwhelming dominance. Six machine learning models underwent thorough evaluation. Logistic Regression initiated the process, succeeded by Decision Tree, and then Random Forest. Gradient Boosting followed, with Passive Aggressive Classifier appearing before XGBoost concluded the series. Performance evaluations relied on standard metrics, though each figure offered unique insights. Training and test accuracy started us off, yet precision and recall, including F1-score, remained under scrutiny. Confusion matrices filled in what the averages missed, offering glimpses into where guesses went right. - or didn't. Among them all, one stood apart without flash or noise. XGBoost reached a 99.86% mark when tested, not guessed, but built instead on patterns found during real trials. Its precision landed at 99.79%, showing few false alarms.

Recall climbed even higher, hitting 99.91%, meaning it caught nearly all true cases thrown its way. The balance between these two settled neatly into an F1-score of 99.85%, firm and unshaken. No model ran closer. Thirteen errors stood out when XGBoost faced 8,980 test cases in the confusion matrix. Because of that, combining TF-IDF with powerful learners such as XGBoost proves effective at telling false stories apart from true ones here.

1. INTRODUCTION

Out of nowhere, the web reshaped where we grab our updates—and toss them into the world. Once upon a time, broadcasts, print sheets, or radio waves carried every headline; today, posting or passing along tales takes mere heartbeats across online spaces. Blazing fast? Sure. Yet that rush brings sharp edges too. Lately, false reports have started gnawing at public trust—a quiet fear spreading behind screens. Later on, incorrect content pretends to be true journalism, tricking many. Because unchecked tales move fast now, the report highlights how dangerous this is—machine learning steps in behind the scenes quietly to catch fakes early. When lies travel fast, they shift what people believe, weaken trustworthy outlets, stir doubt, and leave audiences unsure who to trust. Telling real from fake news? Tough job. Often these stories sound believable—loaded with strong emotions, flashy titles, distorted truths, and sometimes pure invention—all built to catch your eye. Because the web fills up fast each day, checking everything one by one doesn't work. Here's when tech steps into the picture instead. Experts now lean on tools like machine learning and methods for understanding human speech, aiming straight at the root of misleading posts. From raw files to working code—that's where this one begins. Grabbing the data kicks things off right away. Once found, each file loads straight into memory for handling. Labels get attached so the real stands apart from the fake. Merging sources happens before anything else shifts. A quick mix reshuffles rows just enough to break patterns. Empty spots pop up sometimes; those vanish fast. Raw words turn into structured pieces ready for next phases. Text gets stripped down—symbols go, spacing tightens. Numbers step in when words leave, thanks to TF-IDF tricks. Training fires up using what the vectors feed it. Accuracy shows up only after careful scoring

rounds. Not just one guess—the models race against themselves. Confusion charts map wrong turns in clear boxes. The winner emerges by surviving every test run. Even typed sentences pass through at the very end. Past blueprints inspired this flow—not copied but rebuilt. It starts with choices and ends with proof that it works. What stands out here? A complete fake news detection system made step-by-step in Python. Starting fresh, shaping each part carefully. The data used is split evenly—close to forty-five thousand real and false articles. Attention went deep into feature design, pulling meaning from single words and word pairs within TF-IDF structures. Not limiting things to a single idea, testing unfolded across six distinct machine learning approaches, mixing traditional algorithms with grouped strategies. Results shifted clearly when scores appeared—one model rising above the others. Accuracy peaked under XGBoost's handling. Even with everything else done, there was still room for more—like adding a way to let people write their own news snippets and see predictions right away. Taken together, this approach pushes past older work focused only on comparing methods such as logistic regression, Decision Tree, Random Forest, Passive Aggressive Classifier, Gradient Boosting, or XGBoost.

