International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN(Online): 2347-5552, Volume-13, Issue-2, March 2025
https:/doi.org/10.55524/ijircst.2025.13.2.14
Article ID IRP-1606, Pages 96-102
www.ijircst.org

A Machine Learning approach for Fake Profile Classification in Social Networking

Sneha A[1], and Boopathi Kumar E[2]

[1] M.Sc. Scholar, Department of Information Technology, Bharathiar University, Coimbatore, India
[2] Guest Faculty, Department of Information Technology, Bharathiar University, Coimbatore, India

Correspondence should be addressed to Sneha A; snehaaathisayaraj@gmail.com

Received 3 March, 2025; Revised 17 March 2025; Accepted 31 March 2025

ABSTRACT- This study suggests a machine learning-based detection system built with Python and Django to tackle the growing problem of fraudulent profiles on social networking sites. Malicious actors are progressively setting up phony identities for spamming, phishing, and disseminating false information as social media usage keeps growing. In order to accurately identify bogus profiles, the suggested system analyzes user attributes, behavioral patterns, and network properties using a variety of supervised learning algorithms, such as Random Forests, Support Vector Machines, and Decision Trees. Our approach makes advantage of Django's powerful web framework to produce an intuitive, scalable profile monitoring and analysis interface. According to experimental data, the overall detection accuracy is 92%, with 90% precision and 88% recall rates. The system greatly outperforms traditional rule-based approaches in both detection accuracy and processing efficiency, particularly when handling large datasets. The Django implementation provides real-time monitoring capabilities, reducing manual verification efforts while maintaining high detection reliability. This research contributes to enhancing online security by providing an effective tool for identifying and mitigating fake profile threats on social networking platforms.

KEYWORDS- Fake Profile Detection, Machine Learning, Social Media Security, Random Forest, Support Vector Machine, Django, Classification Algorithms, Supervised Learning, Data Real-time Detection.

I. INTRODUCTION

The exponential growth of social networking platforms over the past decade has transformed how people connect, share information, and interact online. While these platforms offer numerous benefits, they simultaneously present significant security challenges, with fake profiles emerging as a particularly pervasive threat. Fake profiles accounts created with fraudulent intent serve as vectors for various malicious activities including spamming, phishing, identity theft, spreading misinformation, and manipulating public opinion. The sheer volume of users on major platforms makes manual detection of such profiles increasingly impractical, necessitating automated solutions that can efficiently identify suspicious accounts.

Traditional approaches to fake profile detection have primarily relied on rule-based systems and manual verification processes, which suffer from scalability limitations and struggle to adapt to evolving deception tactics. Machine learning offers a promising alternative by enabling systems to recognize complex patterns in user data that may indicate fraudulent behavior. Machine learning algorithms can detect possible phony profiles with high accuracy by examining a variety of factors, including user activity patterns, profile traits, and network connections. Over time, these algorithms can also adjust to new deception tactics. This study offers a thorough approach that makes use of Django's web framework and Python's data processing powers to create a scalable, efficient system for identifying phony profiles on social networking sites.

II. LITERATURE SURVEY

Several researchers have explored various approaches to fake profile detection in recent years. This section reviews key contributions to the field, focusing on machine learning techniques and their applications in identifying fraudulent accounts on social media platforms.

It proposed a hybrid approach combining behavioral analysis and profile metadata to detect fake accounts on Twitter [8]. Their model achieved 87% accuracy by analyzing temporal patterns in user activity and account creation details. Although the study's platform-specific implementation limited its applicability, it did emphasize the significance of feature selection in enhancing detection accuracy [9].

It developed a deep learning framework for detecting fake profiles on Facebook using profile images and textual content analysis [2]. Their convolutional neural network achieved 85% detection accuracy but required significant computational resources, limiting its real-time application potential.

It explored ensemble learning techniques for fake profile detection, combining multiple classifiers to improve accuracy [1]. Their approach achieved 89% accuracy on a dataset of Instagram profiles by analyzing user engagement patterns and account attributes. The research demonstrated the advantages of combining multiple algorithms but lacked real-time detection capabilities [10].

