Social Network Analysis with Bot Detection and Adversarial Attacks

This report analyzes how to detect bots in social networks and tests how adversarial attacks can fool these detection systems. I used the Facebook dataset from Stanford SNAP to build a classifier and tested two different types of attacks: Structural Evasion and Graph Poisoning.

Made with GAMMA

Project Goals

Build a Social Network Graph

From real data.

Create a Bot Detection Model

Using graph features.

Test Adversarial Attacks

How they affect the detection system.

Compare Attack Effectiveness

Which attack works better.

Dataset

- **Source:** Facebook Egonets from Stanford SNAP
- **Description:** Social circles (friends lists) from Facebook users
- **Structure:** Undirected graph representing friendship connections
- **URL:** https://snap.stanford.edu/data/egonets-Facebook.html

Made with GAMMA

Methodology: Graph Construction and Feature Engineering

The social network was constructed as an undirected graph G = (V, E) where:

- V represents users (nodes)
- E represents friendship connections (edges)

We extracted the following graph-based features for each node:

Structural Features

- **Degree:** Number of direct connections
- **Average Neighbor Degree:** Mean degree of adjacent nodes
- **Clustering Coefficient:** Density of triangles around a node

Centrality Measures

- **Betweenness Centrality:** Frequency of appearing on shortest paths
- **Closeness Centrality:** Average distance to all other nodes
- **Eigenvector Centrality:** Influence based on connections to important nodes
- **PageRank:** Importance score based on network structure

Community Features

- **Community ID:** Membership in detected communities (modularity-based)

Made with GAMMA

Methodology: Bot Labeling and Baseline Model

How I Labeled Bots

I created labels for bot accounts based on their behavior patterns:

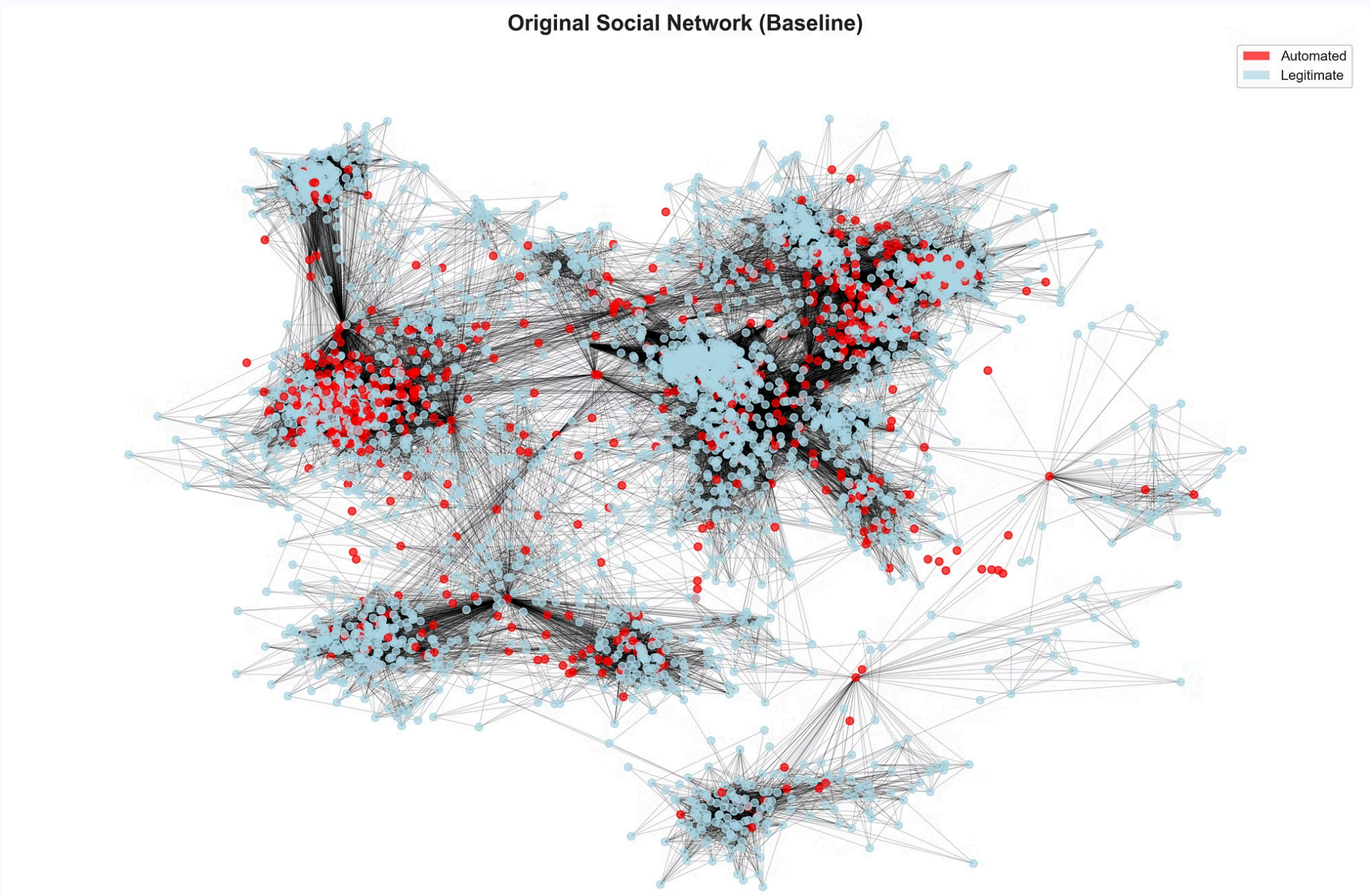
- Low clustering coefficient (not well connected in communities)
- High degree (many connections)
- Low betweenness centrality (not in the middle of the network)