2. Related Works

One way to spot false stories online involves tools like machine learning and language processing methods. Instead of just listing techniques, some researchers actually tested them step by step. Jouhar and the team took real and fake articles, cleaned the text first, then pulled out word patterns using a method called TF-IDF. After that, they ran several models—like Logistic Regression and Decision Trees—to see which one caught fakes best. Among those tried, XGBoost stood out with clearer results than others. What makes their setup useful is how it laid down steps later work can

follow. Though many parts were combined, the core flow stayed straightforward. That earlier design now helps shape what comes next in this area. Older types of machine learning—like Logistic Regression or Decision Trees—show up often in research on spotting false news. Instead of stacking methods unthinkingly, some researchers prefer testing one against another. For instance, Sharma and the team put Naive Bayes to work alongside Random Forest and a few others. What stands out is how frequently the passive-aggressive classifier appears across experiments. Khanam's group leaned into XGBoost, then checked results using SVM and tree-based models. Not far off, Pandey's analysis brought together KNN, SVM, and more classic tools for judging digital headlines. Each study picks its own mix, yet patterns quietly emerge. Fake news research often relies on the ISOT collection, given its mix of authentic and fabricated articles complete with titles, body text, topic labels, and dates. Instead of separate pieces, earlier work by Ahmed, Traore, and Saad leaned on classifying written content through n-gram models [5], [6]. Because of that approach, merging the title with the article text here strengthens how we handle features from Fake.csv and True.csv. Even though newer methods exist, TF-IDF still helps identify meaningful terms—those frequent in one document yet rare overall. Logistic regression's simplicity means it plays nice with high-dimensional TF-IDF vectors, keeping things speedy. Decision trees, rather than leaning on linear assumptions, carve out boundaries with definitive splits. Random Forest, meanwhile, throws a bunch of those trees into the mix, which cuts down on errors. Gradient Boosting and

XGBoost, zeroing in on the examples that got it wrong, build models one step at a time, each fix patching up the last one's mistakes. The passive-aggressive classifier? It's all about quick adjustments when it gets an answer wrong, making it a solid pick for those huge text datasets where getting results fast is key [1], [7], [9]. Even if earlier research showed good results, spotting false news stays hard since how stories are written, what they cover, and how lies spread keep shifting. Because of this, comparing systems matters, along with clear descriptions of data cleanup steps, honest sharing of measurement outcomes, and later testing on external collections. Approaches attempted range from explainable AI to standard evaluations. Deep networks have been employed, along with transformer-based models. Combined CNN-RNN architectures are also used to enhance both the identification of deceptive content and the understanding of the underlying reasons.

3. Exploratory Data Analysis

The data for this work came from a compressed file called News_dataset.zip. Once unpacked, it revealed two separate spreadsheets: one labeled Fake.csv, another True.csv. Though different in size—23,481 entries versus 21,417—they shared an identical structure. Each row carried four pieces of information: headline, body, topic, and timestamp. Structure stayed consistent across both collections despite their opposing labels. A label column appeared once the data loaded: 0 stood for fake news, while 1 meant real. The structure matched the original study, where both types—fake and genuine—were sorted into groups. During early inspection, that work also mapped how each kind spread across the set [1].

Table 1. Dataset description

Dataset File	Number of Records	Columns
Fake.csv	23,481	title, text, subject, date
True.csv	21,417	title, text, subject, date
Combined Dataset	44,898	title, text, subject, date, label

Table 2. Label distribution

News Type	Label	Number of Articles
Fake News	0	23,481
Real News	1	21,417
Total	-	44,898

The combined dataset was shuffled using `random_state=42` to remove class-order bias and improve reproducibility. Missing values were checked across title, text, subject, date, and label columns. No missing values were found, but

`fillna("")` was still used to ensure safe processing during content combination and text cleaning. Exploratory data analysis is important because data quality directly affects model performance in fake-news detection [1].