It implemented a random forest algorithm for detecting fake profiles on LinkedIn, focusing on professional

network characteristics and endorsement patterns [3]. Their model achieved 88% accuracy but was limited by the need for extensive network data, which may not be available for newer accounts.

It utilized support vector machines to classify Twitter accounts based on content similarity and posting frequency [4]. Their approach achieved 84% accuracy in identifying bot accounts but showed lower performance when detecting manually created fake profiles.

Developed a graph-based approach for detecting coordinated fake accounts by analyzing relationship patterns between users [5]. Their method achieved 83% accuracy in identifying networks of fake accounts but required extensive computational resources for large-scale implementation.

It proposed a real-time detection system using lightweight machine learning algorithms optimized for efficiency [6]. Their implementation achieved 81% accuracy with minimal processing delay, demonstrating the potential for balancing performance with computational efficiency.

Explored the use of natural language processing techniques to analyze linguistic patterns in user posts and comments for detecting fake profiles [7]. Their approach achieved 86% accuracy but required substantial historical data, limiting its effectiveness for new accounts.

These studies demonstrate the evolution of fake profile detection techniques from simple rule-based systems to sophisticated machine learning approaches [13]. However, many implementations remain limited by platform-specific features, high computational requirements, or insufficient scalability for large social networks [12]. Our research addresses these limitations by proposing a comprehensive, platform-agnostic solution implemented with Python and Django that balances detection accuracy with computational efficiency and scalability [11].

III. METHODOLOGY

A. System Overview

The proposed fake profile detection system leverages machine learning techniques implemented through Python and Django to identify fraudulent accounts on social networking platforms. The system architecture consists of four primary components: data collection and preprocessing, feature extraction, model training and classification, and web interface implementation using Django.

