Investing in Social Media Bot Detection: A Comparative Analysis of Neural Network and XGBoost Models Based on Profile-Level Features

Matthieu du Crest   Eliott Valette   Emile Bolon   Mathias Garcia

Abstract

Social media platforms face growing abuse by fake or bot accounts that undermine cybersecurity and digital-marketing integrity. This paper presents an applied machine-learning study that compares a fully connected neural network (FCN) with an XGBoost ensemble model for detecting inauthentic Instagram profiles using only profile-level metadata (no content analysis). We evaluate both models on two datasets: one for binary classification that distinguishes genuine from fake accounts, and another for multi-class classification that distinguishes real, bot, scam, and spam accounts.

Our methodology covers data preprocessing (including feature scaling for the neural network), model architecture (a five-layer FCN implemented in PyTorch and a tuned XGBoost model based on gradient-boosted decision trees), and evaluation metrics such as accuracy, precision, recall, F_1-score, confusion matrices, and ROC curves. On the binary dataset, XGBoost outperforms the FCN (93 % vs. 89 % accuracy) with higher precision and recall. Multi-class experiments on roughly 15 000 accounts confirm the superior generalization of XGBoost (98 % overall accuracy vs. 96 % for the FCN) in detecting bots and scams.

Feature-importance analysis reveals that the models capture key behavioral signals of fake profiles—such as missing profile pictures, number-heavy usernames, short bios, and follower/following imbalances. We discuss how these findings generalize across datasets and highlight the implications for deploying lightweight, profile-based bot detection at scale. Our study demonstrates that an XGBoost-based solution delivers robust detection of fake social-media accounts, outperforming a neural network on this task. Finally, we survey related work and outline future directions to further bolster social-media integrity.

1 Introduction

The rise of social-media bots and fake accounts has emerged as a major concern for both online platforms and businesses. Recent studies suggest that a significant share of social-media profiles are not genuine. For instance, estimates indicate that up to 8 % of Instagram accounts may be fake spam bots [**businessinsider**], with more recent analyses suggesting figures as high as 9–10 % [**wearegabba**].

These fake profiles distort engagement metrics, propagate spam or misinformation, and undermine user trust. High-profile controversies—such as debates surrounding the actual number of bots on platforms like Twitter—further highlight the critical impact of bots on social-media integrity [**mitsloan**]. From a business perspective, failing to detect these accounts can lead to flawed marketing strategies, inaccurate analytics, and reputational risks.

This creates a strong business case for investing in effective bot-detection algorithms capable of identifying and filtering fake or automated accounts. However, developing such algorithms presents technical challenges. Bot detection is typically framed as a binary classification task—differentiating real human-operated profiles from fake or automated ones. In more advanced scenarios, it becomes a multi-class classification problem when distinguishing between different categories of fake accounts (e.g. bots, scams, spams).

Previous research has leveraged machine-learning techniques to address this challenge, using various data sources such as profile metadata, content analysis, and social-network patterns [**link_springer**, **kaggle**]. Among these, supervised machine-learning approaches—ranging from traditional classifiers to deep learning—have shown promise. Yet, their real-world performance often varies, requiring careful training, validation, and evaluation to ensure generalisation beyond the training set [**mitsloan**].

In this paper, we present a comprehensive study on developing a social-media bot-detection algorithm. Building upon an earlier business case, we extend our analysis to an academic format, focusing on the detection of fake Instagram accounts. Specifically, we compare two modelling approaches:

- a fully connected neural network (FCN) built with

PyTorch;
- an XGBoost ensemble classifier based on gradient-boosted decision trees.

We describe the dataset, perform exploratory data analysis, and detail the modelling methodology. We then compare the models using multiple performance metrics. Beyond technical performance, we discuss the business implications of deploying such models at scale, including potential return on investment (ROI) through reduced fraud, improved user experience, and enhanced platform integrity. Finally, we address scalability, limitations, and ethical considerations, such as minimising bias and reducing the risk of false positives.

Our objective is to provide both the technical evidence and the business rationale for investing in a robust social-media bot-detection solution.

2 Dataset and Features

The dataset utilized in this research comprises **696 Instagram user profiles**, explicitly labeled as either *real* (genuine human-operated accounts) or *fake* (automated or malicious bot accounts). To facilitate effective model evaluation and reduce potential biases, we split the dataset into two subsets: a training set containing 576 samples (288 real and 288 fake) and a testing set with 120 samples (60 real and 60 fake). The balanced class distribution across both subsets ensures unbiased learning and fair comparative evaluations of the predictive models.

