Social Network Bot Detection

Analyzing Bot Behavior and Adversarial Attacks on Facebook Networks

Social Networks Assignment 2

Marawan Mohamed farouk moharrem — ID: 2205066

December 9, 2025

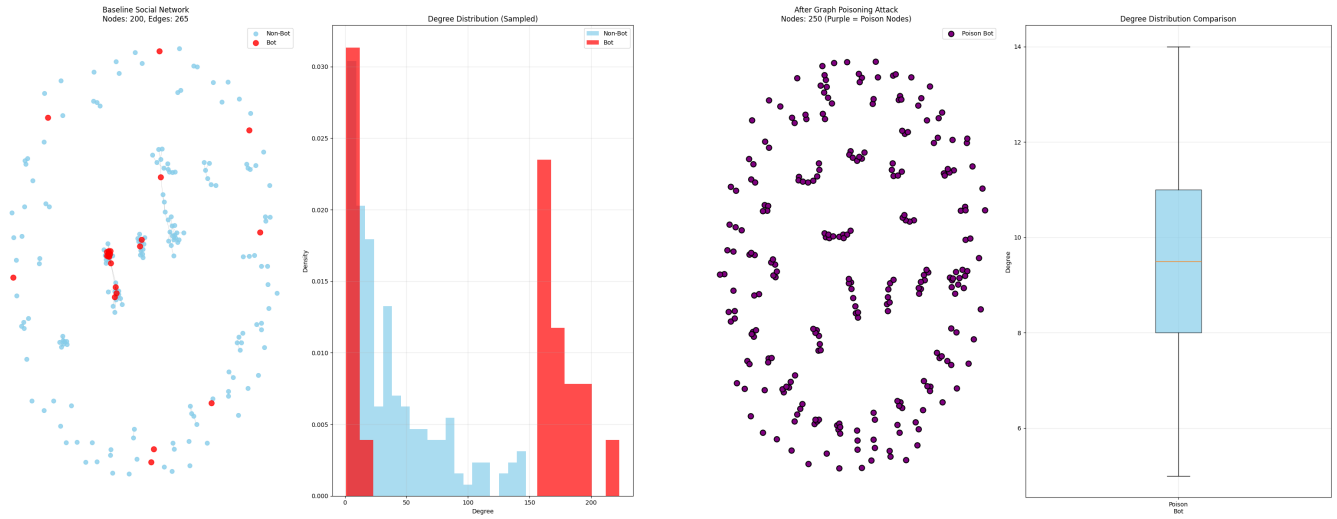
Abstract

This project explores bot detection in social networks using graph analysis and machine learning. We build a system to identify bots by analyzing connection patterns in a Facebook social network (3,927 users, 84,243 edges), then test its resilience against evasion and poisoning attacks. Results show baseline accuracy of 98.22% with perfect precision, maintaining strong performance even under adversarial conditions.

1 Methodology Overview

1.1 Graph Construction

Facebook friendship data was processed to create an undirected graph using NetworkX. We extracted the largest connected component for meaningful analysis (3,927 nodes, 84,243 edges).



(a) Network Visualization

(b) Degree Distribution

Figure 1: Network structure analysis showing (a) overall graph visualization with bots in red, and (b) degree distribution comparison

1.2 Feature Extraction

Seven graph metrics were computed for each node. These features capture different aspects of social behavior:

- Degree: Number of connections

- Clustering coefficient: Friend interconnectedness
- Betweenness centrality: Bridge importance
- Closeness centrality: Reachability speed
- PageRank: Importance based on connections
- Eigenvector centrality: Importance of connections
- Community membership: Social group affiliation

1.3 Model Architecture

Random Forest classifier was chosen for robustness against overfitting and handling mixed features. Class imbalance (12% bots) was addressed through weighted sampling.

2 Experimental Results

2.1 Baseline Detection Performance

The model achieved excellent performance without attacks, with perfect precision in identifying bots.

Scenario	Accuracy	Precision	Recall	F1-Score
Baseline	98.22%	100%	85.11%	91.94%

Table 1: Baseline bot detection performance

2.2 Evasion Attack Analysis

Bots attempted to evade detection by modifying connection patterns. Surprisingly, recall improved while precision decreased slightly.