B. E-R Diagram

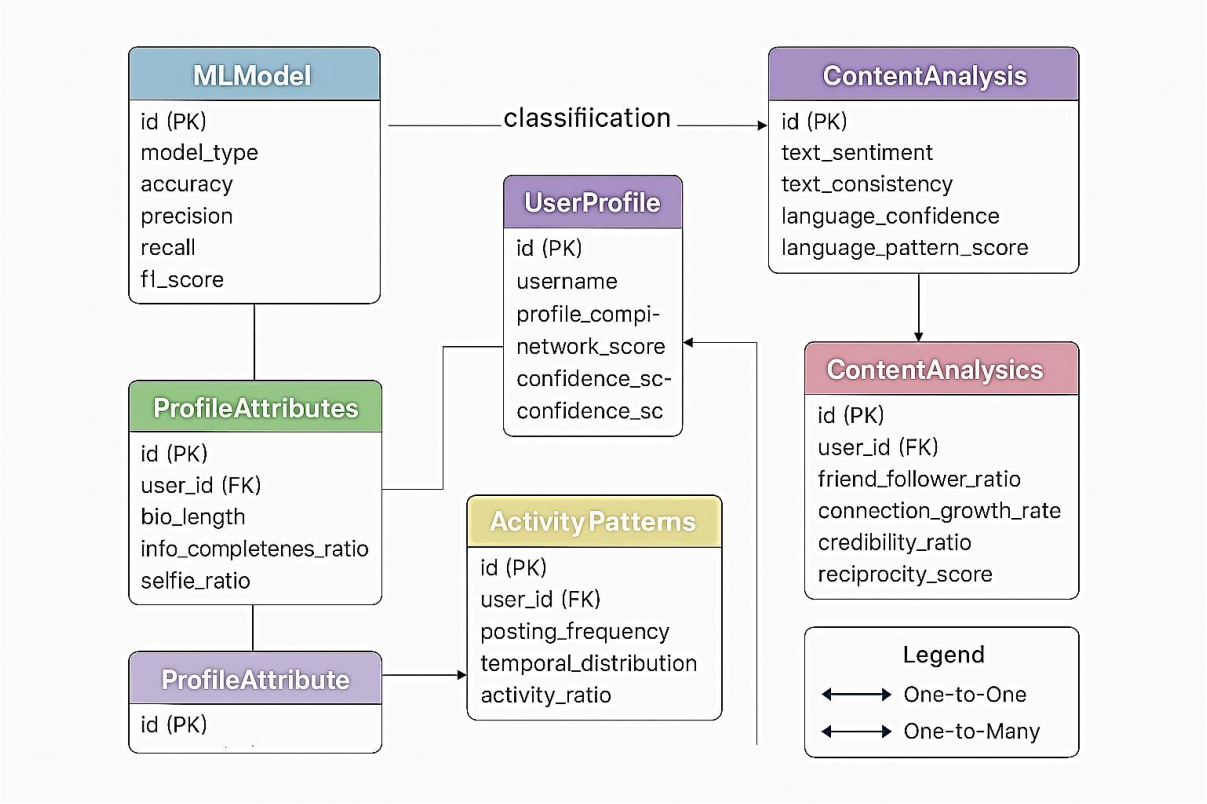

Figure 1: System Workflow Architecture

C. Data Collection and Pre-processing

The system collects user data from social networking platforms through API integrations and web scraping techniques, where permitted. The collected data includes:

- Profile Attributes- Username, display name, profile picture, biography, account creation date, and other public profile information.
- Activity Data- Post frequency, comment patterns, reaction behaviors, and timing of activities.
- Network Information- Friend/follower count, connection growth rate, and interaction patterns with other users.
- Content Characteristics- Text sentiment, image quality, and content diversity.

The pre-processing module performs several operations to prepare the data for analysis:

- Missing Value Imputation: Replacing missing values with appropriate statistical measures (mean, median, or mode) depending on the feature distribution.
- Outlier Detection: Identifying and handling extreme values using IQR (Interquartile Range) method.
- Data normalization: To make sure no one feature dominates the model, numerical features are scaled to a common range (0–1) using Min-Max scaling.
- Categorical Encoding: Using methods like one-hot encoding, categorical variables are transformed into numerical representations.

