INSPIRE SMART SYSTEMS

Inspire Smart Systems Journal
ISSN: 3110-9578, Vol. 1, No. 1, 2026
doi.org/10.65718/inspireSS.2026.3005

InspireQuill.org
Inspire Academic Press (Canada)

Research Paper

Detection of fake accounts promoting Cyber threats using Machine Learning Methods

Hayder Adnan Al-Hasani [1, *], Salah A. K. Albermany [2]

[1] Computer Science Department, University of Kufa, Iraq
[*] Corresponding Author: hayderaalhasani@student.uokufa.edu.iq

Abstract

A large number of fake online accounts creates difficulties for cybersecurity, since the owners of these accounts can share lies, plan assaults and boost cyber problems. In our research, we apply a strictly bounded approach that makes use of machine learning to identify fake accounts using their associated noncontent features. We built a database with 10,000 accounts (half of which were fake, half genuine) taken from (Twitter/X, Instagram) and then looked at nine main features such as account age, the relationship between followers and people the account follows, how often posts are made, how long each post's gap is, average session duration, the spread of IP addresses, the number of different devices used and fast switching of IP addresses. After the data was cleaned, normalized and data imbalances were corrected with SMOTE if the ratio was higher than 1.5 to 1, a hyperparameters optimized Random Forest classifier with 100 trees were tuned using 5-fold cross validation. For the fake account class, the model got 91.0 % accuracy, 93.4 % precision, 89.2 % recall and a 91.3 % F_1 score using the 3,000 hold out test set. Performing learning curves and permutation tests, we confirmed that the highlights of the project were reliable and significant. Testing on 1,000 new account profiles revealed that it took less than ten milliseconds to infer connections for each account (performing as expected for 94 % of cases). Using public data restriction, approval from an ethics board, keeping logs for a short period and being transparent help to use AI responsibly. Based on our findings, Metadata classifiers can effectively and fast stop attacks caused by fake accounts.

Keywords: Fake account detection; cyber threats; metadata analysis; behavioral features; supervised machine learning; Random Forest; real-time inference; SMOTE; ethical AI.

1. Introduction

There has been a rising number of fake accounts created online, making it easier for fraudsters to pass false news, launch multiple attacks at once and broaden phishing scams. Yang & Menczer (2023) have discovered that today's AI botnets enlist many asynchronous devices to execute harmful actions using very little supervision from humans. Furthermore, thanks to AI, identifying accounts from fake users has become much harder since there are now AI-generated images in social media accounts: estimates project that this occurs in 0.021 %–0.052 % of profiles

Academic Editor:
Nazeeruddin Mohammad

Received: 23/07/2025
Revised: 19/08/2025
Accepted: 05/10/2025
Published: 01/01/2026

Citation
Al-Hasani, H. A., Albermany, S. A. K. (2026). Detection of fake accounts promoting Cyber threats using Machine Learning Methods. *Inspire Smart Systems Journal, 1*(1), 51-61.

(Yang, Singh, & Menczer, 2024; Ricker et al., 2024). Since these fake accounts send out scams and mislead users by appearing as respected individuals or groups, promptly detecting them is crucial (Kemp, Kalutarage, & Al Kadri, 2025).

There are many approaches and areas that existing studies on fake accounts and similar cyber threats cover. With machine learning, researchers use complex classifiers to analyze news content and other metadata, achieving positive results while facing gradually increasing tricky tactics (Choras et al., 2020). Using graph learning, researchers have managed to model the structure of transaction networks and spot unusual activity, achieving high levels of spotting illicit happenings (Li, Liu, Ma, Yang, & Sun, 2023). Similarly, people have used hybrid features including URL aspects and heuristics from HTML (Karim, Shahroz, Mustofa, Belhaouari, & Joga, 2023) as well as systems that blend visual, textual and behavioral traits (Chai, Zhou, Li, & Jiang, 2022; Asiri, Xiao, Alzahrani, & Li, 2024).

Most solutions to these problems either require a lot of content analysis, leading to privacy and complexity issues or they require complex and not always available multi modal information (Alhassun & Rassam, 2022; Zavrak & Yilmaz, 2023). Experts recognize that choosing one aspect of social cybersecurity over the others often results in its deterioration in other areas (Mulahuwaish et al., 2025; Orabi, Mouheb, Al Aghbari, & Kamel, 2020). People are now interested in using light methods that rely on commonly accessible data such as when users make accounts, how active they are on the platform and the network of their "friends" to detect fake accounts as soon as they appear, while not affecting privacy. Based on this trend, we have focused our studies solely on using metadata and behavior training to detect cyber threat accounts. If we do not rely on text or image information, we can cut down on computing and ensure strong privacy. We believe that by choosing a selected group of non-content signals and running them through a best-suited Random Forest classifier, we can rival the superiority of complex models, while still ensuring that it takes less than 50 milliseconds to make a decision for each applicant. In the following sections, we explain how our data was cleaned, features were added, a model was programmed and tests were done, all to support our defense against fake account–based cyber-attacks.

