AI-Based Fake News Detection for Multilingual Indian Content: An Explainable Transformer-Ensemble Framework

Bishal Mishra , Mrs. Divya Singh (Assistant Professor)
School of Computer Applications, Babu Banarasi Das University, Lucknow, India
Bishalmishra2002bm@gmail.com , divyasingh.jd@bbdu.ac.in

***Abstract*—Fake news spreads fast in India, in many languages and scripts, and often in mixed text such as Hinglish. Most detection tools are built for English and cannot explain their answers. This paper describes a framework designed to fix both problems. It cleans and converts the text, reads it with four language models (BERT, XLM-RoBERTa, IndicBERT, and MuRIL), combines their votes with a stacked ensemble, and sends unsure cases to a human reviewer. LIME, SHAP, and attention views show which words drove each decision. We explain how the system will be scored with accuracy, precision, recall, and F1, and we set clear targets for each. We also set out six research questions and seven experiments. The performance numbers in this paper are ranges taken from earlier studies and goals we have set for ourselves. They are not results from our own runs, which will be reported once the experiments are finished.**

***Index Terms*—Code-mixing, ensemble learning, explainable AI, fake news detection, Indic languages, transformers.**

I. Introduction

India has more than 800 million internet users, and a lot of them get their news from WhatsApp, Facebook, X, and YouTube rather than from a newspaper or a TV channel. False stories ride the same channels, and they tend to surge around elections, health scares, and communal tension. Studies of online diffusion show that false news travels farther and faster than true news [1], while human fact-checkers work far too slowly to catch up.

Building a detector for India is harder than building one for English. Posts come in Hindi, Bengali, Tamil, Telugu, Marathi, Kannada, and English, written in several scripts. Plenty of people type in Roman letters or switch languages mid-sentence, for example mixing Hindi and English, and the same word may be spelled three different ways [2]. Some of these languages have very little labelled data [3]. On top of that, most deep learning detectors are black boxes. They say "fake" but not why, so a moderator cannot tell whether the model noticed something real.

This paper puts forward one framework that deals with all of these problems at once. Its main parts are:

- a five-layer pipeline that takes raw posts and returns a label with an explanation;
- a team of four language models, combined by stacking, with a confidence check that sends doubtful posts to a person;
- clear evaluation rules, including F1 targets, that tie six research questions to seven experiments; and
- a design for real-time use, with explanations shown next to every answer.

Section II covers related work. Section III states the problem. Section IV describes the framework. Section V explains the data and how we will measure success. Sections VI to IX cover deployment, expected performance, limits and ethics, and our conclusions.

II. Related Work

A. From Simple Models to Deep Learning

The earliest detectors simply counted words and word pairs (TF-IDF and n-grams) and passed the counts to classifiers such as logistic regression, SVMs, and random forests [4]. They are fast, but they ignore context and handle every language on its own. Neural networks came next. CNNs [5] and LSTMs [6] learn their own features and did better on benchmarks like LIAR [7] and FakeNewsNet [8], yet they need a lot of labelled text and stumble on new topics, new scripts, and mixed-language posts.

B. Transformers and Indian-Language Models

The Transformer [9] lets every word look at every other word, and BERT [10] showed that a model pre-trained on huge amounts of text can be fine-tuned for a small task with little data. mBERT covers about 100 languages with one shared vocabulary [11], and XLM-RoBERTa transfers across languages even better [12]. For India, two models stand out: IndicBERT, trained on 12 Indian languages [13], and MuRIL, which also learns from Roman-script text, the way many people really type [14].

C. India, Code-Mixing, and Explanations

Work on Indian misinformation finds that false stories shift their wording from one language to another and that WhatsApp groups are a main route for them [15]. Earlier papers have tackled transformer detection for Indian languages [16], transfer to low-resource languages [3], and benchmark datasets [17], but each covers only one piece of the puzzle. On the explanation side, LIME nudges the input and watches how the answer moves [18], while SHAP divides credit between words using Shapley values [19]; [20] discusses using both for fake news. Attention weights are easy to display but do not always explain a prediction faithfully [21]. Multilingual BERT can also be thrown off by small text changes [22].

TABLE I
How Detection Methods Have Developed

Period	Methods	Strength	Weakness
2015–18	TF-IDF; SVM, RF	Simple, fast	No context; one language
2018–21	CNN, LSTM	Learn features	Weak on new topics and mixed text
2021 on	mBERT, XLM-R, IndicBERT, MuRIL	Context; transfer	Need data; opaque

As far as we can tell, no single study brings together Indian-language models, mixed-text handling, transfer to low-resource languages, built-in explanations, and real-time use. That gap is what this work goes after.