Bot score formula:

```
bot_score = -clustering + 0.5 × (normalized_degree) - 0.3 × betweenness
```

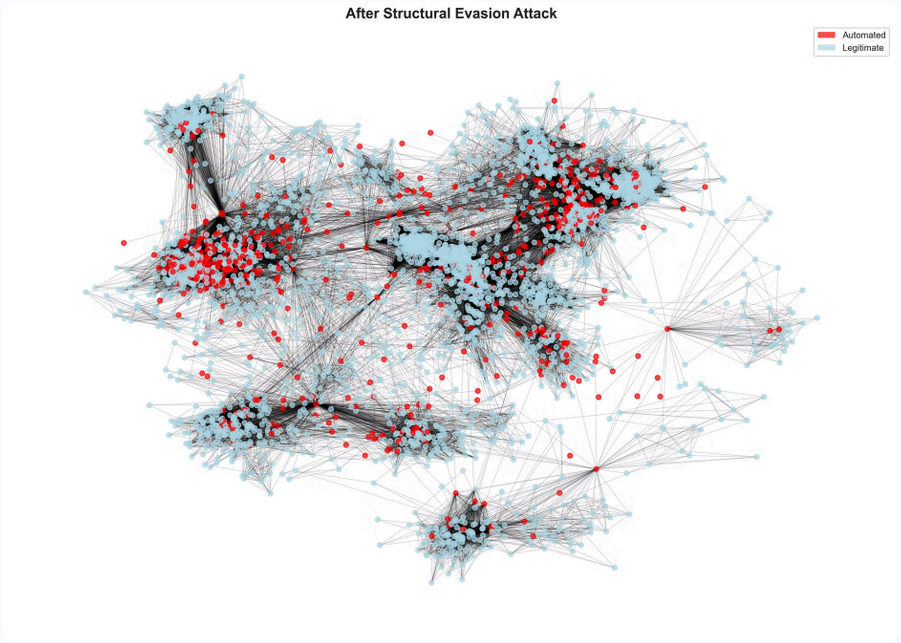
Baseline Model

- **Algorithm:** Random Forest Classifier
- **Parameters:** 100 estimators, max depth 10
- **Features:** All graph metrics except community ID
- **Train/Test Split:** 70/30 with stratification
- **Preprocessing:** Standard scaling