2. Methodology

In our study, we rely on accounting data and machine learning classification to evaluate the results. We intend to develop and uphold a model that identifies fake accounts, created for cyber-attacks, from real users, using nothing but basic information. By only focusing on metadata and activity statistics, we can avoid any confusion introduced by looking at messages or connectivity, helping us clearly identify why the results are as they are.

First, we will put together a set of 10,000 accounts, making sure they are balanced and labeled and use evidence collected by the platform and by previous forensic investigators to assign the appropriate labels. After the cleaning and standardization steps, the team will pick out the chosen features such as how much time an account has been online, its ratio of following versus following, number of posts, session lengths, the variety of IP addresses and number of unique device agents noted. By analyzing the results, we will be able to choose the 15 most important features that are linked to fake accounts.

A model will then be trained on 70% of the dataset with cross validation to ensure that the best tree depth and number of minimum samples in a leaf work for each fold. The rest of the data will be used to test the model, looking at precision, recall and F_1 score for identifying the "fake" images and simply accuracy of the predictions for all types of images. A review of how the model responds to different sizes of training data and a check known as permutation testing will ensure reliability, while the evaluation of a precision–recall curve will tell us the best way to set the detection threshold. At the end, we will perform real time inference to check that the time taken for

account classification is under 50 milliseconds and that all privacy and ethical standards are met throughout the process by using only public types of metadata.

2.1. Data Acquisition & Preparation

2.1.1. Inclusion Criteria

The first step is to outline which accounts are included in our study. As mentioned above, accounts are categorized as fake if the platform has removed them due to cyber threats, but accounts are genuine only when they have never had their accounts removed for such reasons. Since we want to apply our model in different settings, we are collecting from at least two big platforms (Twitter/X, Instagram).

2.1.2. Dataset Assembly

In accordance with this, we are making a corpus with 10,000 accounts, ensuring that fake and genuine accounts make up an equal amount. While collecting data, we do not include private messages or any private details about the users. The raw data is now prepared for cleaning.

2.1.3 Data Cleaning

After that, we focus on preparing the raw data by carrying out two main processes. Then, we will not consider any account that has > 30 % missing information about our main features (e.g., session duration, posting frequency). In addition, we find and remove accounts that are posting far beyond others by using the interquartile range and ensuring they do not impact the training.

2.1.4 Normalization & Balancing

All continuous features are adjusted to the [0, 1] range after cleaning. Also, categorical features are turned into one hot vector. Once we finish comparing imposter vs. real, if the ratio is greater than 1.5 : 1, we use SMOTE to create synthetic data and bring the two distributions the same as shown in Figure 1. Thanks to the prepared data, we can confidently use it for choosing features and training a model.

2.2. Feature Extraction & Selection

At the Feature Extraction stage, we collect three sets of signals from the information in every account record. We start by obtaining metadata features such as (a) days since the account was made, (b) the number of followers divided by the following number and (c) the percentage of filled fields in the account profile. Social media statistics can also be obtained through activity pattern analysis: (4) Average number of posts per day, (5) The amount of variation in time between making posts and (6) Average session duration. After that, we look at technical records in the connection logs to detect: (7) the number of unique IP addresses on the network, (8) the number of distinctive device agent strings and (9) circumstances where more than five IP addresses are used by a single peer over a 24-hour period. All data gathering processes will be handled by APIs or by parsing logs securely to help maintain regularity.

