



FAKE PROFILE DETECTION ON SOCIAL MEDIA USING MACHINE LEARNING

Mrs.Dalavi M.T.¹, Diksha Hanumant Gade²

¹ Faculty Guide & Supervisor | ² dattakala group of institution, India

Corresponding author: gadediksha29@gmail.com

ARTICLE ID IJREK-2026-00130	ARTICLE TYPE Research Article	RESEARCH AREA Computer Science	PUBLICATION Volume 1, Issue Issue 1 (2026)
RECEIVED 21 October 2026	REVISED 25 October 2026	ACCEPTED 22 October 2026	PUBLISHED 27 October 2026

Abstract

Social media supports communication, learning, entertainment, and business, but fake or misleading accounts may spread spam, impersonate people, manipulate engagement, or direct users to fraudulent content. Manual identification is difficult at scale. This paper proposes a supervised machine-learning workflow to classify profiles as potentially genuine or potentially fake using available profile features. The planned pipeline includes a labelled dataset, data cleaning, feature preparation, train-test splitting, and a Random Forest classifier. Candidate features include profile-picture availability, biography and username characteristics, follower/following counts, post count, and account age when lawfully available. Performance will be assessed with precision, recall, F1-score, accuracy, and a confusion matrix. False accusations and missed fake accounts are both important risks. This is a research plan, not a completed experiment; no project-specific performance score is claimed.

Keywords: Fake profile detection; machine learning; social media security; Random Forest; classification; cybersecurity

1.2 PROBLEM STATEMENT

Manual review takes time and does not scale well. Simple rules—for example, labelling every low-follower account as fake—are unreliable because genuine users may also have small audiences or incomplete profiles. The proposed study asks whether a supervised model can use available profile attributes to distinguish labelled fake profiles from labelled genuine profiles in a selected dataset. 1.3 Motivation This student-level cybersecurity project provides practice in preprocessing, classification, evaluation, and responsible machine-learning use. Predictions should help a human reviewer prioritize further checks, not prove dishonesty or trigger punishment automatically.

1.4 RESEARCH GAP

Previous work explores profile-based classification, network analysis, graph methods, and deep learning. Results are difficult to compare because platforms, datasets, labels, features, and evaluation procedures differ. Dataset freshness and changing account behaviour also affect generalization. This project therefore focuses on a transparent, reproducible baseline with multiple metrics, documented limitations, and careful error analysis—not an unverified claim of superior accuracy. 1.5 Objectives Review machine-learning approaches for fake-profile and social-bot detection. Identify profile features available in a labelled dataset. Design a supervised pipeline using Random Forest as the baseline. Define a transparent train-test procedure and standard evaluation metrics. Examine false positives, false negatives, bias, privacy, and generalization. Provide a foundation for later implementation and comparison.

1.6 SCOPE AND CONTRIBUTIONS

The scope is tabular classification on a lawful, labelled dataset. It excludes private-data collection and unauthorized account access. The contribution is a documented system design, feature plan, evaluation procedure, and reporting requirements. Image analysis, natural-language processing, network graphs, and live platform integration are future extensions.

2. LITERATURE REVIEW / RELATED WORK

Fake-profile research overlaps with social bots, spam accounts, Sybil identities, and malicious accounts, but these terms are not interchangeable. A Sybil attack creates multiple identities; a social bot is automated or semi-automated; a fake profile may be human-operated. The dataset must clearly define its positive class.

2.1 SURVEYS OF DETECTION TECHNIQUES

Ramalingam and Chinnaiah reviewed fake-profile detection in large online social networks [1]. Latah surveyed malicious social-bot detection methods [2]. Orabi et al. systematically reviewed bot detection and noted changing bot behaviour and dataset limitations [3]. Aljabri et al. reviewed supervised, semi-supervised, and unsupervised machine-learning approaches [4]. Hayawi et al. reviewed deep-learning methods and preprocessing choices [5]. These studies show the importance of documenting platform, labels, and evaluation conditions.

2.2 FEATURE-BASED AND TRADITIONAL MACHINE LEARNING

Traditional methods use profile metadata, activity statistics, text features, and social connections. Chatterjee describes fake-profile detection using visible profile features and machine-learning methods [7]. A conference study compared SVM, Random Forest, and neural networks [8], while another examined Random Forest, Logistic Regression, and Decision Tree [9]. These studies make Random Forest a reasonable baseline to test, but their results must not be reported as results of this project.

