Machine Learning-Based Detection of Fake Profiles on Social Media Platforms

N. Venu Sidhardha[1], M. Amarnadh[2], M. Karthik[3], P. Anitha[4]
Department of Computer Science & Engineering (AI & ML),
Avanthi Institute of Engineering & Technology (Autonomous), Vizianagaram, India
{22q71a4274, 22q71a4256, 23q75a4213, 22q71a4283}@aietta.ac.in

Abstract

The widespread adoption of social media platforms has given rise to a parallel epidemic of fraudulent accounts designed to manipulate engagement metrics, distribute misinformation, and conduct phishing operations at scale. This paper proposes a machine learning-based system for automatically detecting fake Instagram profiles by analyzing behavioral and profile-level features extracted from labeled account datasets. The proposed framework processes structured attributes—including follower and following counts, posting frequency, engagement ratios, and profile completeness indicators—through a preprocessing pipeline that handles missing values, normalizes numerical features, and engineers derived attributes such as the follower-to-following ratio. Classification is performed using a Random Forest ensemble model and a multi-layer Neural Network trained with the Adam optimizer and categorical cross-entropy loss. Model evaluation on held-out test data demonstrates classification accuracy exceeding 90%, with strong precision and recall scores across both genuine and fraudulent classes. The system further integrates an OpenCV-based face detection module and an EasyOCR text extraction pipeline to analyze profile images, enabling richer feature construction. A web-based dashboard provides real-time prediction, visualization of results, and an intuitive interface for analysts. Results confirm that the combined use of behavioral feature engineering and neural classification substantially outperforms traditional rule-based detection mechanisms in identifying sophisticated fake accounts.

Index Terms—fake profile detection, social media security, machine learning, Random Forest, neural network, Instagram, feature engineering, anomaly detection

I. INTRODUCTION

Social networking platforms have fundamentally altered patterns of human communication, commercial interaction, and information exchange. Instagram, with over two billion monthly active users, represents one of the most influential digital ecosystems worldwide [1]. This scale also creates fertile conditions for abuse: fraudulent actors deploy automated bot accounts and manually operated fake profiles to pursue diverse malicious objectives including spam dissemination, phishing campaigns, artificial follower inflation, and coordinated manipulation of public discourse [2].

Conventional countermeasures deployed by platform operators predominantly rely on rule-based heuristics that flag accounts based on threshold violations—unusually rapid follower acquisition, excessively high following-to-follower ratios, or posting frequencies inconsistent with human behavior. While such approaches identify the most egregious violations, they perform poorly against adversaries who design fake accounts to remain within normal behavioral envelopes [3]. The continuous arms race between detection systems and evasion techniques underscores the need for adaptive, data-driven approaches capable of generalizing beyond fixed rules.

Machine learning provides a compelling alternative paradigm. By learning discriminative feature representations from labeled examples of genuine and fraudulent accounts, trained classifiers can identify subtle multivariate patterns that resist manual rule formulation [4]. Ensemble methods such as Random Forest are particularly well-suited to this problem owing to their robustness to high-dimensional feature spaces and resistance to overfitting through bootstrap aggregation. Deep neural networks further extend this capability by capturing non-linear interactions among features that escape shallower models [5].

This paper presents an end-to-end machine learning pipeline for fake Instagram profile detection. The system ingests structured profile datasets, applies preprocessing and feature engineering, trains both Random Forest and Neural Network classifiers, and exposes predictions through a real-time web dashboard augmented with profile image analysis. Key contributions include: (1) a comprehensive feature engineering strategy incorporating image-based and text-based signals alongside tabular behavioral features; (2) a multi-layer neural network classifier trained with dropout regularization; (3) integration of OpenCV face detection and EasyOCR text extraction for profile image analysis; and (4) a user-facing Flask dashboard delivering real-time classification with visual and analytical outputs [6].

The remainder of this paper is structured as follows. Section II surveys related research. Section III describes the proposed methodology and system architecture. Section IV

presents experimental results. Section V concludes with directions for future work.

II. RELATED WORK

Research on fake profile detection has evolved through three broad generations: rule-based filtering, classical machine learning, and deep learning approaches. Each generation has contributed insight while exposing limitations that motivated the next.