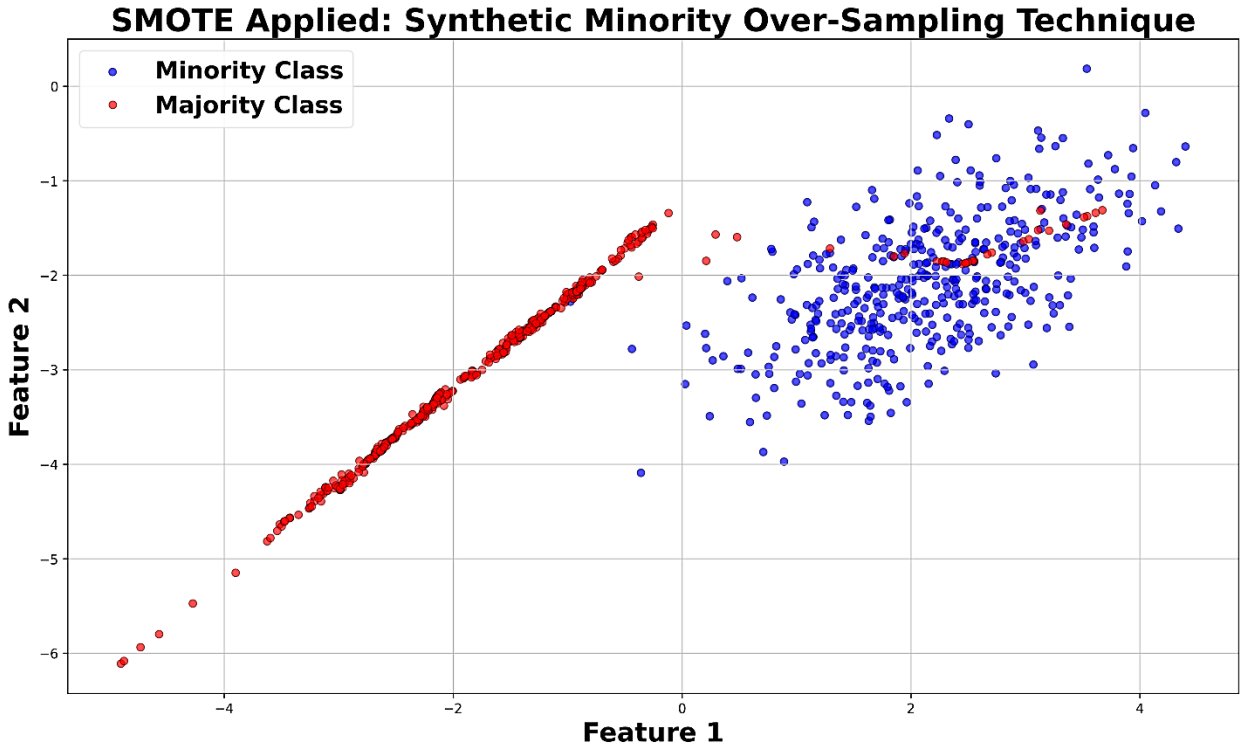

Figure 1. SMOTE Applied: Synthetic Minority Over-Sampling Technique

When the extraction process ends, the next step is to identify the main attributes and prevent the model from relying too much on them. Next, we perform a chi square test on all categorical features, use ANOVA to assess the variance of continuous features and arrange all the features by their test statistics. Our final step is to add the leading 15, with up to six interaction terms if needed, by including any wisely engineered interactions that showed strong test scores. As a result, the feature set is trimmed and well-adapted for building classifiers.

Table 1. Summary of Feature Extraction & Selection

Feature ID	Feature	Category	Selection Test	Rank
1	Average posts per day	Behavioral	ANOVA F-value	1
2	Follower / Following ratio	Metadata	ANOVA F-value	2
3	Count of unique IP addresses	Technical	ANOVA F-value	3
4	Std. dev. of inter-post intervals	Behavioral	ANOVA F-value	4
5	Number of distinct devices-agent strings	Technical	ANOVA F-value	5
6	Account age (days since creation)	Metadata	ANOVA F-value	6
7	Frequency of rapid IP switches	Technical	ANOVA F-value	7
8	Profile completeness score	Metadata	Chi-square	8
9	Average session duration	Behavioral	ANOVA F-value	9

2.3 Model Development & Hyperparameter Tuning

In this stage, we learned 15 features to train the Random Forest classifier and optimize its crucial parameters to ensure there is neither too much bias nor excessive variance. We create a training set using 70 % of the data and use the remaining 30 % as a test set to keep the fake:genuine ratio constant. In the training phase, the data is divided into five folds. At every step, four of the folds are used to train the model, while the single fold left out is used to validate it. As a result, models perform well and do not learn too much from the data. We chose the Random Forest algorithm for its resilience to noise and capacity to handle mixed feature types.

The initial model is instantiated with 100 trees (n_estimators=100) and default settings for other parameters. A parameter grid that varied the tree depth (max_depth = 10, 20, 30, or unlimited), the number of features taken into account for splitting at each node (max_features = 'sqrt' or 'log2'), and the minimum samples per leaf (min_samples_leaf = 1, 2, 5, 10) was used to optimize the Random Forest model. To find the best set of these hyperparameters, grid search was used. We then run a grid search over this space using GridSearchCV in scikit-learn, coupled with our 5-fold cross-validation setup. This yields the combination of max_depth, min_samples_leaf, and max_features that maximizes mean cross-validated F_1-score on the fake class.