2.3 BEHAVIOURAL AND NETWORK-BASED METHODS

Posting frequency, repeated content, interaction patterns, and relationships may reveal signals that a static profile misses. Research has considered changing activity over time [10], identity-deception detection through multiple attributes [11], and detecting fake accounts at registration [12]. Feature availability and detection timing influence system design.

2.4 RECENT RESEARCH AND CONTINUING CHALLENGES

Recent reviews identify labelling quality, class imbalance, changing attacker behaviour, and generalization as continuing challenges [4]–[6], [13]. A 2025 survey

reviews deception in online social ecosystems [14], and a 2025 study benchmarks machine-learning and deep-learning architectures on a public Instagram dataset [15]. Complex models are not automatically best for a small project; performance, computational cost, and error types must be considered.

2.5 SUMMARY AND RESEARCH GAP

The project's methodological contribution is a reproducible baseline that reports dataset source, feature definitions, splitting method, parameters, metrics, and errors. An unusual profile should not automatically be treated as fake.

3.1 SYSTEM OVERVIEW

The system takes a labelled dataset, checks data quality, separates features from the target label, creates training and testing subsets, trains a Random Forest classifier, and evaluates predictions on held-out records. The output is "potentially fake" or "potentially genuine." It is decision support only; a prediction should not by itself lead to accusing, blocking, or reporting a user. Proposed Workflow: Labelled Dataset ↓ Validation and Cleaning ↓ Feature Preparation ↓ Train-Test Split ↓ Random Forest Training ↓ Held-Out Prediction ↓ Metric Calculation ↓ Error Review. .

3.2 DATASET REQUIREMENTS

Use a documented, labelled dataset. Record the platform, collection date if known, record and feature counts, label definitions, licence, and usage restrictions. A possible source mentioned in the draft is the Instagram Fake Spammer Genuine Accounts dataset [16]. Verify access, fields, licence, and academic-use permission before selecting it. The exact dataset has not yet been chosen.

3.3 CANDIDATE FEATURES

Possible fields are profile-picture availability, biography length, username length, follower count, following count, post count, account age, and activity summaries. These are candidate inputs, not rules that prove an account is fake. A low follower count may describe a new or private genuine account. Handle missing values consistently and do not invent absent features.

3.4 DATA PREPROCESSING

Inspect dataset size, column names, data types, and class labels. Remove exact duplicates only when they are confirmed accidental; document the rule. Check missing and invalid values and justify the handling method. Encode categorical features when required. Separate the target label and remove identifiers that may leak labels. Split into training and testing sets, using stratification where appropriate. Fit transformations on training data only, then apply them to test data to prevent leakage.

3. PROPOSED METHODOLOGY / SYSTEM (CONTINUED)

3.5 Proposed

Algorithm: Random Forest Random Forest is an ensemble method that trains multiple decision trees and combines their predictions. Trees use samples of training records and subsets of features when selecting splits. The combined output determines the class. It is selected because it can model nonlinear relationships in tabular data, is available in Scikit-learn, and provides feature-importance tools. Importance scores can be affected by correlated features and do not prove causation.

TABLE I

Metric	Measured value	Meaning
Accuracy	To be measured	Correct predictions / all predictions
Precision (fake class)	To be measured	Predicted fake cases that were labelled fake
Recall (fake class)	To be measured	Labelled fake cases detected
F1-score (fake class)	To be measured	Balance of precision and recall
Test-set size	To be Added	Number of held-out examples

3.6 ALGORITHM STEPS

Input a labelled profile dataset D. Validate the dataset and identify the target column. Prepare feature matrix X and label vector y. Create training and test subsets using a documented split and random seed. Train Random Forest on the training subset. Predict labels for the held-out test subset. Calculate accuracy, precision, recall, F1-score, and confusion matrix. Inspect false positives and false negatives; record limitations. Save the model and experiment configuration only if needed and permitted.

3.7 EVALUATION MEASURES

Accuracy is the proportion of all test examples classified correctly. Precision for the fake class is the proportion of profiles predicted fake that are labelled fake. Recall is the proportion of labelled fake profiles correctly identified. F1-score is the harmonic mean of precision and recall [17]. A confusion matrix reports true positives, true negatives, false positives, and false negatives. Accuracy alone can hide weak performance on a minority class, so per-class or macro-averaged metrics may be needed.