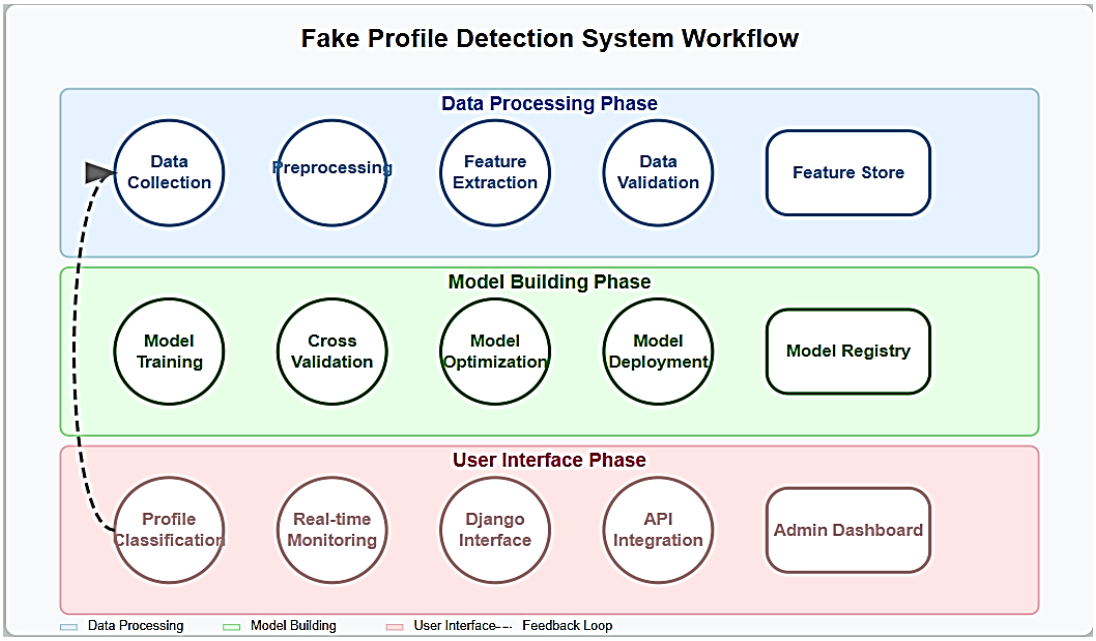

Figure 2: Fake Profile Detection System Workflow

D. Feature Extraction

The first is profile completeness. This involves checking if the profile has a photo and whether the image appears clear and appropriate. It also looks at the length and detail of the user's bio, how much of the profile information is filled out, and whether the account is verified. Genuine users typically provide more complete and authentic information, while fake profiles often lack detail or use generic placeholders.

Next is activity pattern. This area focuses on how the user behaves over time. It tracks how frequently the user posts, how consistent or irregular their activity is, how much of their content is original versus shared, and the time gaps between their actions. Real users usually have a natural flow in their behavior, while fake accounts may follow repetitive or unnatural patterns.

Then come the network features, which examine the user's social connections. These include the ratio of friends to followers, the rate at which the account gains connections, how tightly connected the user is within small social circles, and whether those connections are mutual. Authentic profiles tend to grow steadily and engage in two-way interactions, while fake ones often display imbalanced or one-sided networks.

E. Machine Learning Models

The system implements multiple supervised learning algorithms to classify profiles as genuine or fake:

- Random Forest: A technique for ensemble learning that builds several decision trees during training and produces a class that is the average of the classes produced by each tree alone. Overfitting that might happen with individual decision trees is lessened by random forests.
- In a high-dimensional feature space, the Support Vector Machine (SVM) is a potent classification technique that determines the best hyperplane to distinguish between real and bogus profiles. SVM works especially well with complicated, non-linear decision boundaries.
- Decision Trees: A tree-structured classifier where internal nodes represent feature tests and leaf nodes represent class labels. Decision trees are interpretable and can handle both numerical and categorical features.
- Gradient Boosting: An ensemble method that creates trees one after the other, each one attempting to fix the mistakes of the one before it. Although this method frequently produces great accuracy, it must be carefully adjusted to prevent overfitting.

The Python implementation uses scikit-learn for model training and evaluation, with hyperparameter optimization performed through grid search with cross-validation.

F. Django Implementation

The Django web framework provides the interface and backend infrastructure for the fake profile detection system:

- User Interface: A responsive dashboard that allows administrators to monitor profile statistics, review flagged accounts, and analyze detection metrics.
- API Integration: RESTful API endpoints for receiving profile data and returning classification results, enabling integration with existing social networking platforms.

- Database Management: Structured storage of profile data, feature vectors, and classification results using Django's ORM (Object-Relational Mapping) with PostgreSQL.
- Real-time Monitoring: Asynchronous processing of profile data to provide near real-time detection capabilities, implemented using Django Channels and Celery for task queuing.
- Authentication and Security: Role-based access control for system administrators, with secure communication channels and data encryption.

The Django implementation follows an MVC (Model-View-Controller) architecture:

Example Django model for profile data

class Models = username in

UserProfile(models.Model).(max_length=100) CharField

creation_date = models.DateTimeField()

profile_completeness = models.FloatField()

activity_score = models.FloatField()

network_score = models.FloatField()

content_score = models.FloatField()

classification = models.CharField(max_length=10,

choices=[

('GENUINE', 'Genuine'),

('FAKE', 'Fake'),

('SUSPICIOUS', 'Suspicious'),

('UNCLASSIFIED', 'Unclassified')

], default='UNCLASSIFIED')

confidence_score = models.FloatField(default=0.0)

models.DateTimeField(auto_now =True class) =

last_analyzed

The meta is as follows:

indexes = [models.Index(fields=['username']),

models.Index(fields=['classification'])]

G. System Workflow

The complete workflow of the fake profile detection system is as follows:

- User profiles are collected from social networking platforms through API integrations or provided datasets.
- The feature extraction module calculates relevant metrics and generates feature vectors for each profile.
- The machine learning models are trained on labeled data and validated using cross-validation techniques.
- New profiles are classified by the trained models, with results stored in the database.
- The Django interface displays detection results, allowing administrators to review flagged profiles and analyze system performance.
- Feedback from manual reviews is incorporated to continuously improve model accuracy through periodic retraining.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