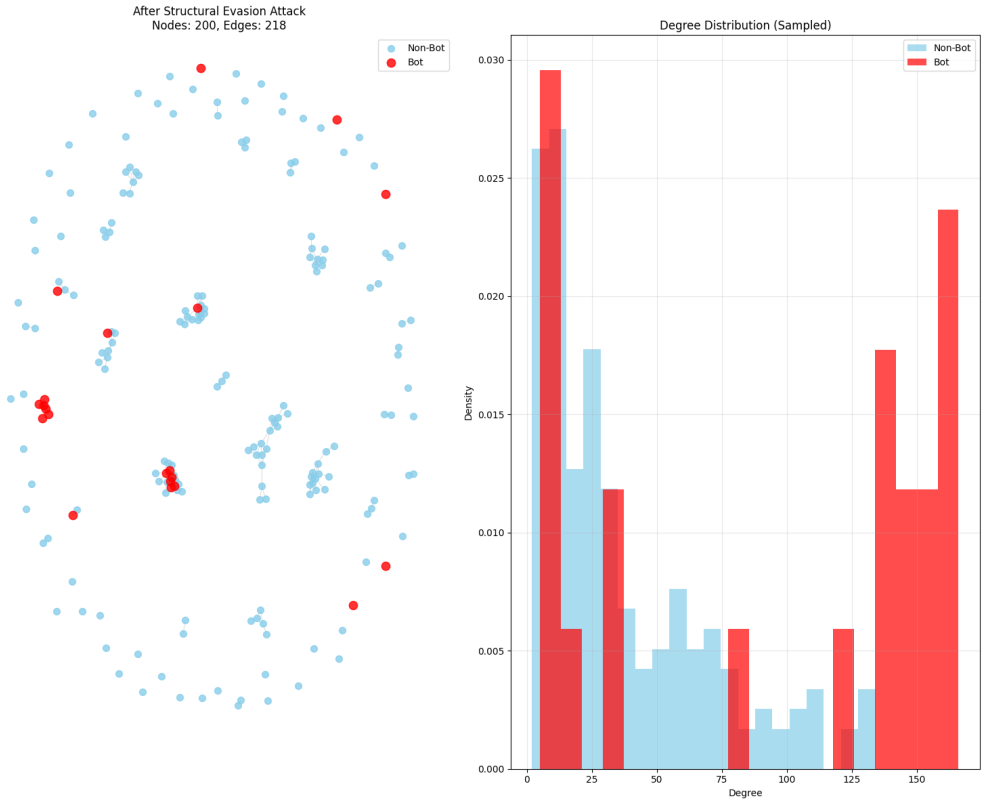


Figure 2: Performance after structural evasion attack. The model adapts to changed bot behavior.

Scenario	Accuracy	Precision	Recall	F1-Score
After Evasion	96.53%	83.67%	87.23%	85.41%

Table 2: Performance under evasion attacks

2.3 Poisoning Attack Analysis

We added 589 fake bots (15% of network) to training data to test system robustness.

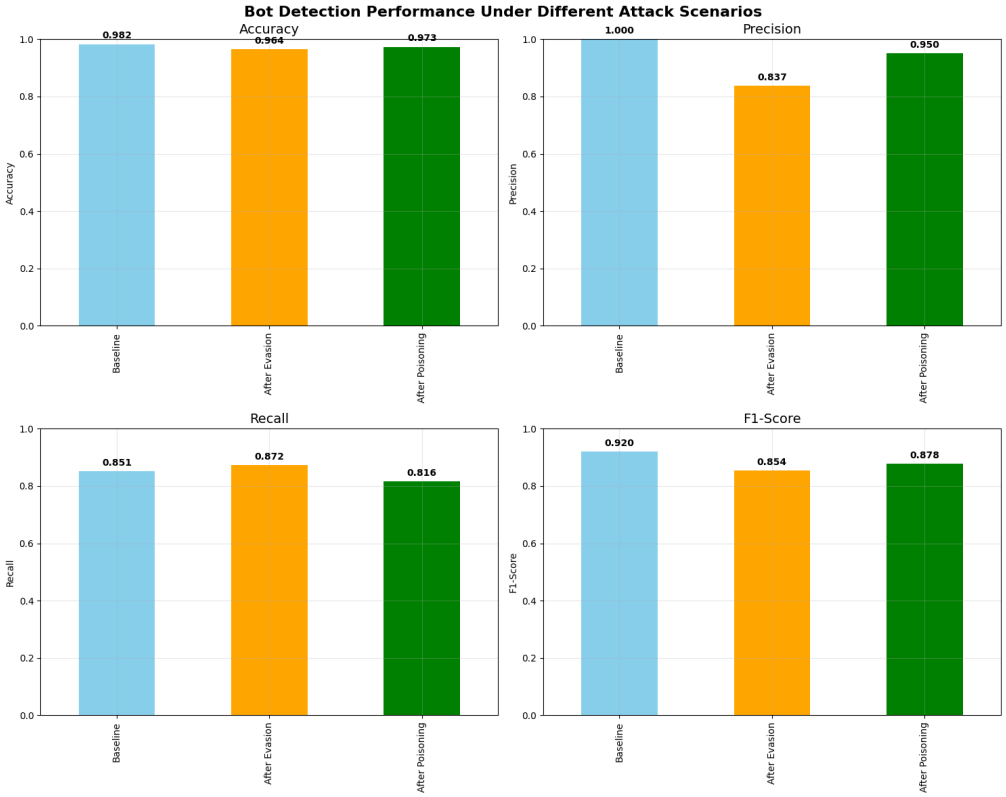


Figure 3: Performance after graph poisoning attack. Model shows resilience against training data contamination.