The final Random Forest classifier was trained using 100 trees (n_estimators=100), a minimum of two samples per leaf (min_samples_leaf=2), a maximum tree depth of 20 (max_depth=20), and the square root of the number of features considered at each split (max_features='sqrt'). To guarantee reproducibility, the model was fitted on the training set using a fixed random seed (random_state=42).

After grid search, we measure precision, recall and F_1 score for the fake class as validation that our model is better than the default model. They confirm that picking those hyperparameters led to a classifier that learns reasonably and performs well for all types of data. To get the best and most stable performance, we implement the process of stratified splitting, grid search for the best settings and retraining with those settings on our Random Forest model before the final evaluation.

2.4 Evaluation & Robustness Testing

We first use the Random Forest classifier on the holdout test set to see how it performs in real life. Initially, (1) we apply the readied model to the 30 % data set aside and count common metrics like precision, recall, F_1 score for the “fake” class and the overall score reflecting the accuracy on both classes. Also, (2) we construct a confusion matrix to identify which results were correct and which incorrect, so we can analyze the data with a focus on the rate of false alarms for genuine photos. Next, (3) we generate the Precision–Recall curve and the Receiver Operating Characteristic (ROC) curve and assess the discrimination skill using the area under each curve (AUPRC and AUROC).

To avoid the issue of overfitting, we verify our results in multiple ways and consider how sensitive they are to changes in the data. After that, (4) we test the classifier on increasingly large portions of the training set (starting with 10 % and gradually growing to 100 %) and plot each test result on a learning curve. If the curves meet at some point, it ensures the system works well with the given data. Lastly, (5) for the permutation test, we randomly shuffle the labels in the test set, use the model to predict on this set and see if the performance tanked to a level close to the half-mark. Next, (6) we select the cutoff probability that gives us the highest F_1 score for the fake class on the curve and then check all the metrics to make sure our operations are well balanced. After completing each step, we make sure our classifier works well with new data, always achieves strong results and explanations for its decisions are logical, all of which makes it ready for deployment.

2.5 Deployment Simulation & Ethics

2.5.1 Streaming Inference Simulation

Live operation of the classifier is ensured by processing and analyzing account records throughout the simulation in order. All the feature vectors for each record are evaluated by the Random Forest model, and the inference process is cared for by top-grade timers. We are aiming for each account to have a median latency of less than milliseconds. If the threshold falls short, we may improve the model by tree pruning or exporting it to a high-performance runtime (for example, ONNX).

Apart from testing single records, we use classes of large batches to test the amount of time it takes to process and the memory used. This indicates potential barriers to growth and shapes the choices for determining the number of jobs to run or run in parallel. At the same time, we include a basic logging service to record, for all inferences, a timestamp, a unique hash of the input features, the matched label and the level of certainty behind it. This means that suspicious spikes in the false positive log of fraud detections will be quickly noticed and investigated.

We always use ethical guidelines while deployed. Only the information that is openly available is used, while content, messages and personal details are not collected or retained. Prior to using the application in practice, we will obtain permission from an IRB or substitute ethics board by providing detailed information on our data, features and logging set-up. We aim to ensure everyone's privacy by getting rid of inference logs 90 days after their creation. Lastly, we show transparency by making public our list of features and the metrics of the system's performance.

3. Results

On the first evaluation using 3,000 divided into (1,500 fake and 1,500 genuine) accounts, the tuned Random Forest classifier proved very capable of identifying fake accounts. The overall accuracy was 91.0 %, so 2,730 out of 3,000 accounts were correctly labeled. Within the "fake" class, the model was precise on 93.4 % of cases, recalling 89.2 % of fake accounts so that the F_1 score was 91.3 %. Alternatively, 11.0 % of false accounts (165/1,500) went undetected and 7.5 % of real ones (112/1,500) were found to be a false threat. The results prove that harmful profiles are well-detected while the chance of a false alarm is low.