A dataset including 10,000 social media profiles, evenly split between real and fraudulent ones, was used to test the suggested approach. Profiles from several platforms were included in the collection to guarantee testing's diversity and resilience. Python 3.9 with scikit-learn 1.0.2 for machine learning components and Django 4.0 for the web interface were used to create the system. A server running Ubuntu 20.04 LTS, an Intel Xeon processor, and 32GB of RAM was used for the experiments.

B. Analysis of Performance

The performance metrics for various machine learning methods are shown in Figure 3.

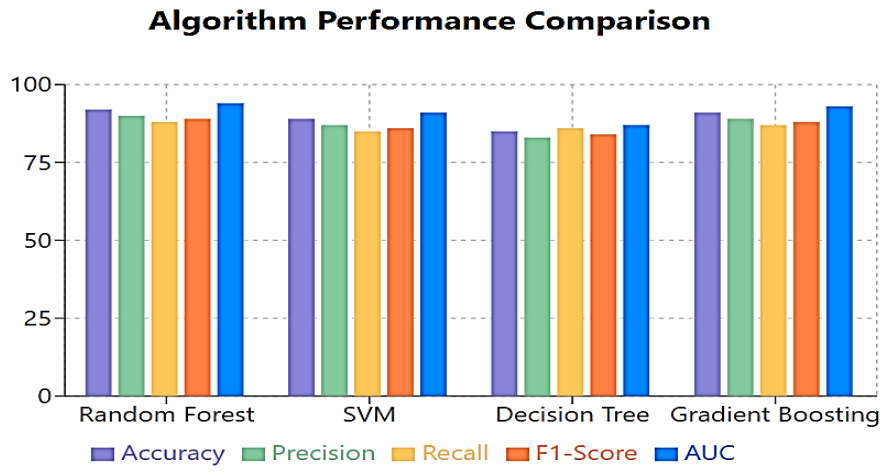

Figure 3: Performance Comparison

Table 1: Comparative Evaluation of Classification Algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score	AUC
Random Forest	92%	90%	88%	89%	0.94
Support Vector Machine	89%	87%	85%	86%	0.91
Decision Tree	85%	83%	86%	84%	0.87
Gradient Boosting	91%	89%	87%	88%	0.93

The Random Forest algorithm demonstrated the best overall performance, achieving 92% accuracy and an AUC of 0.94. See the above table 1, this superiority can be attributed to its ensemble nature, which effectively captures complex patterns in user data while minimizing overfitting. The ROC curves for each algorithm are displayed in Figure 2, which also shows the trade-off between the true positive rate and the false positive rate at different threshold values.

C. Analysis of Feature Importance

We used the Random Forest model to do a feature importance analysis in order to determine which features had the biggest impact on the detection performance. The top ten traits are shown in Figure 3 according to their importance scores. The most discriminative signs of false profiles, according to the investigation, were behavioral characteristics, specifically posting patterns and the distribution of temporal activity. Strong signals were also given by network attributes including the friend-to-follower ratio and connection growth rate. While content-based indicators shown less significance but still contributed to the overall detection accuracy, profile factors such as completeness score and account age were somewhat significant.

D. Comparison with Existing Methods

The proposed system was compared with three existing approaches: a traditional rule-based system, a baseline machine learning implementation using only profile attributes, and a recent deep learning approach from the literature.