III. Problem and Research Questions

Take a post x written in language l, possibly mixed or in Roman letters. A cleaning step φ turns it into a standard token list. Each of M models f_m gives a probability $p_m(x) = f_m(\varphi(x))$ that the post is fake, and a combiner h merges them:

$$p(x) = h(p_1(x), \ldots, p_M(x)) \quad (1)$$

The post is labelled fake if $p(x) \geq \tau$ and genuine otherwise. If the model is not confident enough (confidence below τ_c), a person reviews it. Our goal is to score well on every language, not just the big ones, within one second per post:

$$\max (1/|L|) \Sigma_{l \in L} F1_l \quad s.t. \quad T(x) < 1\ s \quad (2)$$

The six research questions are: RQ1, do transformers beat classical models on Indian languages? RQ2, which mix of cleaning, embeddings, and models works best? RQ3, how much does code-mixing hurt? RQ4, can a model trained on English and Hindi work on Kannada or Telugu? RQ5, do explanations help people trust the system and spot bias? RQ6, what design allows real-time use?

IV. Proposed Framework

A. Five Layers

Fig. 1 shows the design. Layer 1 collects text. Layer 2 cleans it. Layer 3 turns it into numbers, using TF-IDF features and transformer embeddings. Layer 4 classifies it with the ensemble. Layer 5 returns the label, the confidence, and the explanation. Each layer talks to the next through a fixed format,so a model can be swapped or a language added without touching the rest.

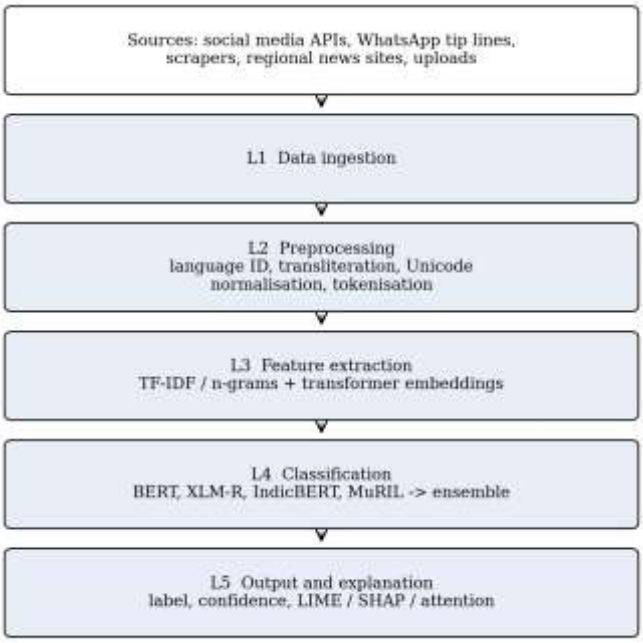


Fig. 1. Five-layer architecture of the proposed system.

B. Cleaning Mixed-Language Text

Table II lists the eight cleaning steps. The tool fastText [23] first detects the language. Roman-script Hindi is then converted to Devanagari, so different spellings of one word become one. For transformer models, removing stop words and reducing words to roots is optional, because it can throw away clues the models already use. We will test this directly in the ablation study.

TABLE II
The Eight Cleaning Steps

#	Step	Purpose
1	Language detection	Find the language; flag mixing
2	Transliteration	Roman Hindi to Devanagari
3	Unicode normalisation	One form per character
4	Tokenisation	spaCy, IndicNLP, NLTK
5	Stop-word removal	Language lists; keep “not”
6	Lemmatisation	Cut word endings (Tamil, Telugu)
7	Special characters	URLs, hashtags, emojis
8	Text normalisation	Fix spellings and numbers

Four more tricks help with mixed text. Indian-language models have vocabularies built for Indian scripts. Translated copies of training posts add data for small languages (never for testing). Language tags on each word help attention notice where the language switches. A character-level CNN copes with deliberate misspellings. When training, we sample each language in proportion to N_l^{α}, with $0 < \alpha < 1$, so small languages are not drowned out.