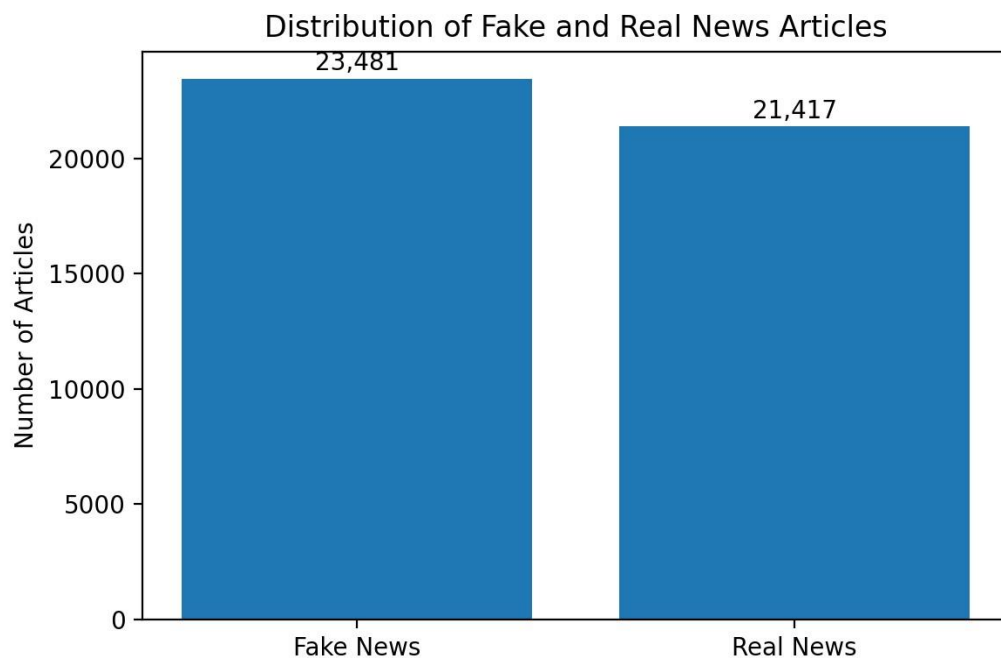


Figure 1. Distribution of fake and real news articles in the dataset.

4. Preprocessing, Vectorization, and Optimization

Before pulling out features, the text got cleaned up first since real news bits often carry uppercase stuff, web links, code tags, digits, odd symbols, and messy spacing. Turning everything lowercase came next—that step also wiped away links, snippets of HTML, numerals, and anything too punctual. Instead of keeping jumbled signs, each strange

mark gave way to cleaner substitutes while squeezing down stretched gaps between words. Afterward, title plus article merged together under one label called content, making room for tidy results saved inside `clean_content`. The earlier study did something close—ditching dots, figures, and filler words just before transforming text using TF-IDF methods [1].

Table 3. Train-test Split

Dataset Part	Percentage	Number of Samples
Training Set	80%	35,918
Testing Set	20%	8,980
Total Dataset	100%	44,898

Table 4. TF-IDF vectorizer configuration

Parameter	Value
stop_words	English
max_df	0.8
min_df	2
ngram_range	(1, 2)
max_features	50,000
sublinear_tf	True

Table 5. TF-IDF feature matrix shape

Feature Matrix	Shape
Training TF-IDF Matrix	35,918 x 50,000
Testing TF-IDF Matrix	8,980 x 50,000

The model worked with single words and also paired them, since ngram_range was set to (1, 2). Limiting the features to 50,000 allowed a wide vocabulary while maintaining efficiency. Term Frequencies were scaled sublinearly so that very common words wouldn't dominate. Earlier studies on spotting false news back up this mix of TF-IDF and n-grams [1], [5], [6].

5. Model Selection and Proposed Work

One reason behind picking six models lies in their varied ways of learning. Not just one method stands out—each follows a unique path through data. Think of logistic regression shaping straight-

line decisions. Then there are decision trees splitting choices like branches in real time. Instead of relying on single predictors, random forest builds many such trees together. Meanwhile, gradient boosting fine-tunes errors step by step across iterations. Another option, passive aggressive classifier, adjusts rapidly during live updates. XGBoost also uses boosting but refines it further under pressure. All these matches exactly what was tested earlier in the original study. That work measured performance head-to-head among identical setups. The results pointed firmly to XGBoost leading in spotting false reports [1].