4.1 PROPOSED SOFTWARE

Planned tools: Python; Pandas and NumPy for data handling; Scikit-learn for splitting, training, and metrics; Matplotlib for optional plots; and Jupyter Notebook or Google Colab as the environment. Record exact library versions during implementation. These tools have not yet produced results for this draft. 4.2 Hardware A standard laptop or desktop should be sufficient for a small tabular dataset. A GPU is not normally required for this Random Forest baseline. Record computer configuration, operating system, Python version, and memory if required by the college.

4.3 EXPERIMENTAL DESIGN

A proposed starting point is an 80:20 training-to-test split, with stratification where suitable. Confirm the setting during implementation and record a fixed random seed. If duplicate profiles, records from the same person, or related accounts exist, a random split may put related examples in both sets and inflate performance. Use grouped or time-based splitting if the dataset requires it.

4.4 BASELINE AND MODEL PARAMETERS

Record the number of trees, maximum tree depth, minimum samples per leaf, and class weighting. Start with a documented baseline. Tune parameters only with training data—for example, through cross-validation—not by repeatedly selecting settings based on the final test set. Logistic Regression or Decision Tree can be compared using the same split and features.

4.5 REPRODUCIBILITY AND ETHICS

Record dataset source/version, preprocessing, selected columns, random seed, software versions, model parameters, and metric definitions. Use only data permitted for research. Avoid private messages, passwords, private account details, and unnecessary personal data. Remove usernames or IDs unless their use is clearly justified. Report all results honestly, including errors and poor performance.

4.6 INFORMATION TO COMPLETE AFTER IMPLEMENTATION

Dataset Information: The name of the dataset, its source URL, access date, licence, and class labels will be documented. Dataset Size: The total number of records and the number of genuine and fake profiles after data cleaning will be recorded. Feature Information: The final list of features used for training the model and any removed features will be documented. Software Information: The versions of Python, Pandas, NumPy, and Scikit-learn used in the experiment will be recorded. System Configuration: The hardware and operating system used for implementation will be documented, if required. Model Configuration: The training and testing split, random seed, Random Forest parameters, and cross-validation settings will be recorded

5. RESULTS AND DISCUSSION

No implementation or experiment has been completed for this draft. Therefore, no project accuracy, precision, recall, F1-score, or model-comparison result is claimed. Results from other papers must not be presented as results of this project.

5.2 CONFUSION MATRIX AND FIGURES

After testing, include a confusion matrix with clearly labelled axes and class names. If algorithms are compared, present measured metrics in a numbered table or bar chart. Number and caption each figure/table and explain its meaning. Never present guessed or illustrative values as experimental results.

5.3 HOW TO DISCUSS RESULTS

Low precision means many genuine profiles may be incorrectly labelled fake. Low recall means many labelled fake profiles may be missed. A strong F1-score may show a useful balance, but dataset size and label quality still matter. Examine the confusion matrix and report per-class or macro-averaged scores when classes are unequal. Results from one dataset do not guarantee equal performance on another platform or at a later date.

5.4 INTERPRETATION LIMITS

The model learns patterns in training data, including possible label errors and biases; it does not know a user's intent. Genuine accounts may look suspicious, and fake accounts may imitate normal profiles. Predictions should trigger additional review, not serve as definitive decisions about people. Claims of improvement require suitable experiments and comparable evaluation.

6. ADVANTAGES AND LIMITATIONS

6.1 Advantages

Understandable workflow suitable for a student project. Python and Scikit-learn provide accessible data-processing and classification tools. Random Forest can model nonlinear patterns in structured features. Multiple metrics provide more insight than accuracy alone. The modular pipeline can later use text, time, or network features. The project encourages reproducibility and honest reporting. 6.2 Limitations The project is not implemented, so effectiveness is not yet established. Performance depends on dataset accuracy, representativeness, and age. Labels may be incomplete or use a different fake-profile definition. Features may differ across platforms. Duplicates or related records can inflate random-split results. Static profile attributes can be manipulated and may disadvantage new or low-activity users. False positives can harm genuine users; predictions must not automatically punish or expose them. Platform APIs, data permissions, and privacy rules restrict available information.