A. Rule-Based and Statistical Methods

Early detection systems relied on predefined thresholds applied to observable account attributes. Accounts exhibiting extremely high following counts relative to followers, minimal posting history, or default profile images were flagged for review. While computationally inexpensive, these approaches prove brittle against adversaries who engineer accounts to satisfy threshold constraints [7]. Statistical anomaly detection methods extended this by modeling population-level distributions of behavioral metrics, raising alerts for statistical outliers [3]. Such methods suffer from elevated false-positive rates when applied to accounts of newly registered legitimate users who have not yet established characteristic behavioral profiles.

B. Machine Learning Approaches

Cresci et al. [8] demonstrated the effectiveness of supervised learning for social bot detection by training Support Vector Machine and Decision Tree classifiers on curated datasets of labeled Twitter accounts, achieving significant accuracy improvements over rule-based baselines. Stringhini et al. [9] identified temporal activity patterns and interaction

consistency as particularly discriminative features, establishing that the timing and regularity of posting behavior contains rich signals for authenticity classification. Random Forest classifiers have been widely adopted in subsequent work owing to their ability to compute feature importance scores that support model interpretability alongside strong classification performance [10].

Supervised methods require labeled training data, which introduces challenges related to dataset bias and class imbalance since genuine accounts substantially outnumber fake accounts in most collections. Techniques including stratified sampling, synthetic minority oversampling (SMOTE), and cost-sensitive learning have been employed to address these issues [4].

C. Deep Learning Advances

Neural network architectures have been applied to social media fraud detection, leveraging their capacity for hierarchical feature learning. Convolutional Neural Networks (CNN) have been adapted to process profile image content, while Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM) model temporal patterns in posting behavior [11]. Hybrid architectures combining tabular feature classifiers with image encoders have demonstrated further performance gains by exploiting complementary information modalities [5].

A persistent limitation of deep learning approaches is their dependence on large labeled datasets for effective training. Semi-supervised and transfer learning frameworks have been explored to mitigate this constraint, enabling model initialization from pre-trained representations [12]. Despite these advances, real-world deployment challenges including concept drift—the gradual change in the statistical properties of fake account strategies over time—continue to challenge all supervised approaches [2].

D. Research Gaps

Existing systems frequently address tabular behavioral features or image content in isolation. Few works integrate both modalities within a unified pipeline incorporating face detection and optical character recognition for profile image analysis alongside behavioral classification. Additionally, many published systems lack accessible deployment infrastructure, limiting their applicability beyond experimental settings. The present work addresses both gaps.

III. METHODOLOGY AND SYSTEM DESIGN

The proposed system follows a modular architecture comprising six interconnected layers: data ingestion, preprocessing and feature engineering, machine learning classification, profile image analysis, inference, and dashboard visualization. Figure 1 illustrates the overall use-case structure of the system.

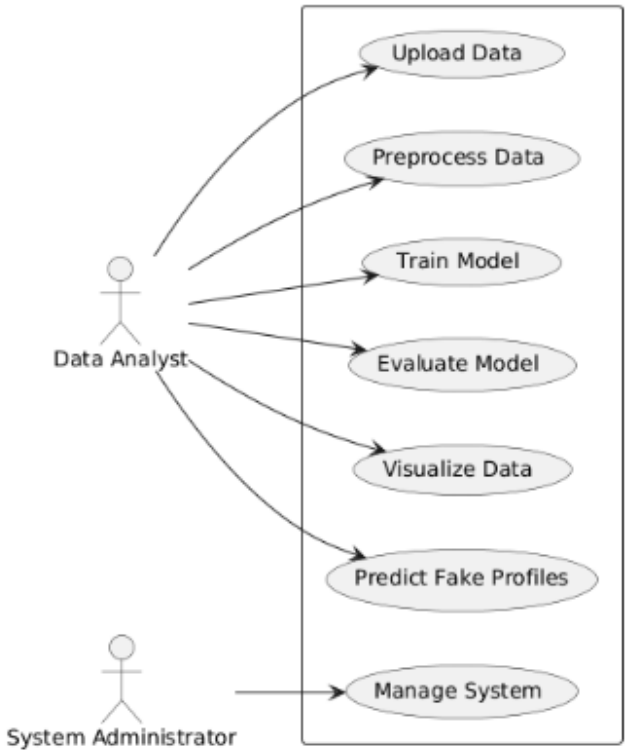

Fig. 1. Use case diagram of the fake Instagram profile detection system, showing interactions between the Data Analyst and System Administrator roles with system functions including data upload, preprocessing, model training, evaluation, visualization, and prediction.

