International Journal of Research in Engineering, Science and Management
Volume 9, Issue 5, May 2026
www.ijresm.com | E-ISSN: 2581-5792 | RESAIM Publishing | www.resaim.com

Multi-Signal Fake Social Media Profile Detection and Reporting: A Machine Learning Approach with Computer Vision and Clone Analysis

P. Ninaad[1*], Mihir S. Raju[2], Shreegouri J. Jahagridar[3], K. Karan Urs[1], Tanushree J. Mallalli[4], K. Jitesh[3]
[1]*Department of Computer Science (Data Science), RNS Institute of Technology, Bengaluru, India*
[2]*Department of Electronics and Communication Engineering, RNS Institute of Technology, Bengaluru, India*
[3]*Department of Computer Science and Engineering, RNS Institute of Technology, Bengaluru, India*
[4]*Department of Mechanical Engineering, RNS Institute of Technology, Bengaluru, India*

***Abstract*: The proliferation of automated, cloned, and AI-generated profiles on platforms such as Instagram and X (Twitter) poses a growing threat to online trust, public discourse, and cybersecurity. Manual reporting mechanisms and platform-side heuristics are insufficient against the scale and sophistication of modern fraudulent accounts. This paper presents a web-based, multi-signal fake social media profile detection and reporting system that integrates a trained Random Forest classifier, OpenCV-based face authenticity analysis, HuggingFace AI-image detection, fuzzy-string clone matching, and keyword-driven spam scoring into a unified, real-time risk engine. Profile data is fetched live via the Instagram Scraper API and Twitter API47 (RapidAPI). Seventeen extracted features — spanning metadata, behavioral, and content dimensions — feed the machine learning pipeline, which outputs a calibrated Fake Probability Score (0–100%) mapped to Low, Medium, and High risk tiers. A SQLite-backed history database records every analysis, and a background monitoring thread continuously re-evaluates watchlisted accounts. Evaluated on the UCI "user_fake_authentic_2class" benchmark dataset with hyperparameter tuning via GridSearchCV, the system achieves a weighted F1-score exceeding 0.92, demonstrating competitive performance against prior single-model baselines. The modular codebase, cross-platform support, and explainable risk breakdown distinguish this system from existing tools that offer only opaque classification outputs.**

***Keywords*: Fake profile detection, Random Forest, OpenCV, AI-image detection, clone detection, fuzzy matching, social media security, explainable AI, Instagram, Twitter/X, risk scoring.**

1. Introduction

Social media platforms have become the de facto public square of the twenty-first century, hosting over 5.4 billion active users worldwide as of early 2026. This unprecedented reach has simultaneously attracted a parallel ecosystem of inauthentic actors: bots scripted to amplify narratives, clone accounts impersonating public figures, AI-generated personas indistinguishable from genuine users, and spam networks monetizing fraudulent engagement. Studies consistently estimate that between 9% and 20% of active accounts on major platforms are fake or substantially automated [1], [2].

The consequences are far-reaching. In the political domain, coordinated inauthentic behavior has been documented in election interference campaigns across multiple countries [3]. In commerce, fake followers inflate influencer metrics and mislead brand marketing budgets. In cybersecurity, impersonation accounts serve as phishing vectors, harvesting credentials from unsuspecting victims. Despite platform-side countermeasures, adversarial actors continuously adapt, employing AI-generated deepfake profile pictures, high-entropy username generation, and behavioral mimicry to evade detection [4].

Traditional defenses rely on manual user reporting, which is reactive, slow, and subject to human error. Rule-based heuristic filters, while faster, are brittle against the evolving tactics of sophisticated bot networks. The research community has responded with increasingly powerful machine learning and deep learning solutions, yet most published systems suffer from three limitations: (i) single-platform scope, (ii) opaque classification with no user-facing explanation, and (iii) lack of real-time data integration, instead relying on static, pre-collected datasets [5], [6].

This paper addresses these gaps with a deployed Flask-based web application that performs real-time, multi-signal analysis of Instagram and X/Twitter profiles. Five independent detection modules, an ML classifier, a face detector, an AI-image detector, a clone checker, and a spam scorer execute in parallel and contribute to a weighted Fake Probability Score. An explainable breakdown shows users exactly which signals triggered the risk rating, supporting informed human decision-making. A background monitoring thread enables continuous surveillance of watchlisted accounts, generating alerts when risk scores shift significantly.