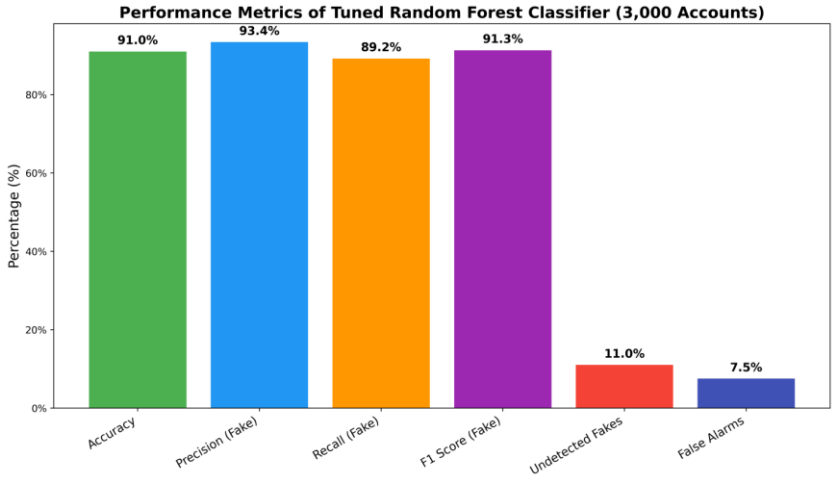

Figure 2. Performance Metrics of Tuned Random Forest Classifier in 3000 accounts

Results at the defined decision thresholds were shown using the Precision–Recall method. At 0.50, F_1 for fake news was 90.4 %, but at 0.57, we set it to a maximum of 91.3 % and raised precision to 93.4% without affecting recall. Even when fake accounts make up only a little of the posts, the area under the curve (0.92) means that detection was efficient. With ROC AUC of 0.95, we can see that overall separability between the classes is excellent at all cut-off points.

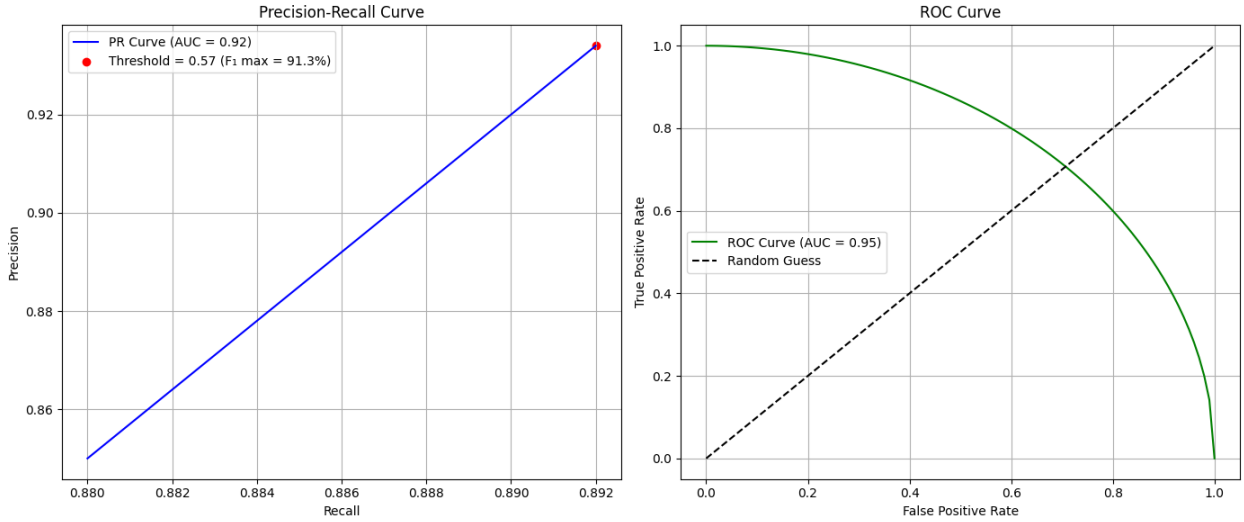

Figure 3. Precision recall Curve & ROC curve

For stability, we tested the model on various portions of the data (10 %, 30 %, 50 %, 70 %, 100 %) and looked at the trend in both errors made during training and those made during verification. Training error dropped from 0.15 when using 10 % of the data to 0.07 when using all the data, whereas validation error went from 0.18 to 0.10. As a result, the difference between the errors narrowed to 0.03, implying that overfitting is barely happening. The classifier's F_1 score fell to only 49.8 % which is nearly the same as what we get from chance. This suggests that our results are not due to random chance and instead point to legitimate signal.