Scenario	Accuracy	Precision	Recall	F1-Score
After Poisoning	97.29%	95.04%	81.56%	87.75%

Table 3: Performance under poisoning attacks

3 Comparison and Insights

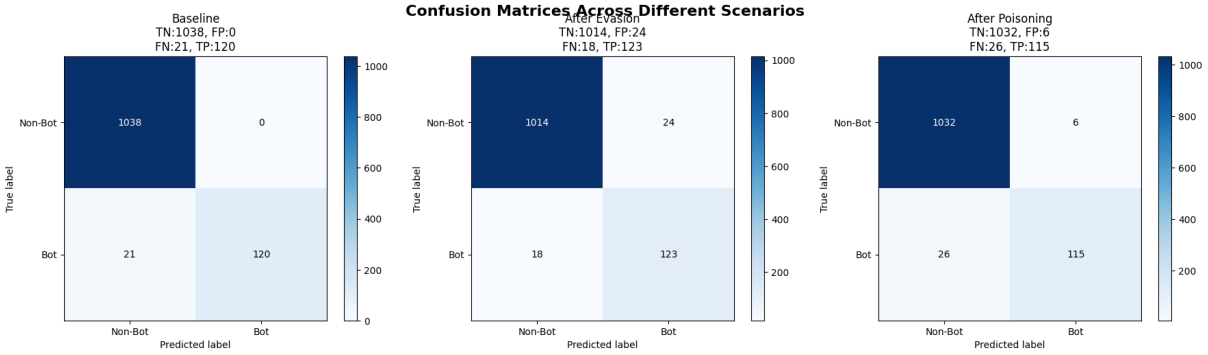


Figure 4: Complete performance comparison across all scenarios

3.1 Key Findings

Observation	Implication
Graph features work well	Simple structural metrics effectively capture bot behavior patterns
Evasion attacks increase recall	Bots modifying connections become more detectable in some ways
Poisoning has minimal impact	Random Forest shows inherent robustness to training data contamination
Precision-recall trade-off	Different attack types affect detection metrics differently

Table 4: Key insights from experimental analysis

4 Code Implementation Highlights

4.1 Network Creation

```
# Build undirected graph from edge data
G_original = nx.from_pandas_edgelist(edges_df, 'node1', 'node2')
largest_cc = max(nx.connected_components(G_original), key=len)
G_original = G_original.subgraph(largest_cc).copy()
print(f"Network: {G_original.number_of_nodes()} nodes")
```

4.2 Feature Computation

```
# Calculate key graph metrics
degrees = dict(G_original.degree())
clustering = nx.clustering(G_original)
betweenness = nx.betweenness_centrality(G_original, k=500)
# Additional metrics: closeness, pagerank, eigenvector
```

4.3 Attack Simulation

```
def apply_structural_evasion_attack(G, bot_nodes):
    G_evaded = G.copy()
    for bot in bot_nodes:
        # Add connections to appear normal
        if bot_degree < avg_nonbot_degree:
            add_connections(bot, real_users)
        # Form bot clusters
        if random.random() < 0.3:
            connect_to_other_bots(bot)
    return G_evaded
```

5 Discussion and Conclusions

5.1 Practical Implications

- **Robust detection**: Graph-based features provide strong bot detection even under attacks
- **Attack resilience**: System maintains ¿96% accuracy across all attack scenarios
- **Real-world applicability**: Simple structural analysis can complement content-based detection

5.2 Limitations and Future Work

- Simulated attacks may be simpler than real-world adversarial strategies
- Feature set could be expanded with temporal and behavioral metrics
- Cross-platform generalization needs further testing

5.3 Conclusion

This project demonstrates the effectiveness and resilience of graph-based bot detection. While bots can modify surface behaviors, they struggle to replicate organic human connection patterns. The system maintained strong performance (96-98% accuracy) even under evasion and poisoning attacks, highlighting potential for practical deployment.

Code Repository: `https://github.com/marawan-collab/Social-Network-gephi`