The remainder of this paper is organized as follows. Section 2 surveys related work. Section 3 describes the system architecture. Section 4 details the feature engineering and multi-module detection pipeline. Section 5 presents the machine learning methodology and experimental results. Section 6 discusses the findings and limitations. Section 7 concludes with

*Corresponding author: ninaadp25csds@rnsit.ac.in

Table 1
Comparison of related fake account detection systems

Study	Method	Platform	Real-Time	Accuracy
Khaled & Mokhtar [8]	SVM-NN Hybrid	Twitter	No	98.3%
Wang & Zhou [12]	CNN Image	Multi	No	97.0%
Wadkar et al. [16]	XGBoost + NLP + OpenCV	Instagram	Partial	95.2%
Karthikeyan et al. [17]	CNN/RNN Ensemble	Multi	No	94.7%
This Work	RF + CV + Clone + Spam + AI-Img	Instagram + X	Yes	≥0.92 F1

directions for future work.

2. Related Work

A. Feature-Based Detection

Early machine learning approaches to fake account detection focused on user-level profile features. Cresci et al. [7] demonstrated that simple metadata features follower count, statuses count, and default profile settings suffice to achieve over 95% classification accuracy on the MIB Twitter dataset using a Support Vector Machine (SVM). Khaled and Mokhtar [8] extended this by incorporating behavioral features derived from RenRen clickstream data, including average clicks per session, friend-request acceptance rates, and session length. Their hybrid SVM-NN algorithm, which feeds the decision values of a trained SVM into a neural network classifier, achieved 98.3% accuracy, outperforming either model alone. This work directly informs our ensemble design philosophy, though we prioritize tree-based methods for their interpretability at inference time.

B. Content and Linguistic Analysis

Yang et al. [9] identified that Twitter spammers adapt their behavioral patterns to evade detection, necessitating temporally robust features. Their feature set including bidirectional link ratio, API ratio, and following rate proved more evasion-resistant than static profile attributes. Varol et al. [10] applied the Botometer framework to analyze over 1,000 features spanning tweet content, temporal activity, and network connections, producing a comprehensive "bot-likeness" score. Ferrara et al. [11] surveyed the social bot landscape comprehensively, identifying that sentiment analysis and linguistic pattern matching are essential complements to structural feature extraction.

C. Image and Visual Analysis

Wang and Zhou [12] demonstrated that Convolutional Neural Networks (CNNs) can identify AI-generated profile images with 97% accuracy by exploiting pixel-level artifacts characteristic of GAN outputs. Gupta et al. [13] showed that adding CNN-based image verification to a metadata classifier improved overall detection accuracy by 18 percentage points. Our system integrates both classical computer vision (OpenCV Haar cascades for human face detection) and a pre-trained transformer-based AI-image classifier from HuggingFace ("umm-maybe/AI-image-detector"), combining noise variance analysis, edge density scoring, and model inference for a two-method consensus verdict.

D. Clone and Impersonation Detection

Boshmaf et al. [14] studied Sybil accounts that impersonate real profiles to gain trust before exploiting connections. Detection of such clone accounts requires fuzzy string matching rather than exact lookup, as clone creators introduce deliberate character substitutions to evade naive filters. Our clone detection module employs RapidFuzz [15] to compute token-ratio similarity scores between candidate and reference usernames and bios, complemented by a curated regex bank of clone-indicative patterns (e.g., "_official", "real_" suffixes, numeric trailing characters).

E. Positioning of this Work

Table 1 summarizes selected prior systems with respect to platform coverage, real-time data access, and explainability. Unlike most prior work, the proposed system combines live API integration, five parallel detection modules, an explainable risk breakdown, and continuous background monitoring within a single, deployable web application.

3. System Architecture

The system follows a seven-stage pipeline: (1) user input, (2) real-time data collection via platform APIs, (3) feature extraction, (4) parallel multi-module analysis, (5) weighted risk score aggregation, (6) web-based result presentation, and (7) persistent history storage with background re-monitoring.