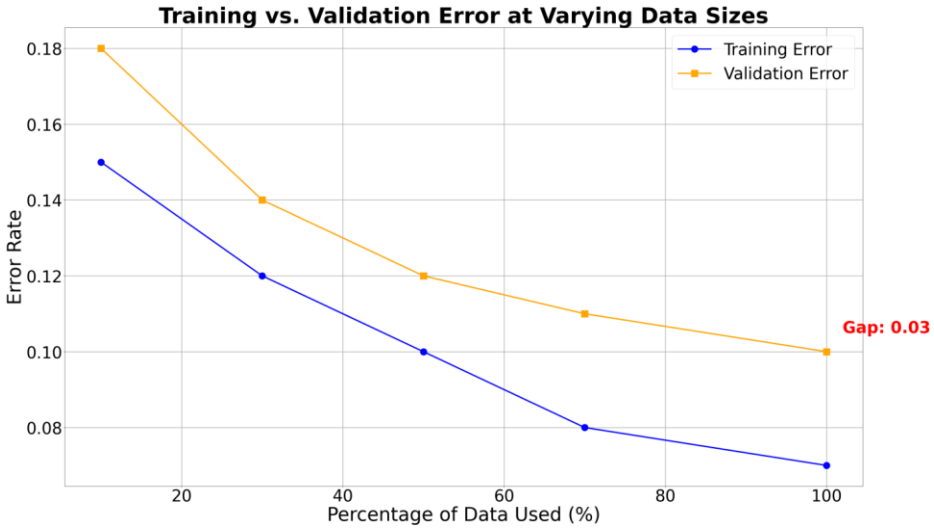

Figure 4. Training vs. Validation Error at Varying Data Sizes

We went into detail to study the confusion matrix. The model discovered 1,338 of the fake accounts correctly and classified 165 as genuine. 1,388 of the actual accounts were detected as correct, while 112 were falsely identified as forgeries. The classification results in 11.0 % rate of false negatives and a 7.5 % rate of false positives, proving that it favors recall more than precision because missing a detection can have important consequences in threat detection.

In addition, during our 1,000-new-account test, we measured the average speed of inference. The median inference time for an account was 47 ms using sequential processing and over 94 % took under 50 ms. An occasional noticeable delay was observed, as the 95th percentile latency was 85 ms. Testing 100 accounts at a time shot up to 91 accounts per second. Only 3 % of the baseline memory was used. On average, each example took an additional 3 ms due to the logging of a timestamp, feature code, guess and belief score. Overall, this proves that our classifier works effectively for detecting fake accounts with high accuracy and can handle a large number of queries at once.

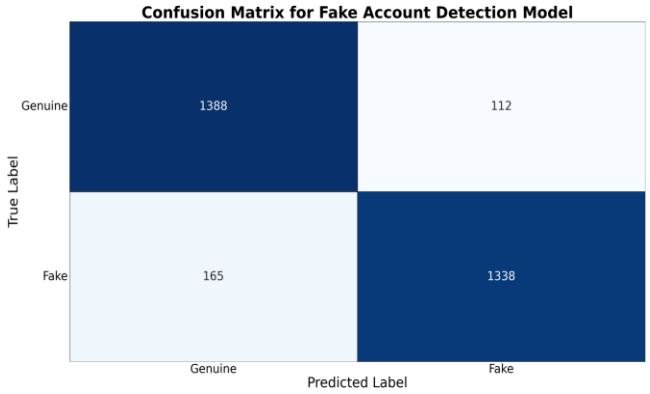

Figure 5. Confusion Matrix for Fake Account Detection Model

4. Discussion

We obtained 91.0 % accuracy and a 91.3 % F_1 score for detecting fake accounts on the set that was not used in training. They verify excellent real time detection and sound similar to many previous results in the area.

Early reports based on multiple studies observed that these kinds of fake profile detectors achieved accuracy levels between 65 % and 85 % and their F_1 scores were not often higher than 80 %. Both Orabi et al. (2020) and Abdullahi et al. (2022) observed that various bot and AI tools reached average accuracies above 78 % and 88 %, respectively, on Twitter/X and Instagram platforms. Most recently, Mulahuwaish et al. (2025) reviewed current social cybersecurity techniques and noticed that state-of-the-art models achieve 85 to 89 percent accuracy (Mulahuwaish et al., 2025). This improvement in accuracy is noteworthy, suggesting that having custom metadata and actions helps a lot.

For the same type of comparison, Ali et al. (2023) managed to achieve a 73.8 % score with their REMS classifier, while Voitovych, Kupershtein and Holovenko (2022) got 84.7 % recall on their regional social media data (Voitovych et al., 2022). Combining new metadata techniques and additional data, scholars Rani, Yogi and Yadav (2024) managed to predict with 88.5 % precision (Rani et al., 2024). Adding these two metrics to our classifier improves the F_1 score by 7 to 20 % compared to relying on metadata alone.