Made with GAMMA

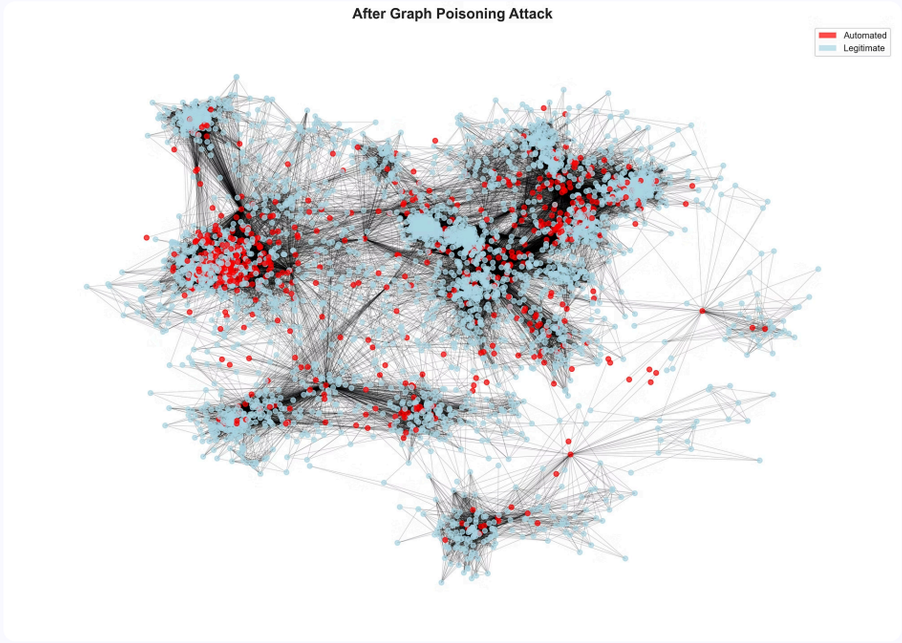
Methodology: Attack Methods



Attack 1: Structural Evasion

This attack changes the graph structure around bot nodes:

1. Add connections to create triangles (increase clustering)
2. Remove some connections to reduce degree
3. Connect to important nodes (look more legitimate)



Attack 2: Graph Poisoning

This attack corrupts the training data:

1. Add fake nodes that look like real users
2. Connect fake nodes to real users
3. Create connections between bots and important users
4. Label fake nodes as legitimate users

Made with GAMMA

Results: Graph Statistics and Baseline Performance

Graph Statistics

Original Graph:

- Nodes: 4,039
- Edges: 88,234
- Average Degree: 43.69
- Average Clustering Coefficient: 0.606
- Number of Communities: 77

Bot Distribution:

- Total Bots: 605
- Bot Ratio: 15.0%

Baseline Performance

Metric	Legitimate	Automated
Precision	1.00	0.99
Recall	1.00	0.98
F1-Score	1.00	0.99

Overall Accuracy: 99.59%

Feature Importance:

1. Clustering coefficient (local network density)
2. Betweenness centrality (bridging position)
3. PageRank (network influence)
4. Average neighbor degree (connection quality)
5. Eigenvector centrality (connection to important nodes)

Made with GAMMA

Results: Post-Attack Performance

After Structural Evasion:

Metric	Legitimate	Automated
Precision	1.00	0.97
Recall	1.00	0.99
F1-Score	1.00	0.98