A. Technology Stack

The backend is implemented in Python 3.11 and served through Flask. The machine learning pipeline uses scikit-learn for the Random Forest classifier and pandas/NumPy for data manipulation. Computer vision tasks are handled by OpenCV (cv2) and Pillow (PIL). The AI-image detection sub-module loads the "umm-maybe/AI-image-detector" model via the HuggingFace Transformers library and PyTorch. Clone similarity scores are computed with RapidFuzz. Profile data is retrieved through the Instagram Scraper API (instagram-scraper-api2 via RapidAPI) and Twitter API47, with HTTP requests managed by the "requests" library. All analysis records are persisted in an SQLite database managed directly through Python's sqlite3 module. The frontend is vanilla HTML/CSS/JavaScript, styled with the Syne and DM Sans typefaces.

B. Data Collection Module

When a user submits a username and selects a platform, the system dispatches an authenticated HTTP GET request to the appropriate RapidAPI endpoint. For Instagram, this retrieves the profile's follower count, following count, post count, biography, profile picture URL, verification status, and private/public flag. For X/Twitter, an equivalent set of fields is fetched. If the API returns an error or rate-limit response, the system gracefully degrades and logs the failure without crashing.

C. Parallel Analysis Architecture

Four analysis modules run concurrently using Python's threading module. The ML classifier, face detector, AI-image detector, and clone/spam checker each operate independently on the fetched profile data, with results collected and aggregated by the main application thread before the response is returned to the client. This design reduces total latency from the sum of individual module times to approximately the maximum of any single module, typically under five seconds for a complete analysis.

D. Persistent Storage and Monitoring

Every completed analysis is written to the "history" table in an SQLite database ("history.db"), recording the username, platform, prediction label, risk score, risk level, confidence, follower/following/post counts, bio length, verification status, spam score, face detection result, and clone detection result, along with a UTC timestamp. A second background thread iterates over accounts stored in the "monitored_accounts" table at a configurable interval, re-running the full analysis pipeline and writing to the "alerts" table whenever a risk score shift exceeds a defined threshold. This enables asynchronous, 24/7 surveillance of high-risk profiles without user interaction.

4. Feature Engineering and Detection Modules

A. Metadata Features

Seventeen features are extracted from each profile and grouped into three categories. Metadata features capture static profile attributes: follower count (flw), following count (flg), post count (pos), bio length in characters (bl), profile picture presence (pic, binary), external link presence (lin, binary), verification status (cs, binary), and private account flag (pr, binary). These features encode fundamental signals: unverified accounts with no profile picture, very short bios, and extreme follower-to-following ratios are disproportionately represented among fake profiles in benchmark datasets [7], [11].

B. Behavioral and Ratio Features

Behavioral features include the follower-to-following ratio ($fo = flw / \max(flg, 1)$), the numerical character ratio in the username ($ni = |\{c \in username : c \in [0–9]\}| / |username|$), and proxy engagement metrics (erl, erc) estimated from post and follower counts. Fake accounts generated by bot farms typically exhibit characteristic ratio signatures: very high following counts relative to followers ($fo << 1$) or, conversely, inflated follower counts purchased through follower marketplaces ($fo >> 100$) [9]. High numerical ratios in usernames reflect automated name generation strategies.

C. ML Classification Module

The trained Random Forest classifier (serialized as "fake_detector_model.pkl") accepts the seventeen-dimensional feature vector and outputs a probability estimate for the "fake" class. GridSearchCV was used to tune hyperparameters over the grid {n_estimators: [100, 200, 300], max_depth: [10, 20, None], min_samples_split: [2, 5, 10], class_weight: ["balanced"]}, with 5-fold cross-validation and weighted F1 as the scoring metric. The best estimator's probability output is multiplied by 100 to yield an ML risk sub-score.

D. Face Detection Module

The profile picture URL is downloaded and decoded into a NumPy array via Pillow. The image is resized to 300×300 pixels and converted to grayscale, followed by histogram equalization to improve contrast under varying lighting conditions. Four OpenCV Haar cascade classifiers, haarcascade_frontalface_default.xml, haarcascade_frontalface_alt.xml, haarcascade_frontalface_alt2.xml, haarcascade_profileface.xml — are applied sequentially with progressively relaxed detection parameters (scaleFactor = 1.05, minNeighbors = 3, minSize = 20×20 px). A positive detection by any cascade returns has_face = True. Absence of a human face is a significant indicator of a fake or bot account, as real users overwhelmingly use personal photographs as profile pictures [12].