C. Models and Settings

We use BERT for English and XLM-RoBERTa, IndicBERT, and MuRIL for the other languages. Each reads a [CLS] token followed by the text and ends with a small classifier on the [CLS] output. Training uses weighted cross-entropy and the AdamW optimiser [24]. We code everything in PyTorch with the Hugging Face library [25]. Table III gives our starting settings, which we will tune on validation data only.

TABLE III
Starting Hyperparameters and Search Method

Model	Starting values	Search
Logistic reg., SVM	C = 1; RBF or linear	Grid
Random forest	200 trees; depth 20	Random
CNN / LSTM / BiLSTM	Hidden 128; dropout 0.3	Random
Transformers	LR 2×10^{-5}; batch 16–32; length 128–256; 3–5 epochs	Bayesian
Ensemble	Logistic-reg. meta-learner; 5 folds	Grid

D. Stacked Ensemble

Fig. 2 shows how the models vote. Stacking [26] trains a small meta-learner on the base models' out-of-fold answers, so it learns which model to trust for which language (Algorithm 1). We also compare plain weighted voting, $p(x) = \Sigma_m\ w_m\ p_m(x)$, and bagging. Posts below the confidence threshold are marked “unsure” and go to a person. Models that make

different mistakes help the ensemble most, so we choose members with different designs and training data.

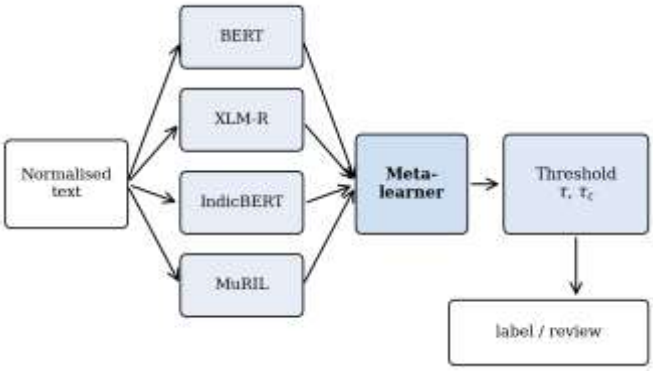


Fig. 2. Stacked ensemble with confidence thresholding.

Algorithm 1. Stacked ensemble training
Input: data D, base models $f_1..f_M$, K folds
1. Split D into K stratified folds
2. for each model f_m: for k = 1..K
train on D minus fold k; predict fold k
3. Build Z = $[p_1(x_i), ..., p_M(x_i)]$ for all i
4. Train meta-learner g on (Z, y)
5. Retrain each f_m on all of D; return f_m, g

E. Explanations

Every answer comes with three views: LIME word weights [18], SHAP values [19], and attention weights as extra evidence only [21]. The words that pushed the decision are highlighted, with the confidence score and a note when a person should check. Explanations take extra computing time, so they are made on request and then saved.

For each post the user sees:

- the label (fake, genuine, or unsure) and a confidence score;
- the detected language and script;
- the text with the most influential words highlighted in colour; and
- a short note on how reliable the system usually is for that language and text type.

V. Data and How We Will Measure Success

A. Data Sources

Table IV lists the data we plan to use. English corpora give plenty of training text. The Indian-language and WhatsApp, X, Reddit, and YouTube corpora give the realistic, informal, mixed-language text. We merge everything into one format and map detailed labels to fake or genuine with a written rule. New labels come from two native speakers, with a third settling disagreements, and we report their agreement with Cohen's kappa. Duplicate posts are removed, and posts about the same claim stay in the same split.

For multi-class tests we use six labels: genuine, made-up, misleading, rumour or unverified, satire, and hyper-partisan or inflammatory. Many corpora hold far more genuine than fake posts, or the reverse, so we balance the training data by oversampling the smaller class and by weighting the loss. We never balance the test set, so scores reflect real conditions. This is also why we report macro-F1 and not accuracy alone: a model that always says "genuine" on a 90%-genuine test set would score 90% accuracy but would catch no fake news at all.