Overall Accuracy: 99.42%

Accuracy Degradation: 0.17%

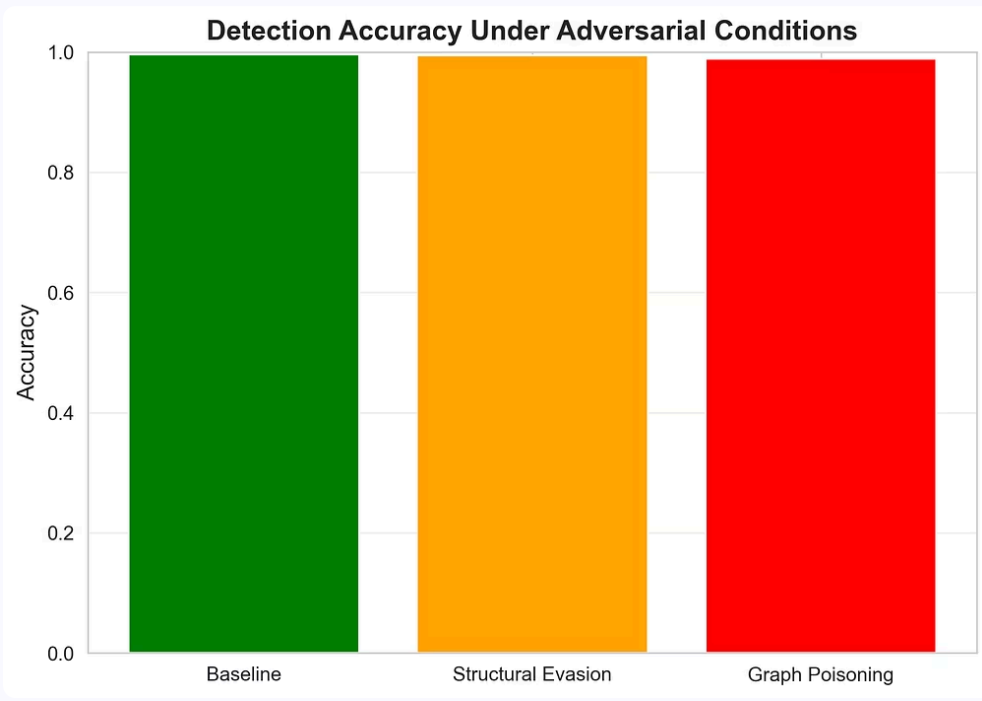
After Graph Poisoning:

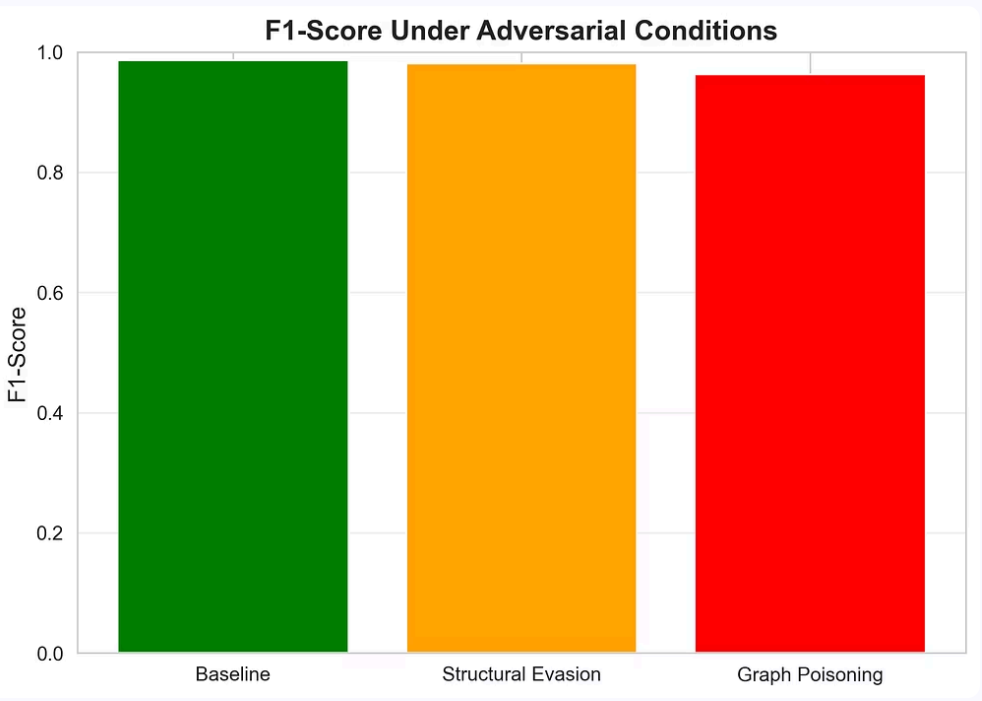
Metric	Legitimate	Automated
Precision	1.00	0.94
Recall	0.99	0.98
F1-Score	0.99	0.96