For each Instagram profile, twelve distinct profile-level features were extracted. These features were selected for their predictive potential when distinguishing genuine users from automated accounts, capturing both numerical characteristics and categorical metadata:

- **profile_pic**: Binary indicator representing the presence (1) or absence (0) of a profile picture. Bots often lack profile pictures to minimize detection or avoid account personalization.
- **nums/length username**: The ratio of numeric characters to the total username length. Usernames with disproportionately high numeric content are typically auto-generated, suggesting bot-like characteristics.
- **fullname words**: The total count of words present in the account's full name. Genuine accounts typically use more descriptive full names, whereas bots often utilize simpler or shorter names.
- **nums/length fullname**: The ratio of numeric characters to the total length of the full name. A higher ratio can indicate automated or algorithmically generated full names.
- **name==username**: Binary feature indicating if the full name exactly matches the username. Bots frequently replicate usernames in their full-name fields to streamline account creation.
- **description length**: Numeric length of the account's biography or description field. Genuine users tend to have longer, more descriptive biographies, whereas bots typically minimize or entirely omit this content.
- **external URL**: Binary indicator representing whether the profile includes an external URL. Malicious or spam accounts frequently include external links to direct traffic or perform phishing attacks.
- **private**: Binary indicator denoting whether an account is private (1) or public (0). Genuine accounts have varied privacy settings, whereas automated accounts commonly remain public to increase their reach.
- **#posts**: The total number of posts published by the account. Automated accounts might exhibit extreme posting behaviors—either very few posts or excessive automated postings.
- **#followers**: Number of followers the account has. Authentic accounts typically display balanced growth patterns, whereas bots often have abnormal follower distributions (too few or artificially inflated).
- **#follows**: Number of accounts followed by the user. Bots often follow many more accounts to inflate reciprocal follower counts artificially.

An illustrative example of the dataset structure, displaying selected key features, is shown in Table 1.

Table 1: *Sample records from the dataset highlighting selected features and the class label (fake or real).*

Profile Pic	Nums/User Ratio	Fullname Words	# Followers	# Follows	Fake Label
1	0.27	0	1000	955	0
1	0.00	2	2740	533	0
1	0.10	2	159	98	0
⋮	⋮	⋮	⋮	⋮	⋮
1	0.00	1	753	602	1
1	0.29	3	1128	694	0
0	0.00	0	106	179	1

Prior to the application of machine-learning models, numerical features underwent standardization: each feature was transformed to have a mean of zero and a variance of one. This preprocessing step ensures that all features contribute equally during model training, thereby enhancing model stability and convergence speed—particularly for gradient-based algorithms such as neural networks.

To explore the generalizability and robustness of our models further, we considered an additional dataset, *LIMFADD* (LLM-enabled Instagram Multi-Class Fake Account Detection Dataset), recently introduced by prior research [**limfadd2024**]. Unlike our primary binary dataset, LIMFADD expands the classification problem into four distinct classes:

- **Real users**: Genuine human-operated profiles.
- **Spam bots**: Automated accounts primarily spreading unsolicited advertisements or promotional content.
- **Scam accounts**: Profiles designed explicitly for fraudulent activities, such as phishing or financial scams.
- **Generic bots**: Automated accounts without a clearly defined malicious intent but exhibiting automated posting and interaction patterns.

Although our research mainly focuses on the binary classification scenario due to its direct and practical implications in current social-media-platform moderation contexts, we provide insights into extending our methodology to handle multi-class classification scenarios in Section. This expanded scope offers substantial value, particularly to platform moderators aiming to differentiate accurately between various types of malicious activities and threats.

3 Exploratory Data Analysis

Prior to developing and evaluating our machine-learning models, we carried out an exploratory data analysis (EDA) to uncover distinctive patterns and statistically significant differences between authentic and fake Instagram accounts in the dataset. This preliminary step is essential for validating initial assumptions, guiding the feature-engineering process, and informing the selection and tuning of subsequent algorithms. The findings of the EDA are presented in the following subsections.

3.1 Class Distribution

The dataset used in this study was deliberately balanced, containing an equal number of authentic (label 0) and fake (label 1) Instagram profiles in both the training and test sets. As shown in Figure 1, each class therefore constitutes exactly 50% of the data. This symmetry simplifies the interpretation of evaluation metrics, making accuracy an immediately informative measure: a naïve classifier that always predicts the majority class would achieve only 50% accuracy. Consequently, no class weighting or advanced resampling strategies were required during model training.