Table 6. Machine learning models and main parameters

Model	Main Parameters Used
Logistic Regression	max_iter=1000
Decision Tree	random_state = 42
Random Forest	n_estimators=100, random_state=42, n_jobs=-1
Gradient Boosting	random_state=42
Passive Aggressive Classifier	max_iter=1000, random_state=42

XGBoost Model	n_estimators=150, max_depth=6, random_state=42	learning_rate=0.1, eval_metric=logloss,
---------------	--	--

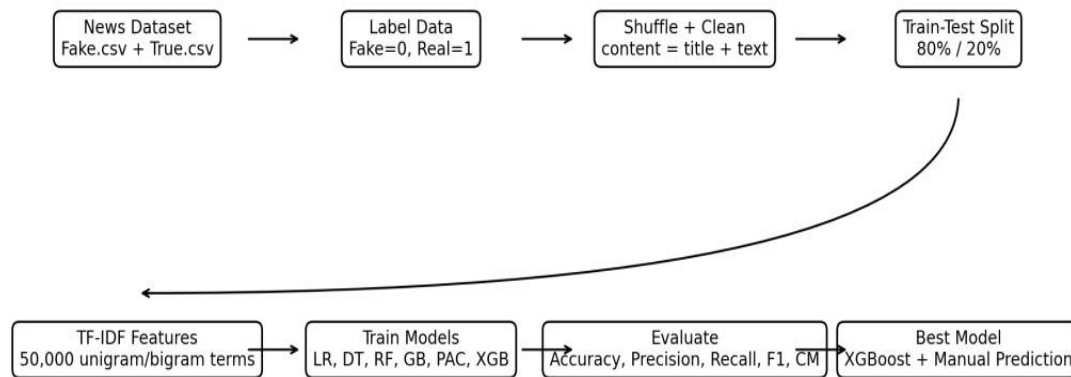


Figure 2. Proposed workflow of the fake news detection system.

5.1 Logistic Regression

A solid starting point came from using Logistic Regression for two-class predictions. Because TF-IDF vectors often have many zero values, this method handles separation well across wide data fields. Linear patterns emerge clearly even when most features are empty. Clarity matters just as much as performance—so the original study picked it too [1].

5.2 Decision Tree

A branching path through features defines how a Decision Tree makes choices, splitting data step by step. When the structure grows deep, it risks mirroring noise instead of patterns. Non-straight relationships are within its reach, yet complexity demands caution. Earlier tests on spotting false stories included this method, showing up in past benchmarks [1], [4].

5.3 Random Forest

One way to spot false news? Try many trees instead of just one. Each tree votes, yet together they smooth out mistakes any single one might make alone. This method builds a forest of 100 trees, each working at once thanks to full CPU use. When grouped smartly, these forests often beat simpler models in telling truth from lies. Their strength comes from variety—randomness in data

and splits keeps them from memorizing too much [7].

5.4 Gradient Boosting

One after another, Gradient Boosting puts together models. Not every model gets it right at first—later ones step in where earlier versions slipped up, nudging accuracy higher. In the original study, this method stood alongside others tested head-to-head [1].

5.5 Passive-Aggressive Classifier

Most times it just sits quiet if the guess hits right. When wrong, it jumps in fast to adjust. Text that spreads thin finds a friend in this classifier. Old work on false stories saw good results using its way of sorting words [1], [2].

5.6 XGBoost

Getting ahead with boosted trees? XGBoost fits the bill—built for speed and performance. Structure matters here: it shines when sorting categories, especially after tweaks showed top marks in recent tests. Turns out, earlier research already flagged it as the go-to model. When fighting false stories, pushing gradients through boosting turns out sharp results [1], [3].

6. Metric Selection

Not every model shines when tested on fresh examples. Performance checks leaned on how well each one labeled training samples correctly, plus results from new data trials. One thing stood out—how often true claims got flagged right mattered a lot. Another point? Spotting false stories without missing many played a big role too. The mix of

these outcomes helped weigh consistency across different measures. Confusion matrices added clarity by showing where mistakes clustered. Researchers have used such indicators before to spot dishonest headlines. What tipped the scales most? How high scores stayed when facing unknown entries during trial runs.