Overall Accuracy: 98.85%

Accuracy Degradation: 0.74%

Comparative Analysis





Condition	Accuracy	F1-Score	Precision (Bot)	Recall (Bot)
Baseline	99.59%	0.9862	0.99	0.98
Structural Evasion	99.42%	0.9809	0.97	0.99
Graph Poisoning	98.85%	0.9624	0.94	0.98

Made with GAMMA

Discussion: Impact of Attacks

What I Found - Structural Evasion

- Accuracy dropped by only 0.17%
- Precision for detecting bots dropped from 0.99 to 0.97
- The attack changed 20 bot nodes by increasing their clustering coefficient by 15-25%

What This Means:

- Making bots look more connected to communities helps them hide
- Connecting to important users makes bots seem more legitimate
- The attack worked but only had a small impact overall
- Bots became harder to detect because they looked more like normal users

What I Found - Graph Poisoning

- Accuracy dropped by 0.74%
- Precision for detecting bots dropped from 0.99 to 0.94
- Only 30 changes (15 fake nodes + 15 edges) caused this damage

What This Means:

- Poisoning the training data is more effective than structural evasion
- It was 4.4 times more effective at reducing accuracy
- The fake nodes confused the classifier during training
- This attack affects the model's overall ability to detect bots
- Adding bad data to the training set is a powerful attack

Made with GAMMA

Discussion: Comparing Attacks and Possible Defenses

Comparing the Two Attacks

Structural Evasion:

Good points:

- Targets specific bots
- Looks like normal network activity
- Doesn't change the overall graph much

Bad points:

- Not very effective (only 0.17% drop)
- Needs to modify many nodes for bigger impact

Graph Poisoning:

Good points:

- More efficient with fewer changes
- Much more effective (4.4x better)
- Affects the whole model

Bad points:

- Might be caught by anomaly detection
- Need to be able to add new nodes
- Fake nodes might be spotted

Possible Defenses

Ways to protect against these attacks:

- Use features that are harder to fake
- Detect unusual changes in the graph structure
- Track user behavior over time
- Use multiple detection methods together
- Clean the data before training

Made with GAMMA

Conclusion and Future Work

Summary of Results

- The bot detection model achieved 99.59% accuracy
- Structural evasion attack reduced accuracy by 0.17%
- Graph poisoning attack reduced accuracy by 0.74%
- Graph poisoning was much more effective (4.4x better)
- Both attacks made bots harder to detect
- The Facebook network has high clustering (0.606) and 77 communities

What I Learned

- Graph-based classifiers can be fooled by adversarial attacks
- The two attacks exploit different weaknesses
- We need multiple defense strategies to protect against attacks
- Using only graph features isn't enough when attackers are involved

Limitations of This Project

- The bot labels are based on heuristics, not real bot data
- Only used structural features, not content or behavior
- Only tested one type of classifier
- The attacks are simplified versions

Future Work

- Test with real bot datasets
- Try classifiers that are resistant to attacks
- Use Graph Neural Networks
- Combine structural, behavioral, and content features
- Build systems that can adapt to new attacks

Name: Ahmed Mohamed

Date: December 10, 2025

Assignment: Social Network Analysis with Bot Detection

Dataset: Facebook Egonets from Stanford SNAP

Made with GAMMA