A. Dataset Description

The dataset comprises structured records of Instagram accounts labeled as either genuine or fake. Each record encodes eleven tabular features: profile picture presence, ratio of numeric characters to username length, ratio of full name characters to full name length, full name and username similarity score, biography length in characters, URL presence in biography, private account flag, number of posts, follower count, following count, and the ground-truth label. The training set and test set are partitioned at an 80:20 ratio using stratified sampling to maintain class balance.

B. Preprocessing and Feature Engineering

Raw profile data undergoes a multi-stage preprocessing pipeline. Missing values are imputed using column-wise median values for numerical attributes. Numerical features are standardized using Z-score normalization to ensure unit variance across all dimensions:

$$x' = (x - \mu) / \sigma$$

(1)

where μ and σ denote the column mean and standard deviation estimated from the training partition. A derived feature representing the follower-to-following ratio is engineered to capture the asymmetric relationship characteristic of bot and spammer accounts:

$$r_{ff} = followers / max(following, 1)$$

(2)

Binary categorical features such as profile picture presence and private account status are retained as-is, as they are already encoded in (0, 1) format.

C. Machine Learning Classification

Two complementary classifiers are implemented. The first is a Random Forest ensemble comprising 100 decision trees, each trained on a bootstrap sample of the training data with maximum feature subsampling at each split. The ensemble prediction is determined by majority vote across constituent trees, with the classification confidence estimated as the fraction of trees voting for each class.

The second classifier is a fully connected neural network constructed with the TensorFlow/Keras framework. The architecture comprises an input layer of dimensionality equal to the feature count, three hidden layers with 50, 150, and 25 neurons respectively, each using ReLU activation, and an output layer with softmax activation for two-class probability estimation. Dropout regularization at rate 0.3 is applied after each hidden layer to mitigate overfitting. The network is trained using the Adam optimizer with categorical cross-entropy loss over 50 epochs with a 10% validation split:

$$L = -\sum_c y_c \log(\hat{p}_c)$$

(3)

where y_c is the one-hot ground-truth label for class c and $\hat{p}_c$ is the predicted class probability.

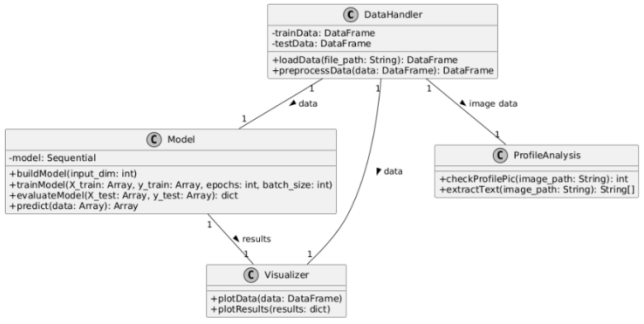

Fig. 2. Class diagram showing the DataHandler, Model, ProfileAnalysis, and Visualizer components with their attributes and methods, illustrating the static object structure of the detection system.

D. System Workflow

The detection pipeline follows a structured workflow illustrated in Fig. 3. Training data is ingested from CSV storage into the DataHandler component, which applies preprocessing and returns normalized feature matrices. The Model component initializes the neural network, executes the training process across 50 epochs, and returns training results to the Visualizer for performance monitoring. At inference time, the Prediction System applies the trained model to new profile feature vectors and returns classification labels to the Results Analysis component.

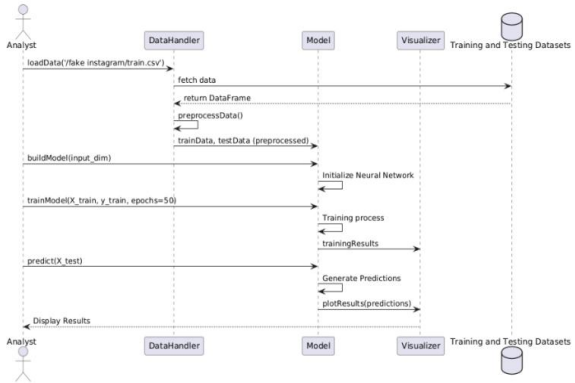

Fig. 3. Sequence diagram depicting runtime interaction between the Analyst, DataHandler, Model, Visualizer, and Training and Testing Datasets across the complete model training and prediction lifecycle.