7. FUTURE SCOPE

Future work can extend the baseline in controlled stages: Compare Logistic Regression, Decision Tree, Support Vector Machine, and Gradient Boosting with Random Forest using the same procedure. Select a more representative dataset with documented labels and lawful access. Use natural-language processing on public biography text only where ethical and necessary. Add temporal features to summarize activity over time. Use graph-based methods to study account relationships when suitable data is available. Build a simple interface showing the predicted class, features used, and a warning about uncertainty. Evaluate fairness, behaviour drift, resistance to manipulation, and the cost of false positives. Implement and test each extension before describing it as completed. 8. Conclusion This paper proposes a machine-learning approach to detecting potentially fake social-media profiles using labelled profile data, preprocessing, feature selection, Random Forest, and standard

evaluation metrics. Research has explored traditional machine learning, behavioural analysis, network methods, and deep learning [1]–[15]. Data quality, changing behaviour, label reliability, and platform differences remain challenges. The project aims to provide a transparent baseline using Python and Scikit-learn. Since implementation has not taken place, no project-specific result is claimed. The final submission must include the actual dataset, configuration, measured results, confusion matrix, and error discussion. Predictions should support human review, not prove an account is fake. 9. References [1] D. Ramalingam and V. Chinniah, "Fake profile detection techniques in large-scale online social networks: A comprehensive review," *Computers & Electrical Engineering*, 65, 165–177, 2018. <https://doi.org/10.1016/j.compeleceng.2017.05.020> [2] M. Latah, "Detection of malicious social bots: A survey and a refined taxonomy," *Expert Systems with Applications*, 151, 113383, 2020. <https://doi.org/10.1016/j.eswa.2020.113383> [3] M. Orabi et al., "Detection of Bots in Social Media: A Systematic Review," *Information Processing & Management*, 57(4), 102250, 2020. <https://doi.org/10.1016/j.ipm.2020.102250> [4] M. Aljabri et al., "Machine learning-based social media bot detection: a comprehensive literature review," *Social Network Analysis and Mining*, 13, 20, 2023. <https://doi.org/10.1007/s13278-022-01020-5> [5] K. Hayawi et al., "Social media bot detection with deep learning methods: a systematic review," *Neural Computing and Applications*, 35, 8903–8918, 2023. <https://doi.org/10.1007/s00521-023-08352-z> [6] I. Ben Sassi and S. Ben Yahia, "Malicious accounts detection from online social networks: a systematic review of literature," *International Journal of General Systems*, 2021. <https://doi.org/10.1080/03081079.2021.1976773> [7] P. Chatterjee, "Fake Social Media Profile Detection—Machine Learning Algorithms and Applications," in *Machine Learning Algorithms and Applications*, Wiley, 2021. <https://onlinelibrary.wiley.com/doi/10.1002/9781119769262.ch11> [8] "Deep Learning Approach for Identification of Fake Profiles in Social Media," *WCONF*, 2023. <https://doi.org/10.1109/WCONF58270.2023.10235178> [9] "Fake Account Detection on Social Media using Random Forest Classifier," *ICICCS*, 2023. <https://ieeexplore.ieee.org/document/10142841/> [10] "Combating the evolving spammers in online social networks," *Computers & Security*, 72, 60–73, 2018. <https://doi.org/10.1016/j.cose.2017.08.014> 9. References (continued) [11] "Cyber-security: Identity deception detection on social media platforms," *Computers & Security*, 78, 76–89, 2018. <https://doi.org/10.1016/j.cose.2018.05.015> [12] "Detecting Fake Accounts in Online Social Networks at the Time of Registrations," *ACM CCS*, 2019. <https://doi.org/10.1145/3319535.3363198> [13] H.-D. Nguyen et al., "Supervised learning models for social bot detection: Literature review and benchmark," *Expert Systems with Applications*, 2024. <https://doi.org/10.1016/j.eswa.2023.122217> [14] A. Safraz et al., "Unmasking deception: detection of fake profiles in online social ecosystems," *Journal of Big Data*, 12, 214, 2025. <https://doi.org/10.1186/s40537-025-01254-y> [15] P. Saurabh et al., "Deep vision against deception using CNN strategies for fake social media profile detection," *Discover Artificial Intelligence*, 2025. <https://link.springer.com/article/10.1007/s44163-025-00613-1> [16] "Instagram Fake Spammer Genuine Accounts Dataset," project resource. https://github.com/Ilmaan-Ahamed/NLP_Group_01. Verify access, licence, and label definitions before use. [17] Scikit-learn Developers, "f1_score—Compute the F1 score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html [18] Scikit-learn Developers, "RandomForestClassifier." <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> [19] Scikit-learn Developers, "Model evaluation: quantifying the quality of predictions." https://scikit-learn.org/stable/modules/model_evaluation.html [20] Scikit-learn Developers, "train_test_split." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html