Table 7. Confusion matrix terminology used in this study

Term	Meaning
True Fake	Fake news correctly predicted as fake
Fake Predicted as Real	Fake news incorrectly predicted as real
Real Predicted as Fake	Real news incorrectly predicted as fake
True Real	Real news correctly predicted as real

7. Model Evaluation and Result Analysis

One after another, each model learned from the TF-IDF training matrix before being checked against the testing version. Starting fresh, the evaluation routine applied the model and then produced forecasts for both the training and test sets. From there, it measured performance, pulling together the most important scores. Midway through, a detailed breakdown showed up on

screen—no fanfare, just numbers laid out. What came back included three pieces: a summary dictionary, a grid of correct and mistaken labels, and the finalized model itself—instead of relying on one measure, mismatch patterns and several scoring methods shaped the verdict. Much like what appeared earlier in source material [1], this path stayed consistent without adding twists.

Table 8. Performance comparison of machine learning models

Machine Learning Algorithm	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
Logistics Regression	0.9963	0.9923	0.9889	0.9951	0.9920
Decision Tree	1.0000	0.9960	0.9967	0.9949	0.9958
Random Forest	1.0000	0.9957	0.9937	0.9972	0.9955
Gradient Boosting	0.9977	0.9962	0.9942	0.9979	0.9960
Passive Aggressive Classifier	1.0000	0.9972	0.9958	0.9984	0.9971
XGBoost	0.9999	0.9986	0.9979	0.9991	0.9985