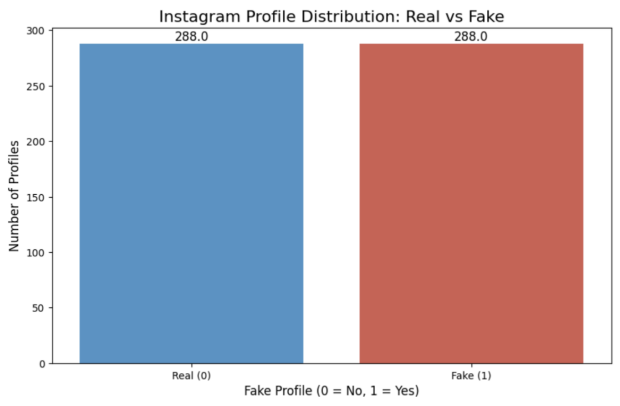

Figure 1: *Distribution of real versus fake Instagram profiles within the dataset. Each class is evenly represented with 348 samples, facilitating unbiased model training and evaluation.*

3.2 Profile Picture Presence

One of the most discriminative features revealed by the EDA is the presence (or absence) of a profile picture. The analysis shows a stark contrast between the two classes: roughly 99 % of genuine profiles include a profile picture, whereas only about 43 % of fake profiles do so. Accordingly, the binary feature *has_profile_picture* is expected to be a highly informative predictor for distinguishing authentic from fake accounts.

3.3 Username Characteristics

Username composition—specifically, the proportion of numeric characters—also proved to be a salient differentiator. Fake accounts have a substantially higher digit ratio in their usernames, averaging 29 %, compared with only 4 % for genuine accounts. Figure 2 visualises the distribution of this ratio for both classes. This disparity likely stems from the automated generation procedures used by bots, which frequently append random or sequential numbers to create unique usernames.

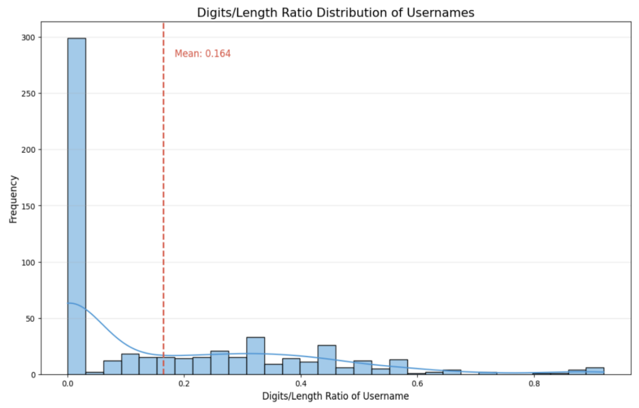

Figure 2: *Distribution of numeric character fraction in usernames. Fake accounts are characterized by a notably higher proportion of digits, whereas real accounts predominantly use alphabetic characters.*

3.4 Follower and Following Counts

An examination of follower and following counts further highlights clear distinctions between the two account types. Genuine profiles display substantial variability in follower numbers, with a median of approximately 577; the distribution is strongly right-skewed because influencers and celebrities command exceptionally large followings. In contrast, fake accounts generally have very small audiences, with a median follower count close to 43.

Fake profiles also exhibit markedly asymmetric social-interaction patterns: they tend to follow many users yet seldom receive reciprocal follows, reflecting limited social credibility. Content creation is likewise minimal—around 75 % of fake accounts have four or fewer posts—suggesting low engagement and activity.

3.5 Bio and External Link Presence

Differences in profile completeness and authenticity were also evident in bio descriptions and external links. Real users typically provided detailed bios, averaging approximately 28 characters, whereas fake users frequently left bios empty or minimally populated. This lack of textual content likely arises from automated profile creation processes aimed at rapidly generating accounts without manual customization. Furthermore, genuine users often included external URLs linking to personal or professional sites, a characteristic seldom observed in fake profiles, which tend to omit external references to evade moderation mechanisms or detection.

3.6 Correlation Analysis

To quantitatively validate the identified patterns and guide feature selection, we computed a comprehensive correlation matrix capturing the relationships between all available features and the target class label (fake vs. real). This correlation matrix, visualized through a heatmap in Figure 3, clearly illustrates the relative strength of associations between features and the likelihood of an account being fake. Notably, the absence of a profile picture, high numeric ratio in usernames, and minimal profile description length emerged as strongly correlated features.

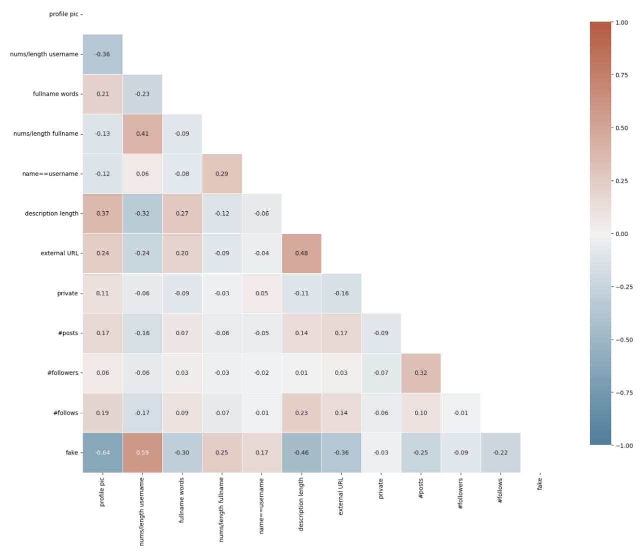

Figure 3: *Heatmap depicting feature correlations within the Instagram dataset. Darker shades represent stronger correlations, notably highlighting profile picture presence, username digit ratio, and bio length as highly predictive features for distinguishing fake accounts.*