TABLE IV
Planned Data Sources

Source	**Language**	**Use**
LIAR [7]	English	About 12,800 statements, six truth levels
FakeNewsNet [8]	English	News with social context
ISOT [4]	English	Fake and real articles
Indic corpora [17]	Hi, Bn, Ta, Te, Mr, Kn	Main multilingual training and testing
WhatsApp, X, Reddit, YouTube	Indian languages, Hinglish	Short, informal, mixed text

B. Metrics: Accuracy, Precision, Recall, and F1

We treat "fake" as the positive class. A fake post caught correctly is a true positive (TP). A genuine post wrongly flagged is a false positive (FP). A fake post missed is a false negative (FN). A genuine post correctly passed is a true negative (TN). Then:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (3)$$

$$Precision = TP / (TP + FP), \quad Recall = TP / (TP + FN) \quad (4)$$

$$F1 = 2 \times Precision \times Recall / (Precision + Recall) \quad (5)$$

Precision says how many flagged posts really were fake. Recall says how many fake posts we caught. F1 balances the two, so it does not reward a model that simply labels everything fake. We report macro-F1, which averages F1 over classes and languages equally, plus ROC-AUC, latency, and throughput. Table V works through the arithmetic with made-up numbers. These are not results.

TABLE V
Worked Example With Made-Up Numbers (Not Results)

200 test posts	**Predicted fake**	**Predicted genuine**
Actually fake (100)	TP = 80	FN = 20
Actually genuine (100)	FP = 10	TN = 90
Accuracy = 170/200	= 0.85	
Precision = 80/90	= 0.89	Recall = 80/100 = 0.80
F1 = 2(0.89)(0.80)/(0.89+0.80)	= 0.84	

We split data 70/15/15 by language and label, use 5-fold cross-validation, fix random seeds, and touch the test set only once. Differences between models are checked with McNemar's test or a paired bootstrap, and transformer results are averaged over several seeds.

C. Experiments

Table VI shows what each experiment tests.

TABLE VI
Experiments and Research Questions

Exp.	**RQ**	**What we do**
E1	1, 2	Compare classical, deep, and transformer models per language
E2	3	Test native, Roman, and mixed text; remove one cleaning step at a time
E3	4	Train on English and Hindi; test on other languages; add a share ρ of target data
E4	2	Stacking vs. voting vs. bagging vs. best single model
E5	5	Check if explanations are faithful and useful to reviewers
E6	3, 5	Attack with misspellings, paraphrases, character swaps
E7	6	Measure speed before and after quantisation and caching

VI. Deployment

The system runs as a Flask API with a Streamlit web page in front, sitting behind Nginx in Docker containers. Each result goes into MongoDB or MySQL together with its explanation, so we never have to compute it twice. A message queue such as Kafka absorbs bursts of traffic. To get under one second per post we shrink the models to 8-bit integers, which can cut their size by about 75% with little loss in accuracy [27], and run them in ONNX Runtime. Batching and caching help a bit more, and a fast first-pass model can settle the easy posts so the full ensemble only handles the harder ones. Content comes from permitted APIs and public pages [28]. WhatsApp is the exception: it cannot be watched in bulk, so we only check messages that users send in themselves. The work is planned over 16 weeks: data collection (weeks 1–4), the cleaning pipeline (5–8), model training (9–12), and deployment, explanations, and writing up (13–16).

Table VII lists the main project risks and how we plan to handle them.

TABLE VII
Main Risks and Planned Responses

Risk	Planned response
Little data in small languages	Transfer from English and Hindi; translated copies; verified fact-check sources
Noisy labels	Two annotators, kappa reported, sample audits
API limits or rule changes	Public sources and permitted APIs; modular collection
Slower than one second	Quantisation, batching, caching, fast first-pass model
Not enough computing power	Base-size models; short cloud-GPU runs
Privacy	Hide or hash identifiers; research use only
Too much trust in labels	Confidence check; human review; evidence shown

VII. Expected Performance and Targets

Table VIII shows two things. “Reference” is what earlier studies on similar tasks report, taken from our project synopsis. “Target” is the goal we set for our own system. Neither is a result of our experiments. The last column stays empty until we have run them.

TABLE VIII
Reference Ranges, Our Targets, and Measured Results

Model	Reference acc. (%)	Our target	Measured
Logistic regression	75–80	baseline	to be recorded
SVM	78–82	baseline	to be recorded
Random forest	80–85	baseline	to be recorded
CNN / LSTM / BiLSTM	85–93	baseline	to be recorded
BERT (English)	93–95	F1 ≥ 0.90	to be recorded
XLM-RoBERTa	92–94	F1 ≥ 0.88	to be recorded
IndicBERT, MuRIL	91–94	F1 ≥ 0.88	to be recorded
Ensemble	94–96	macro-F1 ≥ 0.90	to be recorded
Speed per post	–	< 1 second	to be recorded