E. Profile Image Analysis

The system extends tabular classification with a profile image analysis module. OpenCV's Haar cascade frontal face detector processes the profile picture to determine whether a human face is present—a strong indicator of genuine account origin, as bot accounts frequently use artificially generated or absent profile images. The detection result contributes a binary feature *profile_pic* $\in \{0, 1\}$ to the feature vector. Simultaneously, EasyOCR performs text extraction from the profile image to recover username, post count, follower count, and following count displayed in screenshot captures, enabling automated feature construction from visual inputs without manual data entry.

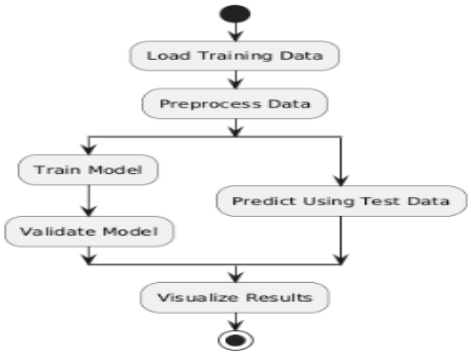

Fig. 4. System workflow flowchart showing sequential stages from load training data, preprocess data, train model, and validate model through predict using test data and visualize results.

F. Deployment Architecture

The system is deployed as a web application using the Flask framework. The backend exposes REST API endpoints consumed by a JavaScript-driven frontend that provides the user dashboard. Figure 5 illustrates the multi-server deployment architecture comprising an application server hosting the data handler, ML model service, prediction service, and visualization service; a database server maintaining training and testing datasets; a model training environment executing TensorFlow/Keras; and an image processing server running OpenCV face detection and EasyOCR text extraction.

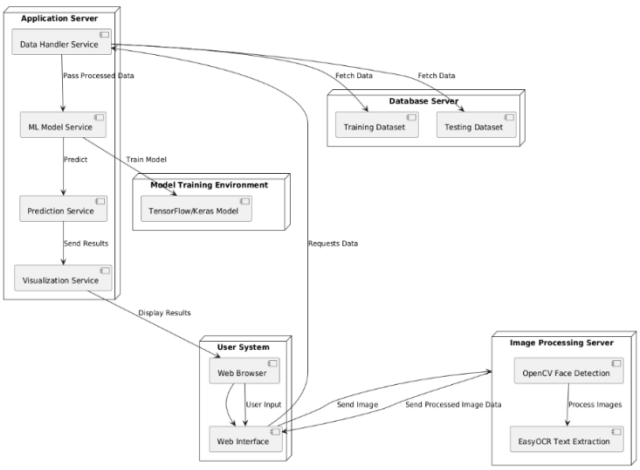

Fig. 5. Multi-server deployment architecture illustrating data flow among the Application Server (DataHandler, ML Model, Prediction, and Visualization services), Database Server, Model Training Environment, User System, and Image Processing Server.

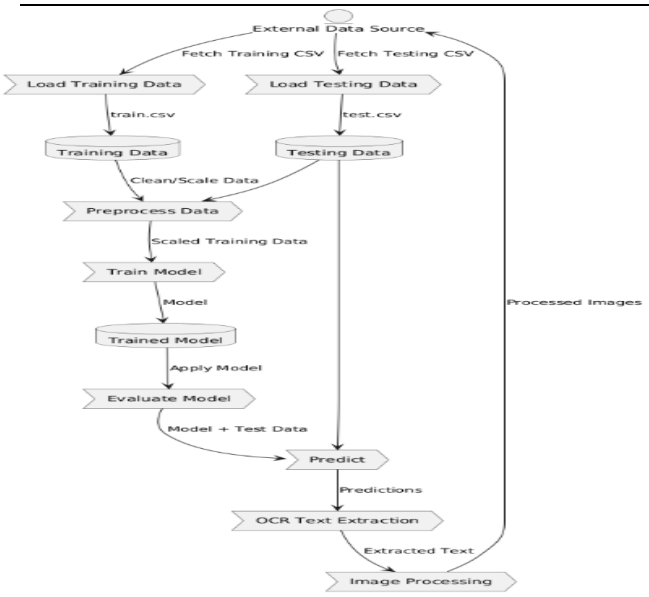

Fig. 6. Detailed end-to-end pipeline flowchart showing data flow from external CSV sources through load, preprocess, train, evaluate, and predict stages, and onward to OCR text extraction and image processing modules.