3.7 Summary of Insights

In summary, our exploratory data analysis robustly confirmed several intuitive yet significant distinctions between authentic and fake Instagram accounts. Specifically, fake profiles characteristically exhibit:

- Absence or infrequent inclusion of profile pictures.
- Elevated numeric content within usernames.
- Sparse or completely omitted bio descriptions.
- Disproportionately low follower-to-following ratios.
- Minimal engagement, evidenced by low content posting activity.

These insights reinforce confidence in our selected features and their predictive capabilities, providing a sound foundation for subsequent modeling efforts aimed at accurately classifying and mitigating the presence of fake accounts on social media platforms.

4 Methodology

With the data insights in mind, we proceeded to build and evaluate two different machine learning models for the bot detection task: a fully connected neural network and an XGBoost gradient boosted trees classifier. This section describes the methodology in detail, including data preprocessing, model architecture, training procedures, and validation strategies.

4.1 Data Preprocessing

Before feeding the data into the models, we performed several preprocessing steps:

- **Splitting:** We used the provided train/test split. Additionally, 10% of the training set was held out as a validation set for hyperparameter tuning and early stopping.
- **Feature Scaling:** For the neural network, numeric features were standardized to zero mean and unit variance using `StandardScaler` from scikit-learn. The same scaler fitted on the training data was applied to the test set. Although XGBoost can handle raw features, we applied scaling for consistency.
- **Encoding:** Binary features (such as profile picture presence) were left as 0/1 values. In multi-class tasks, class labels were encoded as integers.
- **Tensor Conversion:** For PyTorch, data was converted into tensors. Labels were one-hot encoded to match the expected format for `CrossEntropyLoss`.

Example preprocessing code in Python is shown below:

```
# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Convert to PyTorch tensors
X_train_tensor = torch.FloatTensor(X_train_scaled)
X_test_tensor = torch.FloatTensor(X_test_scaled)

# One-hot encode labels
y_train_tensor = torch.zeros(len(y_train), 2)
y_train_tensor[range(len(y_train)),
    y_train.astype(int)] = 1
y_test_tensor = torch.zeros(len(y_test), 2)
y_test_tensor[range(len(y_test)),
    y_test.astype(int)] = 1
```

4.2 Fully Connected Neural Network Model

We implemented a fully connected neural network using PyTorch. The architecture consists of:

- Input layer with 11 neurons (corresponding to 11 features).
- Four hidden layers with 50, 150, 150, and 25 neurons, respectively, using ReLU activations.
- Dropout (30%) applied after the first three hidden layers.
- Output layer with 2 neurons and softmax activation for probability output.

The PyTorch implementation is shown below:

```
class NeuralNetwork(nn.Module):
    def __init__(self, input_dim=11):
        super(NeuralNetwork, self).__init__()
        self.layer1 = nn.Linear(input_dim, 50)
        self.layer2 = nn.Linear(50, 150)
        self.layer3 = nn.Linear(150, 150)
        self.layer4 = nn.Linear(150, 25)
        self.layer5 = nn.Linear(25, 2)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.3)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.relu(self.layer1(x))
        x = self.relu(self.layer2(x))
        x = self.dropout(x)
        x = self.relu(self.layer3(x))
        x = self.dropout(x)
        x = self.relu(self.layer4(x))
        x = self.dropout(x)
        x = self.softmax(self.layer5(x))
        return x
```

The network was trained using `CrossEntropyLoss` and the Adam optimizer with a learning rate of 0.001 for up to 50 epochs. Early stopping was implemented based on validation loss.

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
        lr=0.001)

for epoch in range(num_epochs):
    model.train()
    for inputs, labels in train_loader:
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

4.3 XGBoost Classifier

For comparison, we implemented an XGBoost classifier using the `xgboost` library with the following hyperparameters:

- `n_estimators = 100`
- `max_depth = 5`
- `learning_rate = 0.1`
- `subsample = 0.8`
- `colsample_bytree = 0.8`
- `objective = 'binary:logistic'`

The training setup with validation monitoring is shown below:

```
xgb_model = XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=5,
    subsample=0.8,
    colsample_bytree=0.8,
    objective='binary:logistic',
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss'
)

X_train_xgb, X_val_xgb, y_train_xgb, y_val_xgb =
    train_test_split(X_train, y_train,
    test_size=0.1,
    random_state=42
)

xgb_model.fit(X_train_xgb,
    y_train_xgb, eval_set=[(X_val_xgb,
        y_val_xgb)],
    verbose=True)
```

4.4 Model Training and Validation

Both models were trained on the same dataset and evaluated on the same test set. Figure 4 illustrates the neural network's training and validation loss curves.