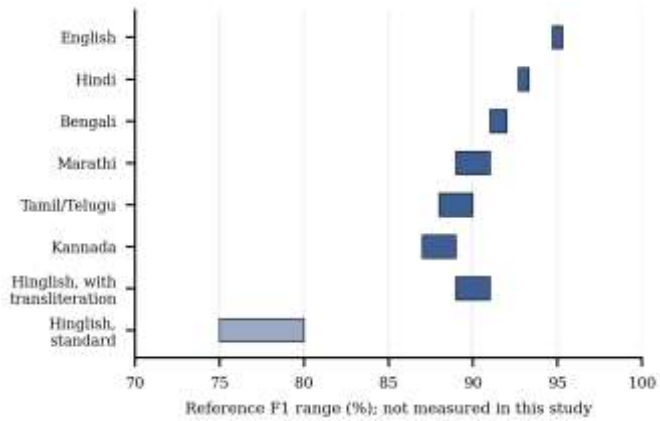


Fig. 3. Reference F1 ranges by language (from earlier studies; not our results).

Table IX breaks the expectations down by language and Table X does the same for transfer between languages. Again, these are reference values and goals, not our measurements.

TABLE IX
Language Coverage, Reference F1, and Our Targets

Language	Script	Data	Ref. F1 (%)	Target F1	Meas.
English	Latin	High	about 95	≥ 0.90	–
Hindi	Devanagari	High	about 93	≥ 0.90	–
Bengali	Bengali	Medium	91–92	≥ 0.85	–
Marathi	Devanagari	Low	89–91	≥ 0.85	–
Tamil, Telugu	Tamil, Telugu	Low	88–90	≥ 0.85	–
Kannada	Kannada	Low	87–89	≥ 0.85	–
Hinglish	Roman + mixed	Medium	89–91 with transliteration; 75–80 without	≥ 0.85	–

TABLE X
Reference Expectations for Transfer Between Languages

Setting	Reference acc. (%)	Meas.
English only → Hindi (no target data)	65–70	–
English only → Tamil (no target data)	60–65	–
English only → other small languages	55–60	–
Add about 20% target-language data	80–82	–
English or Hindi as stepping stone	5–10 points above direct	–
Hindi → Bengali (related languages)	about 85% of English score	–
Hindi → Tamil (different family)	about 75% of English score	–

The literature leads us to expect three things. First, scores should rise from classical to deep to transformer models, and the ensemble should add about two to three points over the best single model. Second, languages with more data should score higher: English near 95% F1, Kannada near 87–89%, and Hinglish about 89–91% with transliteration but only 75–80% without it. Third, a model trained only on English should reach roughly 55–70% accuracy on other languages, rising to about 80–82% when we add around 20% target-language data, with Hindi to Bengali working better than Hindi to Tamil. For every language we also aim for F1 of at least 0.85. If our measured numbers are far from these ranges, we will first check the data, the splits, and the code.

Table XI lists the ablations we plan and what we predict. These are guesses from design reasoning, and some may turn

out wrong. If cutting word endings helps Tamil or Telugu transformers, we will change the pipeline.

TABLE XI
PLANNED ABLATIONS AND PREDICTED EFFECT

Part removed	Predicted effect on macro-F1
Transliteration	Big drop on Hinglish
Language detection	Drop on small languages
Stop-word removal, lemmatisation	Little change for transformers
Character-level branch	Drop under misspelling attacks
Translation augmentation	Drop on small languages
Ensemble (use one model)	Drop of two to three points
Confidence threshold	More automatic errors; fewer reviews

To judge explanations (RQ5), we delete the words an explanation ranks highest and see how far the probability falls compared with deleting random words. We also check how often LIME and SHAP agree, and ask native-speaker reviewers whether the highlighted words are real evidence. For fairness, we report the gap between the best and worst language and try more data, re-weighting, or per-language thresholds if the gap is large.

Several things could weaken our conclusions, and we will keep them in mind. Labels from fact-checkers or automatic sources contain mistakes. Unequal data across languages can make one language look better only because it has more examples. The same claim appearing in both training and test data would inflate scores, which is why we remove duplicates and keep claims together. Models trained on old events may do worse on new topics, so we will test on data from a later period where possible. Public benchmarks are cleaner than live social media, so real-world scores may be lower than benchmark scores.