E. AI-Image Detection Module

The face detection module identifies the presence of a human face but cannot distinguish between a genuine photograph and a photorealistic AI-generated image. To address this gap, the system applies a two-method consensus. Method 1 (pixel analysis) computes three statistics from the image array: Laplacian variance ($noise\_var = \sigma^2(\nabla^2 I)$), Gaussian blur residual ($smoothness = \text{mean}(|I - \text{blur}(I)|)$), and Canny edge density ($edge\_density = |\{p : edges(I)[p] > 0\}| / |I|$). Thresholds of $noise\_var < 100$, $smoothness < 8$, and $edge\_density < 0.05$ each contribute one "AI signal." Method 2 passes the image through the HuggingFace AI-image-detector transformer and extracts the probability assigned to labels containing the tokens "artificial," "fake," or "ai." A verdict of "Very likely AI-generated" is issued when model probability exceeds 0.75 and pixel signals ≥ 2; a verdict of "Possibly AI-generated" when either condition alone is satisfied.

F. Clone Detection Module

Clone detection operates at three levels. First, the username is matched against eleven regular expression patterns (e.g., .*_+official.*, real_+.*, .*_+[0-9]+$) that encode common clone naming strategies, contributing 30 points to a clone score per matching pattern. Second, if a reference username is provided, RapidFuzz token-ratio similarity is computed; similarity > 85% adds 40 points, similarity > 70% adds 20 points. Third, if a reference biography is provided, bio similarity is computed analogously. A clone score ≥ 60 yields a verdict of "Likely Clone," while 30–59 yields "Possibly Clone." Optionally, a reverse image search via the SerpAPI Google Reverse Image endpoint checks whether the profile picture appears on multiple third-party sites, adding 30 points if found on more than two locations.

G. Spam Detection Module

A curated keyword bank covering domains including pharmaceutical spam, financial scams, follower-sale services, adult content solicitation, and political disinformation is applied

Table 2
Classification performance on held-out test set

Metric	Baseline RF	Tuned RF (This Work)	SVM-NN [8]
Accuracy	0.885	0.924	0.983
Precision (fake)	0.891	0.931	~0.97
Recall (fake)	0.876	0.918	~0.97
F1-Score (weighted)	0.883	0.922	0.983
False Positive Rate	0.114	0.076	0.013

to the profile biography using substring matching. Each keyword match increments a spam score. The final spam score is normalized and incorporated into the overall risk computation.

H. Risk Score Aggregation

The five module outputs are combined into a weighted Fake Probability Score $S \in [0, 100]$:

$$S = w_1 \cdot ML + w_2 \cdot (1 - face) + w_3 \cdot AI\_img + w_4 \cdot clone + w_5 \cdot spam$$

where weights $\{w_1, w_2, w_3, w_4, w_5\}$ are empirically calibrated to balance the relative discriminative power of each module. Verified accounts bypass the ML and clone modules, receiving a baseline "Low Risk" verdict regardless of other signals, consistent with platform-level identity assurance. $S \geq 60$ maps to "High Risk" (fake), $40 \leq S < 60$ maps to "Medium Risk" (suspicious), and $S < 40$ maps to "Low Risk" (likely genuine).

5. Machine Learning Methodology and Experimental Results

A. Dataset

The ML classifier was trained and evaluated on the publicly available "user_fake_authentic_2class" dataset, which contains 696 Instagram profiles labeled as fake (f) or real (r) across 17 numeric features: pos (posts), flw (followers), flg (following), bl (bio length), pic (has profile picture), lin (has external link), cl (has comment links), cz (is celebrity), ni (numerical ratio in username), erl, erc, lt (has location tag), hc (has hashtag in caption), pr (is private), fo (follower–following ratio), cs (is verified), and pi (profile image score). The class distribution is approximately 50/50, validated by value_counts() prior to splitting.

To further enrich the training set, the "improved_model.py" pipeline additionally parses a collection of JSON-format Instagram profile exports, extracting the same 17 features for each account. All JSON accounts are labeled fake, reflecting their provenance from known fake-account repositories. The combined dataset provides broader coverage of bot-farm naming patterns and follower distribution anomalies.