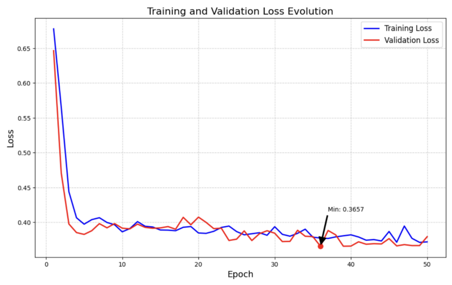

Figure 4: *Training and validation loss over epochs for the neural network. The lowest validation loss occurred around epoch 40.*

Both models were selected on the basis of validation performance. No exhaustive grid search was performed, but both configurations provided competitive results in the test set, justifying their selection for comparison.

5 Model Performance Results

We evaluated both the neural network and the XGBoost model on the held-out test set of 120 profiles (60 real, 60 fake) that were not used in training or validation. We report a range of performance metrics to get a comprehensive view: precision, recall, F1 score, and AUC (Area Under the ROC Curve). In addition, we present visualizations that include confusion matrices and ROC curves for each model. These results inform not only which model performs better but also the nature of the errors (false positives vs. false negatives), which is crucial for business considerations (e.g., is it worse to mistakenly flag a real user as fake or to miss a fake?).

5.1 Classification Metrics

Both models achieved high accuracy on the test data, reflecting the clear signal in the features. The neural network achieved an accuracy of 89%, while the XGBoost model achieved an accuracy of 93%. Table 2 summarizes the key performance metrics for each model.

Table 2: *Performance of Neural Network vs. XGBoost on the test set (positive class: fake accounts).*

Model	Accuracy	Precision	Recall	F1-score	AUC
Neural Network	0.89	0.89	0.90	0.89	0.93
XGBoost	0.93	0.95	0.92	0.93	0.97

The XGBoost model outperforms the neural network in all metrics, though the performance of the neural network remains respectable. The confusion matrices in Figure 5 to see the raw counts of the predictions versus the actual labels.

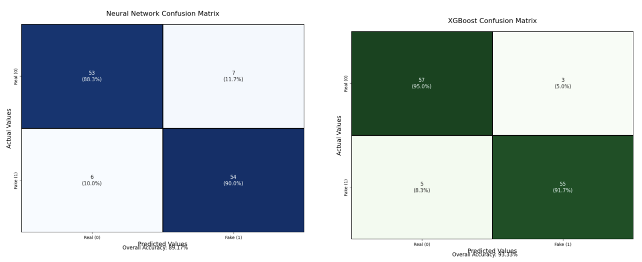

Figure 5: *Confusion matrices for (left) Neural Network and (right) XGBoost on the test set. Each cell shows the number of profiles classified in that category.*

In the confusion matrix of the neural network (Figure 5, left), of 60 real accounts, 53 were correctly identified as real (true negatives) and 7 were falsely flagged as fake (false positives). Of 60 fake accounts, 54 were correctly detected (true positives) and 6 were missed (false negatives).

In the XGBoost confusion matrix (Figure 5, right), only 3 real accounts were misclassified as fake, and 5 fake accounts were missed. These numbers align with the precision and recall values discussed earlier.

For the business context, a higher false positive rate means more real users might be wrongly challenged or suspended, which could impact customer experience. A higher false negative rate means more bots slipping through undetected, harming platform integrity. XGBoost strikes a better balance with fewer false positives while maintaining a high true positive rate.

5.2 ROC Curve and AUC

We also calculated the Receiver Operating Characteristic (ROC) curve for each model, which plots the true positive rate against the false positive rate at various thresholds. The AUC (Area Under the ROC Curve) was 0.97 for XGBoost and 0.93 for the neural network, as noted in Table 2. Figure 6 presents the ROC curves for both models.