VIII. LIMITATIONS AND ETHICS

This study looks at text only, so images, audio, video, and deepfakes are out of scope. Data for the smaller languages is scarce, labels can be noisy, public benchmarks may be easier than live streams, and fake-news tactics keep shifting. An explanation can also look convincing without being faithful to the model. The ethical risks are real: people may lean too hard on an imperfect tool, wrongly flagged posts can silence legitimate or minority voices, and monitoring can slide toward surveillance [29]. For these reasons we see the system as an aid for people, not a judge. Doubtful cases go to a human reviewer, nothing is removed automatically, personal details are hidden or hashed, platform rules are followed, and results are shown per language so that gaps stay visible.

IX. CONCLUSION

We have described a framework for catching fake news in many Indian languages, including mixed-language text. It brings together careful cleaning, Indian-language and multilingual transformers, a stacked ensemble with human review of doubtful cases, and an explanation for every answer. We have also laid out how success will be measured, using accuracy, precision, recall, and F1 with clear targets. Whether the system hits those targets is the next question. We will report the real numbers, including any shortfall, when the experiments are done. Later work could add images and video, more languages, links to fact-check articles, and learning from reviewers' corrections.

REFERENCES

[1] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018.
[2] S. Khan and N. Ahmed, "Attention mechanisms for code-mixed text classification," *J. Natural Lang. Eng.*, vol. 30, no. 1, pp. 45–62, 2024.
[3] R. Gopal, P. Kumar, and A. Singh, "Transfer learning for low-resource Indian languages: Benchmarks and evaluation," *Comput. Linguistics Quart.*, vol. 28, no. 3, pp. 89–112, 2024.
[4] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using N-gram analysis and machine learning techniques," in *Proc. ISDDC*, 2017, pp. 127–138.
[5] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014, pp. 1746–1751.
[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[7] W. Y. Wang, "'Liar, liar pants on fire': A new benchmark dataset for fake news detection," in *Proc. ACL*, 2017, pp. 422–426.
[8] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "FakeNewsNet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big Data*, vol. 8, no. 3, pp. 171–188, 2020.
[9] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.
[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
[11] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?" in *Proc. ACL*, 2019, pp. 4996–5001.
[12] A. Conneau *et al.*, "Unsupervised cross-lingual representation learning at scale," in *Proc. ACL*, 2020, pp. 8440–8451.
[13] D. Kakwani *et al.*, "IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages," in *Findings of EMNLP*, 2020, pp. 4948–4961.
[14] S. Khanuja *et al.*, "MuRIL: Multilingual representations for Indian languages," *arXiv:2103.10730*, 2021.
[15] N. Desai, A. Nair, and J. Singh, "WhatsApp misinformation detection: Patterns and mitigation strategies in Indian elections," *Media & Society Rev.*, vol. 12, no. 4, pp. 201–225, 2023.
[16] D. Verma, M. Sharma, and A. Kumar, "Transformer-based fake news detection for multilingual Indian languages," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 1245–1258, 2024.
[17] R. Nair, M. Gupta, and V. Kumar, "Benchmark datasets for multilingual fake news detection in Indian languages," *Data in Brief*, vol. 51, Art. no. 109847, 2023.
[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. ACM SIGKDD*, 2016, pp. 1135–1144.
[19] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. NeurIPS*, 2017, pp. 4765–4774.
[20] V. Patel, R. Sharma, and K. Verma, "Explainable AI in fake news detection: LIME and SHAP-based interpretation," *J. Mach. Learn. Res.*, vol. 47, pp. 334–358, 2023.
[21] S. Jain and B. C. Wallace, "Attention is not explanation," in *Proc. NAACL-HLT*, 2019, pp. 3543–3556.
[22] J. Singh, V. Kumar, and A. Sharma, "Adversarial robustness of multilingual BERT models," *IEEE Trans. Inf. Forensics Security*, vol. 19, no. 6, pp. 4725–4738, 2024.
[23] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proc. EACL*, 2017, pp. 427–431.
[24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2019.
[25] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proc. EMNLP: System Demonstrations*, 2020, pp. 38–45.
[26] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.
[27] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE CVPR*, 2018, pp. 2704–2713.
[28] S. Chopra, P. Dubey, and K. Sharma, "Real-time misinformation detection architecture for social media platforms," *IEEE Softw.*, vol. 41, no. 2, pp. 52–61, 2024.
[29] C. Wardle and H. Derakhshan, "Information disorder: Toward an interdisciplinary framework for research and policymaking," Council of Europe, Strasbourg, France, 2017.