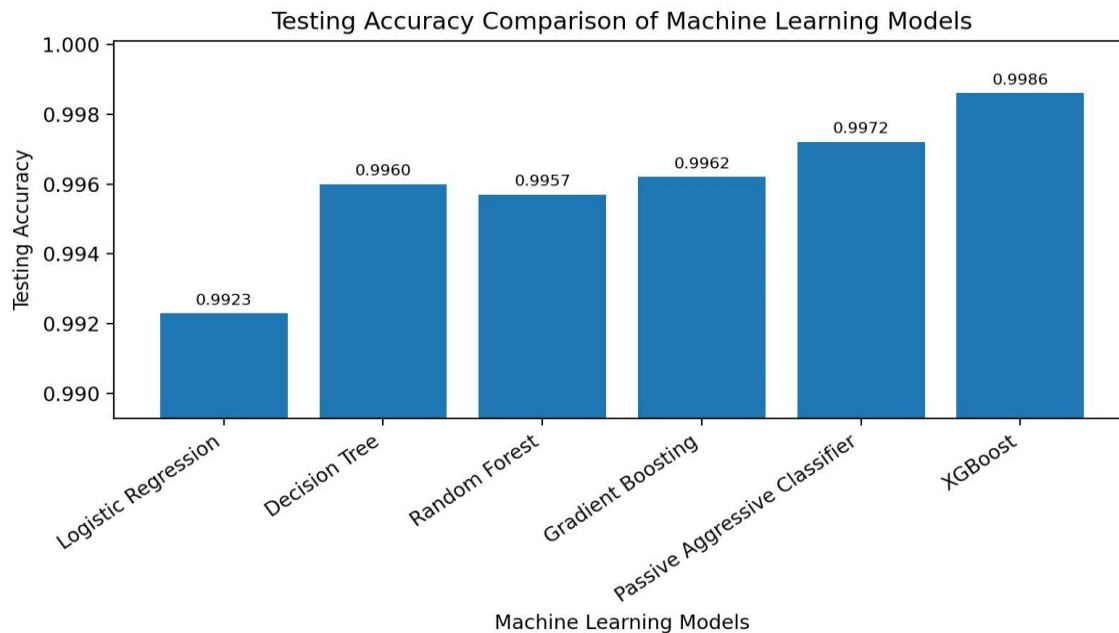


Figure 3. Testing accuracy comparison of all trained models

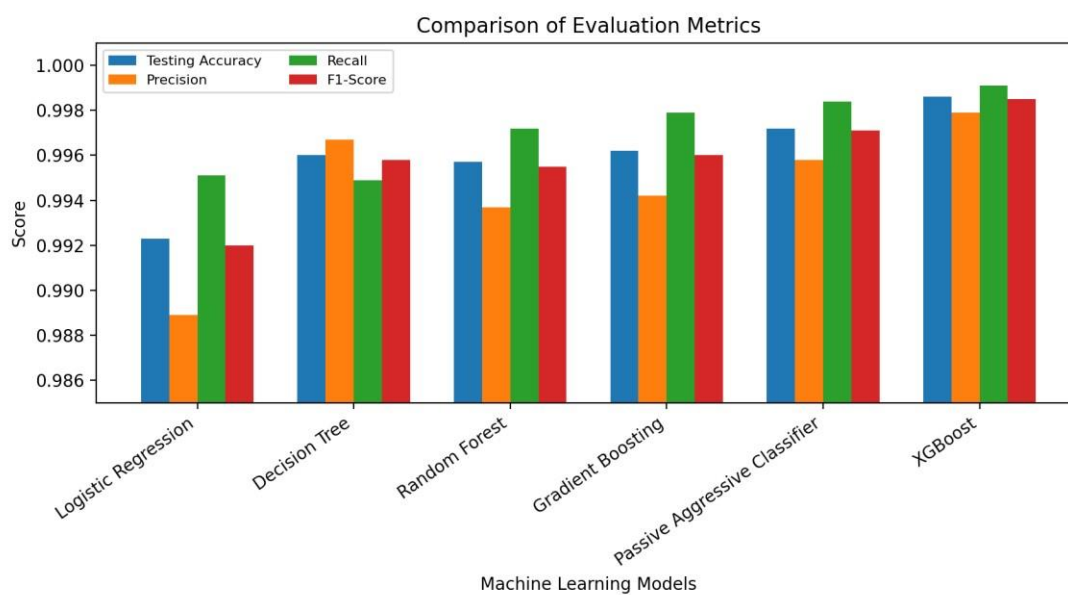


Figure 4. Precision, recall, F1-score, and testing accuracy comparison

Every model turned out well. What stands out is Logistic Regression hitting 99.23% on test data—a sign that TF-IDF works very well here. Close behind, Decision Tree scored 99.60%, while Random Forest came in at 99.57%. Not far off, Gradient Boosting reached 99.62%. One surprise was Passive Aggressive landing at 99.72%, taking

second place. At the top, XGBoost hit 99.86% accuracy, along with near-perfect scores: 99.79% precision, 99.91% recall, and an F1-score of 99.85%. So here it goes—XGBoost came out on top again in the new round of tests. Just like before, when the original work showed it handled

test data better than the others picked for comparison [1].

Table 9. Confusion matrix values for all models

Model	True Fake	Fake Predicted as Real	Real Predicted as Fake	True Real	Total Errors
Logistics Regression	4,648	48	21	4,263	69
Decision Tree	4,682	14	22	4,262	36
Random Forest	4,669	27	12	4,272	39
Gradient Boosting	4,671	25	9	4,275	34
Passive Aggressive Classifier	4,678	18	7	4,277	25
XGBoost	4,687	9	4	4,280	13

XGBoost proves stronger, backed by the confusion matrix. Correctly spotting 4,687 fake pieces as fake marks its edge. Real news? It tagged 4,280 correctly. Only nine fakes slipped through and were labeled real. Four genuine items got flipped—

tally comes from 8,980 tested cases. Second place went to the Passive Aggressive Classifier, making 25 mistakes overall. Earlier studies already hinted that methods like boosting tend to work well spotting false news—this time, XGBoost backs that up again [1], [3], [7].