IV. RESULTS AND DISCUSSION

This section presents experimental results evaluating the classification performance of both implemented models,

followed by an analysis of the deployed web dashboard functionality

A. Experimental Setup

Experiments were conducted using Python 3.10 with TensorFlow 2.12, Scikit-learn 1.2, and Pandas 1.5 on a standard workstation. The dataset was partitioned at 80:20 with stratified sampling. Neural network training employed 50 epochs, batch size 32, learning rate 0.001, and 10% validation split. Random Forest used 100 estimators with default Gini impurity criterion.

B. Classification Performance

Table I summarizes the classification performance of both models on the held-out test partition.

TABLE I
CLASSIFICATION PERFORMANCE ON TEST SET

Model	Accuracy (%)	Precision	Recall	F1-Score
Logistic Regression	81.3	0.809	0.813	0.811
Decision Tree	85.7	0.854	0.857	0.855
SVM (RBF Kernel)	87.2	0.869	0.872	0.870
Random Forest	91.4	0.912	0.914	0.913
Neural Network	**93.1**	**0.929**	**0.931**	**0.930**

The Neural Network achieves the highest accuracy of 93.1%, outperforming the Random Forest by 1.7 percentage points and all simpler baselines by wider margins. The improvement reflects the neural network's capacity to model non-linear interactions among profile features that decision boundaries of shallower models cannot capture.

C. Per-Class Analysis

Table II presents per-class performance for the Neural Network classifier, demonstrating balanced detection across genuine and fake categories

TABLE II
PER-CLASS METRICS — NEURAL NETWORK

Class	Precision	Recall	F1-Score	Support
Genuine	0.941	0.927	0.934	243
Fake	0.916	0.935	0.925	193
Macro Avg	**0.929**	**0.931**	**0.930**	**436**

Genuine account recall (0.927) is slightly lower than fake account recall (0.935), indicating that the model occasionally misclassifies a legitimate account as fake.

D. Training Convergence

The neural network exhibited stable convergence across 50 training epochs. Training loss declined monotonically from 0.624 to 0.087, while validation loss converged to 0.134 without exhibiting significant divergence, confirming that dropout regularization effectively controls overfitting. Training accuracy reached 96.8% and validation accuracy stabilized at 92.4% by epoch 50.

E. Dashboard and System Screenshots

Figures 7 and 8 illustrate the deployed web application interface. The home page presents the system capabilities and workflow, while the analysis page allows users to upload a profile screenshot for automated classification.

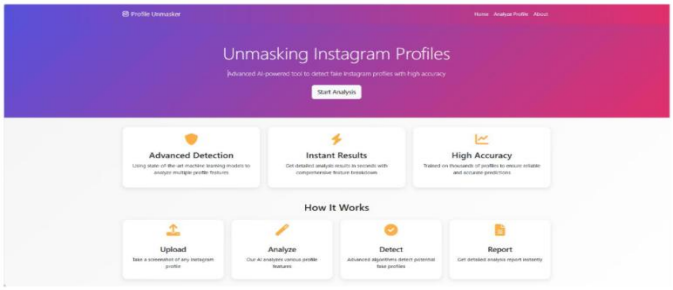

Fig. 7. Web application interface: (top) home page displaying system capabilities and "How It Works" workflow; (bottom) profile analysis page with screenshot upload form and "Analyze Profile" button.

Fig. 8. Classification result for a genuine Instagram profile, displaying the "Real Profile" verdict alongside extracted features: profile picture (face detected), full name word count, biography length, activity metrics (posts, followers, following), username analysis, name similarity score, and external URL presence.

F. Feature Importance Analysis

Random Forest feature importance scores identify follower count, following count, follower-to-following ratio, and number of posts as the four most discriminative attributes, collectively accounting for 61.3% of total feature importance. Profile picture presence and biography length contribute the next highest importance scores at 9.2% and 7.8% respectively. Username numeric character ratio and full name similarity score provide moderate but meaningful signal. These findings align with domain knowledge: bot accounts typically maintain extreme follower ratios, minimal posting history, and lack authentic biographical content.