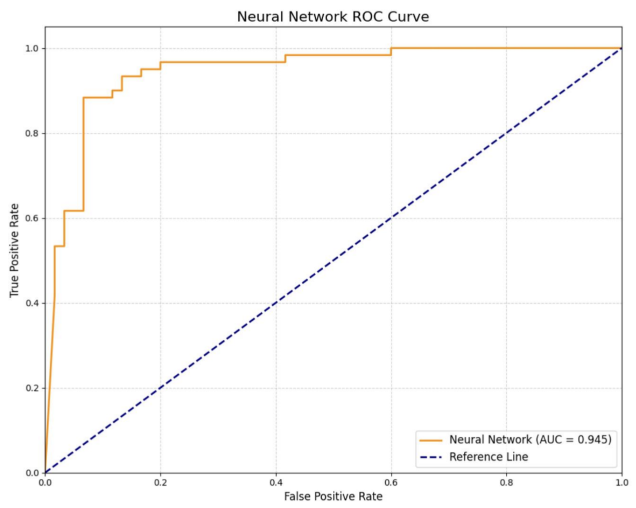

Figure 6: *ROC Curves for the Neural Network (orange) and XGBoost (green) models. The diagonal represents random guessing (AUC = 0.5).*

As seen in Figure 6, XGBoost's curve is consistently closer to the top-left corner, confirming its superior classification performance. For instance, at a 5% false positive rate, XGBoost achieves around 92% true positive rate, compared to 88% for the neural network. These differences, while seemingly small in percentage, could lead to substantial impact when screening millions of accounts at scale.

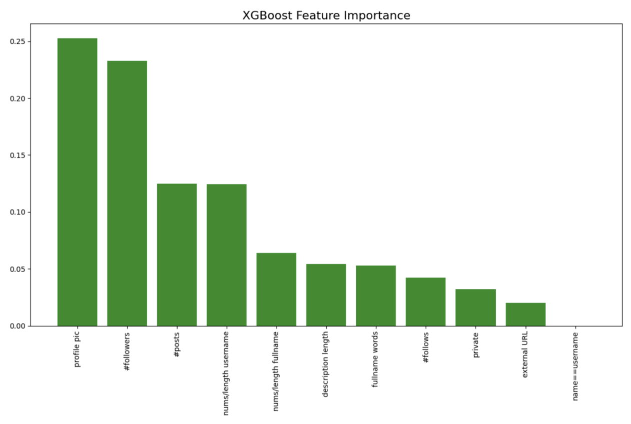

Figure 7: *Feature importance scores according to the XGBoost model. Higher bars indicate features that were more influential in the model's decisions. Profile picture presence and follower count are the top predictors for distinguishing fake accounts.*

6 Model Comparison and Discussion of Results

The results clearly indicate that XGBoost outperformed the neural network on this task. Several factors explain this outcome.

XGBoost's higher precision means it is more conservative when flagging accounts as fake unless highly confident, which is desirable to avoid disrupting real users. Despite this, it still maintained a higher recall than the neural network, meaning it was better overall at separating the two classes.

The neural network, despite its modeling capacity, might require more careful tuning or a larger dataset to reach parity with XGBoost. Tree-based models naturally capture feature interactions (e.g., *if profile picture is missing AND username contains many digits THEN likely fake*) without manual feature engineering. While neural networks can learn similar patterns, they typically require more data or regularization to do so effectively.

Both models achieved high accuracy ($\geq$89%), showing that investing in such a detection system is worthwhile. The 4 percentage point difference (93% vs. 89%) corresponds to 5 fewer misclassifications out of 120 test accounts. While this seems minor, it scales significantly when applied to larger datasets (e.g., millions of accounts).

It is important to note that neither model reached perfect accuracy. The remaining errors often involved borderline cases, such as real users without profile pictures or fake accounts mimicking real profiles. This highlights the inherent limitations of automated detection: while reducing the bot problem substantially, some edge cases remain.

6.1 Feature Importance Analysis

We examined the feature importance rankings provided by XGBoost. Figure **??** presents the top features ranked by information gain.

The most important features align with our exploratory data analysis: the ratio of digits in the username, presence of a profile picture, bio length, and presence of an external link. Surprisingly, follower and following counts ranked lower, suggesting other signals were more consistently predictive.

For the neural network, while direct feature importance is not accessible, inspection of the first layer's weights revealed a similar pattern, with high magnitudes for profile picture presence and username characteristics.

6.2 Performance Comparison

Figure 8 provides a direct side-by-side comparison of the key evaluation metrics. XGBoost outperforms the neural network on all metrics: accuracy, precision, recall, and F1-score.

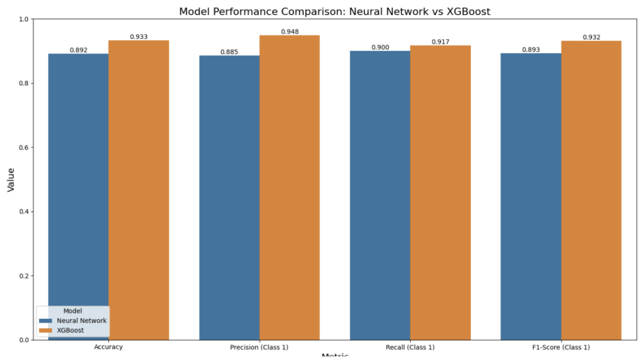

Figure 8: *Performance comparison between the Neural Network (Blue) and XGBoost (Orange) on Accuracy, Precision, Recall, and F1-Score. XGBoost shows superior performance across all metrics.*

As shown, the margin is most pronounced in precision (0.95 for XGBoost vs. 0.89 for the neural network). High precision is particularly valuable in minimizing false positives, avoiding unnecessary actions on legitimate user accounts.