B. Training Procedure

Data is split 80/20 into training and held-out test sets using a fixed random seed (random_state=42) for reproducibility. The Random Forest classifier is initialized with class_weight="balanced" to compensate for any residual class imbalance. Hyperparameter search is conducted over the Cartesian product of the grid defined in Section IV-C using GridSearchCV with 5-fold stratified cross-validation. The scoring metric is weighted F1-score, which accounts for both precision and recall across both classes. The best-performing estimator is serialized with Python's pickle module as "fake_detector_model.pkl".

C. Results

Table 2 reports classification metrics for the baseline Random Forest (n_estimators=100, default hyperparameters) and the tuned model on the held-out test set.

The tuned Random Forest improves weighted F1 from 0.883 to 0.922, a 4.4 percentage point gain attributable primarily to optimal tree depth regularization (max_depth=20) and balanced class weighting. The False Positive Rate drops from 11.4% to 7.6%, reducing the risk of incorrectly flagging legitimate users — a critical operational metric given the reputational consequences of false accusations. The SVM-NN hybrid of Khaled and Mokhtar [8] reports higher absolute accuracy (98.3%), but was trained and evaluated on the MIB Twitter dataset under controlled laboratory conditions without live API integration; direct numerical comparison across datasets is therefore not meaningful.

D. Feature Importance

Random Forest provides native feature importance scores via the mean decrease in impurity (MDI) across all trees. The top five features by MDI are: (1) fo (follower–following ratio), (2) flw (follower count), (3) ni (numerical character ratio in username), (4) bl (bio length), and (5) pic (profile picture presence). These align with findings in the literature: ratio anomalies and structural profile incompleteness are the strongest discriminators between fake and real accounts [7][11]. Notably, pr (private account) and cs (verified) rank lower than expected, reflecting that bot operators have increasingly adopted private-account settings to evade detection — an adversarial adaptation documented in recent surveys [4].

6. Discussion

A. Explainability and User Trust

A key design principle of the system is that every risk verdict is accompanied by a per-signal breakdown, listing which of the five detection modules flagged the account and the specific behavioral triggers (e.g., "No human face detected," "Username matches clone pattern: .*_official.*," "Bio contains spam keyword."). This aligns with the emerging field of Explainable AI (XAI), which argues that opaque classifier outputs are insufficient for high-stakes decisions such as account suspension. By surfacing the reasoning behind each verdict, the system empowers users and platform moderators to apply judgment, challenge incorrect flags, and learn the

characteristics of inauthentic behavior.

B. Adversarial Robustness Limitations

The system's primary limitation is susceptibility to adversarial adaptation. A sophisticated actor aware of the detection features could craft a profile with a plausible username, a populated biography, a purchased follower base yielding a realistic fo ratio, and a photorealistic AI-generated profile picture that passes both Haar cascades (by including a genuine human face) and the AI-image detector (by using a high-quality generation model trained to minimize pixel artifacts). Defending against such adversaries requires continuous retraining on fresh labeled data and integration of temporal behavioral signals (posting frequency, engagement rate over time) that are not accessible through static profile scraping.

C. Platform API Constraints

A practical limitation encountered during development was rate limiting and data access restrictions imposed by platform APIs. The Instagram Graph API (official Meta endpoint) restricts access to the authenticating user's own account, making it unsuitable for third-party profile analysis. The system therefore relies on third-party scraper APIs (RapidAPI), which are not officially sanctioned by Meta and may be subject to service interruptions. Future work should explore browser-extension-based data collection, which operates within the authenticated user's session and avoids third-party scraper dependencies.

D. False Positive Analysis

The most common sources of false positives observed during manual validation were (i) newly created legitimate accounts with low post counts and no profile picture, (ii) professional brand accounts with very high following counts and keyword-rich bios that triggered the spam scorer, and (iii) celebrity accounts with extreme for ratios (many followers, few following) that resemble purchased-follower profiles. Mitigation strategies include adding a verification bypass (already implemented), a "account age" feature (not available through current API endpoints), and a user-override mechanism allowing manual reclassification.

7. Conclusion

This paper has presented a real-time, multi-signal fake social media profile detection system that integrates machine learning, computer vision, AI-image analysis, clone detection, and spam scoring into a unified, explainable risk engine deployed as a Flask web application. The system achieves a weighted F1-score of 0.922 on a standard benchmark dataset after hyperparameter tuning, with a 7.6% false positive rate on the held-out test set. Five detection modules execute in parallel against live API-fetched profile data, returning a complete analysis typically within five seconds. A SQLite-backed history and monitoring subsystem enables 24/7 surveillance of high-risk accounts.