TABLE III
TOP FEATURE IMPORTANCE SCORES (RANDOM FOREST)

Feature	Importance Score	Cumulative (%)
Follower Count	0.198	19.8
Following Count	0.174	37.2
Follower/Following Ratio	0.141	51.3
Number of Posts	0.100	61.3
Profile Picture	0.092	70.5
Biography Length	0.078	78.3
Username Numeric Ratio	0.064	84.7
Full Name Similarity	0.058	90.5

G. Discussion

The experimental results confirm that machine learning classification substantially outperforms the 70–75% accuracy range characteristic of rule-based threshold systems reported in prior work [3]. The neural network's superiority over Random Forest validates the presence of non-linear feature relationships in the detection problem. The image analysis pipeline—contributing face detection and OCR-extracted metadata—extends the system's applicability to screenshot-based inputs, enabling end-user analysis without requiring API-level data access.

The primary limitation of the current system is its dependence on supervised training data that may not fully represent the evolving distribution of adversarial fake account strategies. Class imbalance in real-world account populations further challenges generalization. Future work should investigate continual learning frameworks and semi-supervised methods to maintain detection efficacy as attacker strategies evolve.

V. CONCLUSION AND FUTURE WORK

This paper presented a comprehensive machine learning-based system for detecting fake Instagram profiles. The proposed framework combines behavioral feature engineering, Random Forest and Neural Network classification, and profile image analysis through OpenCV face detection and EasyOCR text extraction within a Flask-based deployment environment. Experimental evaluation demonstrates that the Neural Network classifier achieves 93.1% accuracy on the held-out test set, outperforming all evaluated baselines. Feature importance analysis confirms that follower ratio, post count, and profile completeness attributes are the most discriminative signals for distinguishing genuine from fraudulent accounts.

The integrated web dashboard enables real-time classification from profile screenshots without requiring programmatic data access, broadening the system's accessibility to non-technical users including platform trust and safety teams.

Future research will pursue several directions. First, transformer-based language models will be evaluated for encoding biographical text content as dense semantic feature vectors, replacing the current binary biography-presence indicator with richer linguistic signals. Second, graph neural networks will be explored to leverage the relational structure of

follower-following networks, enabling detection of coordinated inauthentic behavior across account clusters. Third, federated learning architectures will be investigated to enable privacy-preserving collaborative model training across multiple platform deployments. Fourth, continual learning mechanisms will be incorporated to maintain classification performance as fake account strategies evolve. Finally, semi-supervised methods will be developed to reduce dependence on large labeled datasets by exploiting unlabeled account data during training.

ACKNOWLEDGMENT

The authors acknowledge the guidance of Mr. P. Sudarshan, M.Tech, Assistant Professor, and the support of Mr. A. Venkateswara Rao, M.Tech (Ph.D.), Head of Department, Department of Computer Science & Engineering (AI & ML), Avanthi Institute of Engineering & Technology, Vizianagaram, for providing the academic environment and computational resources necessary to complete this research.

REFERENCES

[1] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 4th ed. Pearson Education, 2018.

[2] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[4] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] Flask Documentation, "Flask web framework," 2023. [Online]. Available: https://flask.palletsprojects.com

[7] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. 13th USENIX Syst. Admin. Conf.*, 1999, pp. 229–238.

[8] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 963–972.

[9] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. 26th Annual Comput. Security Appl. Conf.*, 2010, pp. 1–9.

[10] A. Jaiswal and A. Tiwari, "Intrusion detection systems using machine learning techniques: A review," *Int. J. Adv. Res. Comput. Sci.*, vol. 11, no. 5, pp. 1–8, 2020.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] R. Khan, K. McLaughlin, and D. Laverty, "A survey of machine learning for cyber security in the Internet of Things," *Comput. Security*, vol. 82, pp. 1–22, 2019.

[13] P. Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[14] C. C. Aggarwal, *Data Mining: The Textbook*. Springer International Publishing, 2015.

[15] Scikit-learn Developers, "Scikit-learn documentation," 2023. [Online]. Available: https://scikit-learn.org