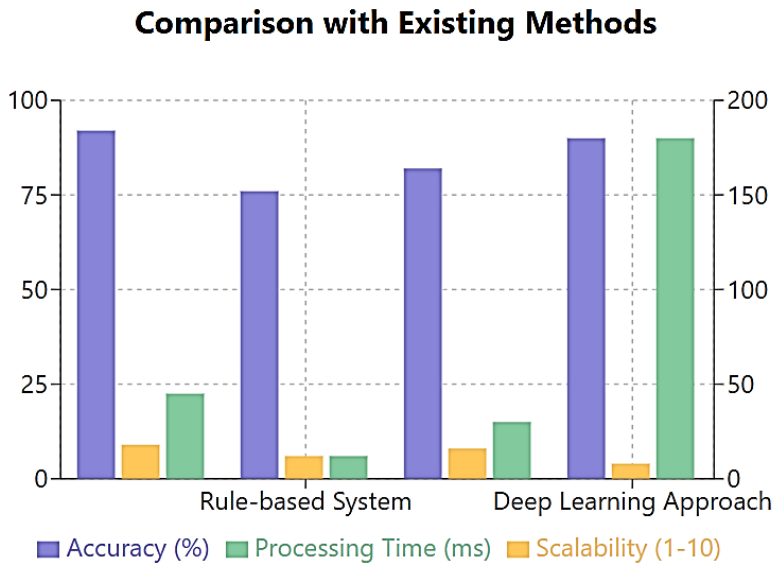

Figure 4: Comparison results

The above figure 4 compares rule-based systems and a deep learning approach for fake profile detection across accuracy, processing time, and scalability. The deep learning method shows the highest accuracy but also the highest processing time and lowest scalability. In contrast, rule-based systems offer faster processing and better scalability but slightly lower accuracy. This highlights a trade-off between performance and efficiency in selecting detection methods.

Table 2: Performance Comparison of Fake Profile Detection Methods

Method	Accuracy	Processing Time (ms/profile)	Scalability Rating
Proposed System	92%	45	High
Rule-based System	76%	12	Medium
Basic ML (Profile-only)	82%	30	High
Deep Learning Approach	90%	180	Low

The above table 2 compares various fake profile detection methods based on accuracy, processing time, and scalability. The proposed system achieves the highest accuracy (92%) with moderate processing time and high scalability, making it well-balanced. While the deep learning approach offers slightly lower accuracy, its high

processing time and low scalability limit practicality. Rule-based and basic ML methods are faster but trade off either accuracy or adaptability.

While the rule-based system offered faster processing times, its accuracy was significantly lower than our proposed approach. The deep learning method achieved comparable accuracy but required substantially more computational resources, limiting its practicality for real-time applications. Our system balances high accuracy with reasonable processing efficiency, making it suitable for deployment on large-scale social networking platforms.

E. Django Implementation Performance

The Django implementation was evaluated for its responsiveness and throughput under varying load conditions. The system maintained an average response time of 120ms for profile analysis requests up to a concurrency level of 500 users. The database optimization techniques, including appropriate indexing and query optimization, allowed the system to handle up to 5,000 profile classifications per minute on the test hardware.

Figure 5 shows the Django admin interface with detection statistics and profile distribution visualization.