The primary contributions of this work are: (i) a deployable, full-stack implementation combining five detection modalities not previously integrated in a single system; (ii) an explainable risk breakdown that surfaces the specific behavioral triggers behind each verdict; (iii) a background monitoring thread that autonomously tracks risk score evolution over time; and (iv) an AI-image detection pipeline combining pixel-level artifact analysis with a pre-trained HuggingFace transformer to detect photorealistic AI-generated profile pictures.

Future work will focus on three directions. First, incorporating temporal behavioral features (posting cadence, engagement rate decay, follow/unfollow velocity) requires extended API access or browser-extension-based collection and is expected to significantly improve recall for sophisticated human-mimicking bots. Second, the system will be extended to Facebook and TikTok through modular data fetcher additions. Third, adversarial training — augmenting the training set with synthetically generated adversarial profiles that deliberately target known feature thresholds — will improve robustness against evasion.

Acknowledgment

The authors thank Ms. Rachita E (Assistant Professor, Dept. of CSE-DS, RNSIT) for her guidance throughout this project. The authors also acknowledge the reviewers Dr. Prashanna Kumar M (CSE), Dr. MS Ramya (CV), and Dr. Ohileshwari M.S. (ECE) for their constructive feedback during the Phase-II presentation. Computations were performed on personal development machines using open-source software libraries. No external funding was received for this work.

References

[1] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[2] S. Alothali, N. Zaki, E. Mohamed, and H. Alashwal, "Detecting social bots on Twitter: A literature review," *Computer Science Review*, vol. 29, pp. 1–14, 2018.

[3] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer, "Truthy: Mapping the spread of astroturf in microblog streams," in *Proc. 20th Int. Conf. Companion World Wide Web (WWW)*, 2011, pp. 249–252.

[4] L. Wu, F. Morstatter, K. M. Carley, and H. Liu, "Misinformation in social media: Definition, manipulation, and detection," *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 2, pp. 80–90, 2020.

[5] S. Ahmed and M. Abulaish, "A generic statistical approach for spam detection in online social networks," *Computer Communications*, vol. 36, no. 10–11, pp. 1120–1129, 2013.

[6] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. 26th Annual Computer Security Applications Conf. (ACSAC)*, 2010, pp. 1–9.

[7] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake Twitter followers," *Decision Support Systems*, vol. 80, pp. 56–71, 2015.

[8] S. Khaled and H. M. O. Mokhtar, "Detecting fake accounts on social media," in *Proc. IEEE Int. Conf. Big Data*, 2022.

[9] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving Twitter spammers," *IEEE Trans. Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, 2013.

[10] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proc. 11th Int. Conf. Web and Social Media (ICWSM)*, 2017, pp. 280–289.

[11] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, Jul. 2016.
[12] Z. Wang and Y. Zhou, "Deep learning for fake account detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2023.
[13] A. Gupta, R. Joshi, and K. Patel, "Deep learning for fake social media accounts," in *Proc. Int. Conf. Machine Learning and Applications*, 2022.
[14] Y. Boshmaf, M. Ripeanu, K. Beznosov, and E. Santos-Neto, "Thwarting fake OSN accounts by predicting their victims," in *Proc. 8th ACM Workshop on Artificial Intelligence and Security*, 2015, pp. 81–89.
[15] M. Bachmann, "RapidFuzz: A fast string matching library," 2021. Available: https://github.com/maxbachmann/RapidFuzz
[16] S. Wadkar, R. Patil, O. Choudhari, O. Patil, S. Bhosale, T. Kulkarni, and S. Gaikwad, "Fake social media accounts and their detection," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 7, no. 2, Mar.–Apr. 2025.
[17] M. Karthikeyan, B. Venkatesh, G. Muneesh, Y. Abhilash, and E. S. Vignesh, "Fake social media accounts and their detection," *International Journal of Scientific Research in Engineering and Management*, vol. 9, no. 4, Apr. 2025.
[18] P. K. Shukla *et al*., "Fraudulent account detection in social media using hybrid deep learning," *Scientific Reports*, 2025.
[19] F. Pedregosa *et al*., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.