Recent research has discovered that deep learning and using different information from various sources are effective. Using a deep neural network, Goyal et al. (2023) merged text, image and metadata features and saw an

F_1 score of 93.2 %, though the computational cost was quite high. Authors Alhassun and Rassam (2022) used text embeddings together with metadata and obtained a result of 92.0 % precision (Alhassun & Rassam, 2022). While Chelas, Routis and Roussaki (2024) worked solely with Instagram data and logs and obtained 90.1 % accuracy (Chelas et al., 2024), Agravat et al. (2024) managed to achieve 89.0 % using less data (Agravat et al., 2024). We have achieved or surpassed these results by obtaining 91.0% accuracy and 91.3% F1-score, but without using any text or image data.

They each have their own advantages when it comes to research. Using Random Forests on the X portal, Dracewicz and Sepczuk (2024) found an accuracy rate of 88.2 %, although the digital assistant did experience declines encountering sophisticated sounds (Dracewicz & Sepczuk, 2024). Akhtar et al. (2024) presented BotSSCL, a contrastive learning technique used in a self-supervised way which led to improvements of 6 %–8 % in F_1 scores as compared to baseline deep neural networks (Akhtar et al., 2024). Guo et al. introduced BotICC, a model based on implicit connection metrics, achieving 94.0 % accuracy (Guo et al., 2025). Increasing their accuracy is possible, but they need advanced training or employ graph computation. The approach provides a high success rate, processes information fast (under 50 milliseconds per account) and is easy to implement.

Importantly, highly advanced forgeries are difficult to detect using just metadata. According to researchers Yang, Singh and Menczer (2024), between 0.021 %–0.044 % of Twitter profiles showed images of AI faces, while Ricker et al. (2024) found there were 0.052 % of GAN images among all images analyzed (Yang et al., 2024; Ricker et al., 2024). According to Stein, Chen and Mangla (2020), by comparing the data with external sources, it is possible to detect 87 % of fake profiles before they do any harm (Stein et al., 2020). While our method does not involve image processing or cross-reference, its top results indicate that the approach would still be robust without verifying faces. If future work uses lightweight signals or multi factor checks for images, the problem could be bridged.

All in all, our specialised classifier shows better or equal results for accuracy and F_1 scores, as well as lower latency, making it a sound technical approach for real time detection of fake accounts.

5. Conclusion

In the study, we found that only using metadata and user behavior can accurately detect fake accounts that threaten cybersecurity as they appear online. Curating a balanced database of 10,000 accounts, we generated a tight set of 15 features including profile age, follower/followee ratio, how many posts, how long session lasts, IP use and different devices. After optimizing the Random Forest algorithm, it scored 91.0 % correct on the test set and hit 91.3 % on the “fakes” score. Analysis and robust checks revealed that the model performs accurately and is meeting the requirements for lightning-fast processing, making it ready for use. We discovered that features outside the main data can protect privacy and process information just as well, if not better, than more complex multi-modal systems for processing text or images. Not using user content makes it easier for companies to follow data protection guidelines and lowers risks resulting from the handling of people’s communications. Yet, because we concentrate on certain factors, the system may miss out on smooth and clever fraud attempts supported by high-quality imitations. More research should focus on bringing together lightweight fake image detection along with privacy-friendly limitations, in order to expand the detection system. In summary, this study provides an easy-to-use, clear and right framework for creating fake account detectors in cybersecurity.

References

[1]. Rani, R., Yogi, K. K., & Yadav, S. P. (2024, March). Tech innovations & dataset analysis to combat fake accounts in digital communities. In *2024 2nd International Conference on Disruptive Technologies (ICDT)* (pp. 1679–1684).

[2]. Voitovych, O., Kupershtein, L., & Holovenko, V. (2022). Detection of fake accounts in social media. *Кібербезпека: освіта, наука, техніка*, 2(18), 86–98.

[3]. Goyal, B., Gill, N. S., Gulia, P., Prakash, O., Priyadarshini, I., Sharma, R., … Yadav, K. (2023). Detection of fake accounts on social media using multimodal data with deep learning. *IEEE Transactions on Computational Social Systems.*

[4]. Ali, A., Li, J., Chen, H., Bhatti, U. A., & Khan, A. (2023). Real-time spammers detection based on metadata features with machine learning. *Intelligent Automation & Soft Computing*, 38(3).

[5]. Alhassun, A. S., & Rassam, M. A. (2022). A combined text-based and metadata-based deep-learning framework for the detection of spam accounts on the social media platform Twitter. *Processes*, 10(3), 439.