Confusion Matrices of All Models

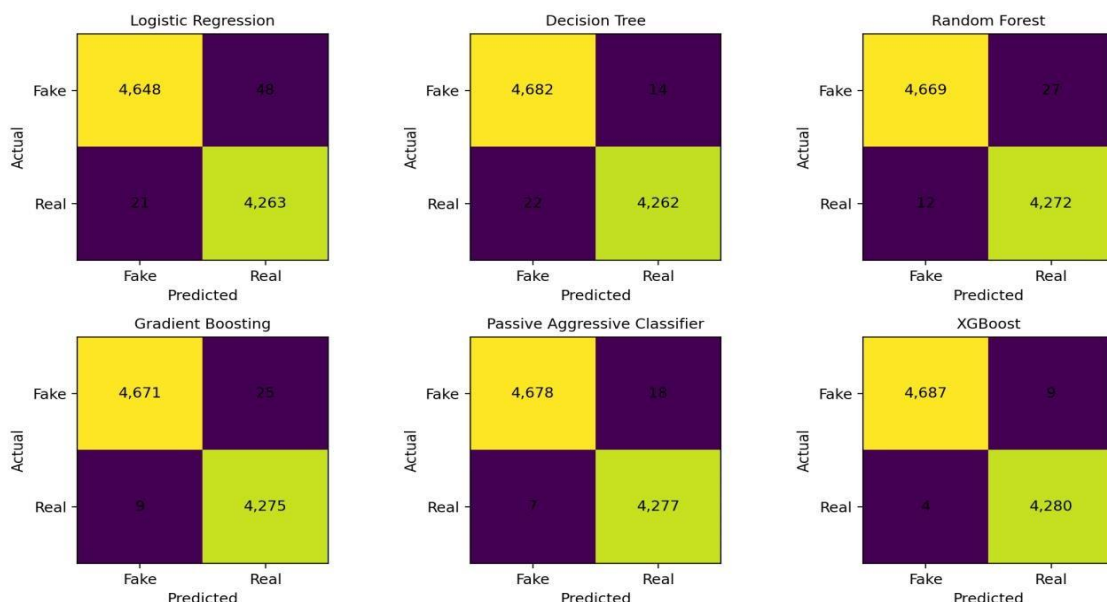


Figure 5. Confusion matrices generated for all machine learning models.

Table 10: Model ranking based on testing accuracy

Rank	Model	Testing Accuracy	Total Errors
1	XGBoost	0.9986	13
2	Passive-Aggressive Classifier	0.9972	25
3	Gradient Boosting	0.9962	34
4	Decision Tree	0.9960	36
5	Random Forest	0.9957	39
6	Logistic Regression	0.9923	69

7.1 Best Model Selection and Manual Input Prediction

Later on, every model outcome went into a single dataframe and was then written out as model_results.csv. Picking the top performer happened without manual input, guided purely by test accuracy scores. That leading method turned

out to be XGBoost, landing a test score of 0.9986. Such hands-off picking aligns with standard comparison tactics across machine learning studies—multiple models get tested, then the most accurate one wins, backed by references [1] and [14].

Table 11. Best model summary

Item	Updated Result
Best Model	XGBoost
Highest Testing Accuracy	0.9986 / 99.86%
Precision	0.9979 / 99.79%
Recall	0.9991 / 99.91%
F1-Score	0.9985 / 99.85%
Total Test Samples	8,980
Total Errors	13

Table 12. Sample manual prediction result

Sample Input	Model Output	Interpretation
The government announced a new education policy today. The policy aims to improve schools and provide better facilities for students.	Fake News	The model can classify manual input text; however, short general statements may not contain enough context for reliable real-world classification.

Starting with whatever text someone types, the system runs it through cleanup steps just like during setup. After that comes conversion via the learned TF-IDF tool, preparing it for judgment. Then the top-performing model assigns a label based on patterns it knows. When the outcome lands on zero, the verdict shows up as "fake news."

One triggers the opposite—real news appears. One trial entry ended up tagged as fake. That moment needs attention since brief or vague phrases often lack depth when stacked beside complete stories used earlier.

8. Conclusion with Future Scope

This research built a tool to spot false stories by turning words into numbers through TF-IDF and applying smart algorithms that learn patterns. Inside were 44,898 pieces of news—more than half labeled untrue and just under half confirmed true. Training happened on roughly four out of five cases, leaving one-fifth aside strictly for checking results later. From each article, the system pulled out short word sequences, making a list of fifty thousand unique pairs and singles before teaching began. Everything followed earlier work done in code, cleaning text first, then extracting features, trying different models, and measuring outcomes step by step [1].

One after another, six machine learning setups got trained: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, Passive Aggressive Classifier, and then XGBoost. XGBoost stood out because its test accuracy hit 99.86%, along with 99.79% precision, 99.91% recall, and an F1-score of 99.85%. Thirteen misclassifications appeared in the confusion matrix when tested on 8,980 examples—fewer than any other method managed. Not far behind came the Passive Aggressive Classifier, landing a solid 99.72% in testing accuracy. Results like this match earlier work where XGBoost again led performance for spotting false news [1].