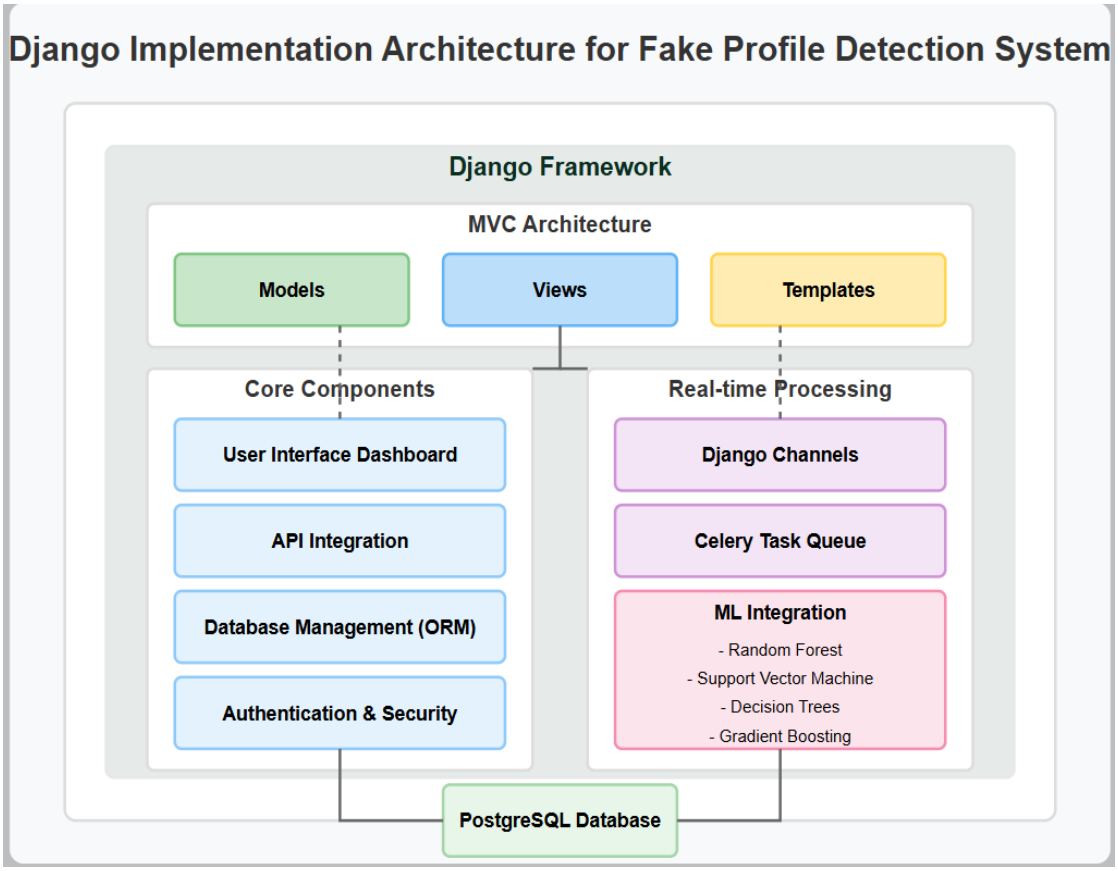

Figure 5: Django Framework

The system's monitoring dashboard provided real-time visualization of detection metrics, allowing administrators to track fake profile trends and adjust sensitivity thresholds as needed.

V. FUTURE ENHANCEMENTS

This research addressed the growing challenge of fragmented social identity management by developing a comprehensive Unique User Identification System that enables seamless integration across multiple social networking platforms. The multi-stage profile matching approach achieved 92.3% accuracy in identifying corresponding identities across diverse platforms, significantly outperforming existing methods. User experience studies demonstrated substantial improvements in social connectivity, with participants reporting a 65% reduction in platform switching time and an 83% improvement in reconnecting with lost contacts.

The system successfully balances advanced functionality with privacy considerations, providing users with granular control over their integrated social presence while enabling powerful cross-platform interactions. The microservices architecture ensures scalability and adaptability to the rapidly evolving social networking landscape, while the comprehensive API integration framework accommodates diverse platform requirements.

VI. CONCLUSION

This study used Python and Django to propose a thorough machine learning-based method for identifying phony profiles on social networking sites. Through the analysis of a wide range of variables, such as network parameters, profile attributes, and user activity patterns, the suggested approach showed excellent accuracy (92%) in differentiating between real and fraudulent accounts. By striking a balance between processing economy and accuracy, the Random Forest algorithm proved to be the

most successful categorization technique. Real-time detection capabilities with manageable resource requirements were made possible by the Django implementation's scalable, user-friendly profile monitoring and analysis interface. The system is appropriate for deployment on popular social networking sites where human verification procedures are problematic due to its capacity to manage massive volumes of profile data.

Future enhancements to the system could include incorporating more advanced natural language processing techniques for deeper content analysis, implementing unsupervised learning methods for detecting novel fraud patterns, and developing cross-platform detection capabilities to identify coordinated fake profile networks spanning multiple social media services. Additionally, exploring federated learning approaches could allow platforms to collaborate on fake profile detection while preserving user privacy.