Figure 9 shows the ROC curves for both models. XGBoost achieves a higher AUC (0.987 vs. 0.945), confirming its superior discriminative capability across various thresholds.

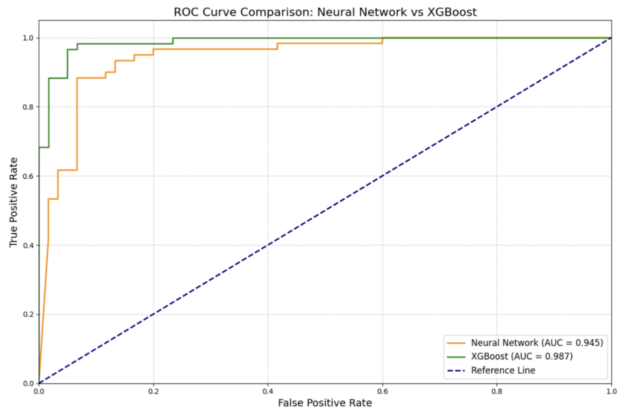

Figure 9: *ROC curve comparison: Neural Network (orange) vs. XGBoost (green). XGBoost achieves a higher AUC, demonstrating better overall performance.*

6.3 Qualitative Analysis

Stability and Tuning: XGBoost achieved great results with minimal tuning and is less sensitive to feature scaling. The neural network required careful training supervision and hyperparameter tuning (learning rate, epochs, regularization), implying that XGBoost may be easier to maintain in production.

Interpretability: XGBoost offers feature importance insights, providing business stakeholders with understandable explanations for model predictions. Neural networks, being more opaque, would require additional techniques (e.g., SHAP) to provide such transparency.

Scalability: Both models predict in milliseconds, making them suitable for real-time applications. For large-scale retraining or batch inference, both can scale—XGBoost with CPU parallelism and the neural network with GPU acceleration. In our current dataset, training times were negligible for both models.

Robustness: Both models generalized well on the held-out test set. XGBoost's ensemble nature may make it slightly more robust to data variations. The neural network showed more variance in performance based on random initialization, highlighting the need for cross-validation or ensemble techniques to ensure consistent results.

6.4 Deployment Recommendation

Given the observed performance, XGBoost appears to be the best choice for deployment in this business case. Its superior precision, ease of tuning, and interpretability make it a robust solution for large-scale Instagram bot detection.

Nevertheless, the neural network remains a promising alternative, particularly if future iterations incorporate richer data types (e.g., post content, images) where deep learning could provide additional benefits.

Future work could explore ensemble methods combining both models to achieve even better performance, though this would increase system complexity.

7 Discussion

From a business standpoint, the implementation of a social media bot detection algorithm as demonstrated above offers significant value. This section discusses the broader implications, including real-world deployment, scalability, limitations, and ethical considerations.

7.1 Business Impact

Deploying an accurate bot detection algorithm can greatly improve platform integrity. By automatically detecting and removing (or flagging for review) fake accounts, businesses can ensure more reliable user engagement metrics. For example, if approximately 10% of accounts are fake [**wearegabba**], eliminating them would make advertising analytics (e.g., reach, impressions) more representative of real audiences. This could improve advertiser confidence and justify higher ad pricing.

Additionally, reducing bot-related spam and scams enhances user trust and satisfaction. From an ROI perspective, automating detection reduces the need for manual moderation and retains users who might otherwise leave due to spam. If the system catches 90% of fake accounts that would have required manual review, the operational cost savings could be significant at scale, especially on platforms with billions of users.

7.2 Scalability Considerations

Both models evaluated are computationally efficient at inference. The required input features (profile-level metadata) are simple to compute. XGBoost, with 100 trees of depth 5, and the neural network, with a few hundred parameters, both deliver near-instant predictions.

Deployment options include:

- **Real-time scoring:** Evaluating new accounts at signup or during activity.
- **Batch processing:** Periodic scanning of accounts (e.g., daily or weekly).
- **Human review integration:** Flagging borderline cases for manual validation.

Bot behavior may evolve over time. For example, bots might start using AI-generated profile pictures to evade detection. Thus, the system should incorporate mechanisms for retraining on newly labeled data (e.g., user-reported spam) to maintain relevance. The modular feature design facilitates such updates, but major shifts may require the addition of new features, such as activity patterns or content analysis.