Even though outcomes look solid, there are constraints built in. Training and testing happened using one single data pool, meaning real-world performance could slip when faced with fresh articles. Unusually sharp precision might reflect quirks of that particular collection. On top of that, pattern recognition drives decisions—truth checking against outside references isn't part of the process. Later on, researchers might try out data from outside sources. Models like BERT or RoBERTa could step in instead of older methods. Alongside that, embedding techniques may face off against classic TF-IDF approaches. Fine adjustments in model settings would come into play at some point. Making the system's decisions easier to understand would also be useful. One path leads toward building an online tool for spotting false stories. Work done lately backs these ideas—especially those focusing on clarity in

results, advanced classifiers, performance checks, and mixed neural strategies [10], [11], [12], [14], [15].

REFERENCES

- J. Jouhar, A. Pratap, N. Tijo, and M. Mony, "Fake News Detection using Python and Machine Learning," *Procedia Computer Science*, vol. 233, pp. 763-771, 2024, doi: 10.1016/j.procs.2024.03.265.
- Uma Sharma, Sidarth Saran, and Shankar M. Patil, "Fake News Detection using Machine Learning Algorithms," *International Journal of Engineering Research & Technology (IJERT)*, NTASU-2020 Conference Proceedings, Special Issue, 2021.
- Z. Khanam, B. N. Alwasel, H. Sirafi, and M. Rashid, "Fake News Detection Using Machine Learning Approaches," *Journal of Physics: Conference Series*, vol. 1099, no. 1, 012040.
- Shalini Pandey, Sankeerthi Prabhakaran, N. V. Subba Reddy, and Dinesh Acharya, "Fake News Detection from Online Media using Machine Learning Classifiers," *Journal of Physics: Conference Series*, vol. 2161, no. 1, 012027.
- H. Ahmed, I. Traore, and S. Saad, "Detecting Opinion Spams and Fake News Using Text Classification," *Security and Privacy*, vol. 1, no. 1, 2018.
- H. Ahmed, I. Traore, and S. Saad, "Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques," in *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, ISDDC 2017, Lecture Notes in Computer Science*, vol. 10618, Springer, Cham, pp. 127-138, 2017.
- I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, "Fake News Detection Using Machine Learning Ensemble Methods," *Complexity*, 2020, pp. 1-11.
- S. R. Sahoo and B. B. Gupta, "Multiple Features Based Approach for Automatic Fake News Detection on Social Networks Using Deep Learning," *Applied Soft Computing*, vol. 100, 106983, 2021.

- J. C. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto, "Explainable Machine Learning for Fake News Detection," in Proceedings of the 10th ACM Conference on Web Science, pp. 17-26, 2019.
- S. Kumar, R. Asthana, S. Upadhyay, N. Upreti, and M. Akbar, "Fake News Detection Using Deep Learning Models: A Novel Approach," Transactions on Emerging Telecommunications Technologies, vol. 31, no. 2, e3767, 2020.
- 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, no. 8, pp. 11765-11788, 2021.
- Y. Wang, W. Yang, F. Ma, J. Xu, B. Zhong, Q. Deng, and J. Gao, "Weak Supervision for Fake News Detection via Reinforcement Learning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 1, pp. 516-523, 2020.
- M. D. Ibrishimova and K. F. Li, "A Machine Learning Approach to Fake News Detection Using Knowledge Verification and Natural Language Processing," in Advances in Intelligent Networking and Collaborative Systems, pp. 223-234, 2019.
- J. Y. Khan, M. T. I. Khondaker, S. Afroz, G. Uddin, and A. Iqbal, "A Benchmark Study of Machine Learning Models for Online Fake News Detection," Machine Learning with Applications, vol. 4, 100032, 2021.
- J. A. Nasir, O. S. Khan, and I. Varlamis, "Fake News Detection: A Hybrid CNN-RNN Based Deep Learning Approach," International Journal of Information Management Data Insights, vol. 1, no. 1, 100007, 2021.