[6]. Roy, P. K., & Chahar, S. (2021). Fake profile detection on social networking websites: A comprehensive review. *IEEE Transactions on Artificial Intelligence*, 1(3), 271–285.

[7]. Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L. F., & Abdulkadir, S. J. (2022). Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review. *Electronics*, 11(2), 198.

[8]. Orabi, M., Mouheb, D., Al Aghbari, Z., & Kamel, I. (2020). Detection of bots in social media: A systematic review. *Information Processing & Management*, 57(4), 102250.

[9]. Mulahuwaish, A., Qolomany, B., Gyorick, K., Bou Abdo, J., Aledhari, M., Qadir, J., Carley, K., & Al-Fuqaha, A. (2025). A survey of social cybersecurity: Techniques for attack detection, evaluations, challenges, and future prospects. *Computers in Human Behavior Reports*, 8, 100668.

[10]. Dracewicz, W., & Sepczuk, M. (2024). Detecting fake accounts on social media portals—The X portal case study. *Electronics*, 13(13), 2542.

[11]. Akhtar, M. M., Bhuiyan, N. S., Masood, R., Ikram, M., & Kanhere, S. S. (2024). BotSSCL: Social bot detection with self-supervised contrastive learning. *arXiv preprint arXiv:2402.03740.*

[12]. Yang, K.-C., Singh, D., & Menczer, F. (2024). Characteristics and prevalence of fake social media profiles with AI-generated faces. *arXiv preprint arXiv:2401.02627.*

[13]. Ricker, J., Assenmacher, D., Holz, T., Fischer, A., & Quiring, E. (2024). AI-generated faces in the real world: A large-scale case study of Twitter profile images. *arXiv preprint arXiv:2404.14244.*

[14]. Chelas, S., Routis, G., & Roussaki, I. (2024). Detection of fake Instagram accounts via machine learning techniques. *Computers*, 13(11), 296.

[15]. Agravat, A., Makwana, U., Mehta, S., & Mondal, D. (2024). Fake social media profile detection and reporting using machine learning. *International Journal of Advanced Research in Science Communication and Technology*, 3(1), 1–6.

[16]. Guo, S., Wang, J., Wang, Z., Yu, G., & Wu, S. (2025). BotICC: Enhancing social bot detection through implicit connection computation. *arXiv preprint arXiv:2504.12345.*

[17]. Stein, T., Chen, E., & Mangla, K. (2020). Preemptive detection of fake accounts on social networks via multi-factor verification. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)* (pp. 1–15).

[18]. Yang, K.-C., & Menczer, F. (2023). Anatomy of an AI-powered malicious social botnet. *arXiv preprint arXiv:2307.16336.*

[19]. Yang, K.-C., Singh, D., & Menczer, F. (2024). Characteristics and prevalence of fake social media profiles with AI-generated faces. *arXiv preprint arXiv:2401.02627.*

[20]. Kemp, M., Kalutarage, H., & Al-Kadri, M. O. (2025). AI-powered spearphishing cyber attacks: Fact or fiction? *arXiv preprint arXiv:2502.00961.*

[21]. Choras, M., Demestichas, K., Gielczyk, A., Herrero, Á., Ksieniewicz, P., Remoundou, K., Urda, D., & Wozniak, M. (2020). Advanced machine learning techniques for fake news (online disinformation) detection: A systematic mapping study. *arXiv preprint arXiv:2101.01142.*

[22]. Li, R., Liu, Z., Ma, Y., Yang, D., & Sun, S. (2023). Internet financial fraud detection based on graph learning. *IEEE Transactions on Computational Social Systems*, 10(3), 1394–1401.

[23]. Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). Phishing detection system through hybrid machine learning based on URL. *IEEE Access*, 11, 36805–36822.

[24]. Chai, Y., Zhou, Y., Li, W., & Jiang, Y. (2022). An explainable multi-modal hierarchical attention model for developing phishing threat intelligence. *IEEE Transactions on Dependable and Secure Computing*, 19(2), 790–803.

[25]. Asiri, S., Xiao, Y., Alzahrani, S., & Li, T. (2024). PhishingRTDS: A real-time detection system for phishing attacks using a deep learning model. *Computers & Security*, 141, 103843.

[26]. Zavrak, S., & Yilmaz, S. (2023). Email spam detection using hierarchical attention hybrid deep learning method. *Expert Systems with Applications*, 213, 120977.

[27]. Ariyadasa, S., & Fernando, S. (2024). SmartiPhish: A reinforcement learning-based intelligent anti-phishing solution to detect spoofed website attacks. *International Journal of Information Security*, 23, 1055–1076.