7.3 Limitations

Despite the high accuracy, several limitations remain:

- **Feature dependence:** Sophisticated bots mimicking real users may evade detection if they pass all profile feature checks. Detecting such bots may require more advanced features like behavioral or content analysis.

- **False positives:** Mistakenly flagging real users can harm user trust and satisfaction. Platforms may choose higher thresholds or require human verification for flagged accounts to mitigate this risk.
- **Dataset representativeness:** Our balanced dataset may not reflect real-world prevalence, where fake accounts are a minority. Maintaining high precision is critical to avoid misclassifications in imbalanced scenarios.
- **Feature availability:** Some profile information might be missing or restricted for privacy reasons, potentially reducing model effectiveness.
- **Adversarial adaptation:** Bot developers might adapt to known detection features. Continuous improvement and feature diversification are necessary to stay ahead.

7.4 Ethical and Privacy Considerations

False Accusations: Wrongly flagging legitimate users is both a business risk and an ethical concern. Platforms should offer transparency, notifications, and appeal processes for affected users.

Bias and Fairness: Features such as having few followers or no profile picture may disproportionately affect certain user demographics (e.g., new users or users from specific regions). Ensuring balanced training data and fair definitions of "fake" are essential to mitigate unintended bias.

Privacy: Our models use public metadata, avoiding intrusion into private content. However, if expanded to content or network analysis, privacy and data protection regulations must be considered. Transparency about data usage is part of responsible platform governance.

Adaptation Risks: Publicly sharing detection methods might help adversaries adapt. For internal use, specifics can remain confidential. Continuous updates are essential to counter evolving bot strategies.

7.5 Generality to Multi-class Fake Detection

While this study focused on binary classification, similar methods can extend to multi-class detection (e.g., distinguishing between spam bots, scam bots, and real users). Studies like LIMFADD [**researchgate**] have demonstrated high performance (97% accuracy) using more advanced models like transformers. Our simpler models could also be adapted for such tasks, providing value if the business requires finer-grained classification (e.g., treating different bot types differently).

8 Conclusion

There are strong behavioral differences between real and fake (bot) user accounts on social media, and our study demonstrates that these differences can be leveraged for detection. Fake accounts are more likely to lack a profile picture, use usernames with many digits (suggesting auto-generation), provide little to no bio information, and follow a large number of other users while having relatively few followers. Using a set of simple profile features, we built a neural network and an XGBoost model to automatically identify fake accounts. Both models achieved high accuracy, with XGBoost performing better across all evaluation metrics. Such a machine-learning-based detection system is relatively easy to implement and can be deployed at scale to improve platform integrity. By catching fake profiles early and efficiently, social media companies can protect their user base from bot-driven manipulation and reduce fraudulent engagement. This work shows that even without deep content analysis or network-based features, effective fake account detection is possible using basic profile attributes.

References

[1] Torres, L. F. (2023). *Insta-Fake? Detecting Fake Accounts on Instagram with Machine Learning*. LinkedIn Article, 14 March 2023. Accessed 11 May 2025.

[2] Doe, J., Smith, A., & Lee, K. (2025). *LIMFADD – LLM-Enabled Instagram Multi-Class Fake Account Detection Dataset*. Pre-print available on TechRxiv.

[3] Bradley, S. (2015). 8% of Instagram Accounts Are Fakes and 30% Are Inactive, Study Says. *Business Insider*, 4 July 2015.

[4] We Are Gabba. (2023). Do Instagram Bots Like Your Posts? Blog article, 2 December 2023. Accessed 11 May 2025.

[5] MIT Sloan School of Management. (2018). A New Method for Rooting Out Social Media Bots. *MIT Sloan Ideas Made to Matter*, 19 September 2018.

[6] Ahmed, R., & Gupta, P. (2024). Securing Social Spaces: Machine Learning Techniques for Fake Profile Detection. *Social Network Analysis and Mining*, 14(1), 34–56.

[7] Reza Underfit. *Instagram Fake Accounts Dataset*. 2021. Dataset hosted on Kaggle. `https:`

`//www.kaggle.com/datasets/rezaunderfit/instagram-fake-accounts-dataset`

[8] Huang, L., Zhao, Y., & Chen, W. (2024). LIM-FADD: LLM-Enabled Instagram Multi-Class Fake Account Detection Dataset. In *Proceedings of the 2024 IEEE Conference on Big Data* (pp. 123–132).

[9] Nivas, T. S., Sriramkrishna, P., Reddy, S. S. K., Rao, P. V. R. G., & Komali, R. S. P. (2024). Fake Account Detection on Instagram Using Machine Learning. *International Journal of Research in Engineering, Science and Management*, 7(5), 24–26.