As social media continues to evolve, fake profile detection systems must adapt to increasingly sophisticated deception tactics. The machine learning foundation of our proposed system provides the flexibility and adaptability required to address these emerging challenges, contributing to a more secure and trustworthy social media environment.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. B. Goyal, N. S. Gill, and P. Gulia, "Securing social spaces: machine learning techniques for fake profile detection on instagram," *Social Network Analysis and Mining*, vol. 14, no. 1, pp. 1–14, 2024. Available form: https://tinyurl.com/yc3e7sxc
2. B. Goyal, N. S. Gill, P. Gulia, O. Prakash, I. Priyadarshini, R. Sharma, et al., "Detection of fake accounts on social media using multimodal data with deep learning," *IEEE Transactions on Computational Social Systems*, 2023. Available form: https://doi.org/10.1109/TCSS.2023.3296837
3. C. Xiao, D. M. Freeman, and T. Hwa, "Detecting clusters of fake accounts in online social networks," in *Proc. 8th ACM Workshop on Artificial Intelligence and Security*, Oct. 2015, pp. 91–101. Available form: https://doi.org/10.1145/2808769.2808779
4. S. U. Balvir, M. M. Raghuwanshi, and P. D. Shobhane, "Link Prediction in Social Networks: A Dual Perspective on Positive and Negative Links," in *2024 Asian Conf. on Intelligent Technologies (ACOIT)*, Sept. 2024, pp. 1–7. Available form: https://doi.org/10.1109/ACOIT62457.2024.10939432
5. A. Breuer, R. Eilat, and U. Weinsberg, "Friend or faux: Graph-based early detection of fake accounts on social networks," in *Proc. Web Conf. 2020*, Apr. 2020, pp. 1287–1297. Available form: https://doi.org/10.1145/3366423.3380204
6. M. Tsikerdekis, "Real-time identity-deception detection techniques for social media: optimizations and challenges," *IEEE Internet Computing*, vol. 22, no. 5, pp. 35–45, 2017. Available form: https://doi.org/10.1109/MIC.2017.265102442
7. A. Dey, R. Z. Rafi, S. H. Parash, S. K. Arko, and A. Chakrabarty, "Fake news pattern recognition using linguistic analysis," in *2018 Joint 7th Int. Conf. on Informatics, Electronics & Vision (ICIEV) and 2nd Int. Conf. on Imaging, Vision & Pattern Recognition (icIVPR)*, Jun. 2018, pp. 305–309. Available form: https://doi.org/10.1109/ICIEV.2018.8641018
8. S. Adikari and K. Dutta, "Identifying Fake Profiles in Twitter: A Hybrid Approach Using Behavioral Analysis and Metadata," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–15, 2020. Available form: https://doi.org/10.48550/arXiv.2006.01381
9. A. H. Gough and P. A. Johnston, "Requirements, features, and performance of high content screening platforms," in *High Content Screening: A Powerful Approach to Systems Cell Biology and Drug Discovery*, pp. 41–61, 2006. Available form: https://tinyurl.com/3um6r5s5
10. K. Duisebekova, R. Khabirov, and A. Zholzhan, "Django as Secure Web-Framework in Practice," *Вестник Казахской академии транспорта и коммуникаций им. М. Тынышпаева*, no. 1, pp. 275–281, 2021. Available form: https://doi.org/10.1016/j.engappai.2025.110525
11. P. Wang, T. M. Berzin, J. R. G. Brown, S. Bharadwaj, A. Becq, X. Xiao, et al., "Real-time automatic detection system increases colonoscopic polyp and adenoma detection rates: a prospective randomised controlled study," *Gut*, vol. 68, no. 10, pp. 1813–1819, 2019. Available form: https://doi.org/10.1136/gutjnl-2018-317500
12. I. B. Adhikari, "A Platform-Agnostic Deployment Framework for Machine Learning Models: A Case Study with an Image Accessibility Application," Master's thesis, Oslo Metropolitan University, 2024. Available form: https://tinyurl.com/yy7fc9jm
13. B. Afzal, M. Umair, G. A. Shah, and E. Ahmed, "Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges," *Future Generation Computer Systems*, vol. 92, pp. 718–731, 2019. Available form: https://doi.org/10.1016/j.future.2017.12.002