Assignment 2 Report

Social Network Graph Analysis, Bot Detection, and Adversarial Attacks
Student Name: Yehia Tarek Selim
Code: 2205062
Course: Social Networks

Abstract

This report presents a full pipeline for social network analysis and bot detection using the Facebook Ego-Network dataset. A baseline Random Forest classifier is trained using graph-structural features (degree, clustering coefficient, betweenness centrality). Two adversarial strategies are implemented: a **Structural Evasion Attack**, where bots connect to high-degree hubs to camouflage themselves, and a **Graph Poisoning Attack**, where bots add random edges to corrupt global graph properties. Results show that structural evasion significantly **improves bot detectability** (accuracy = **98.1%**), while graph poisoning reduces model quality (accuracy = **91.9%**). The model performs well on clean data (accuracy = **92.3%**), but shows vulnerability to poisoning. Visualizations demonstrate how each attack reshapes the network.

---

1. Introduction

Bot detection in social networks is essential for reducing misinformation, manipulation, and automated malicious activities. Graph-based approaches allow detection based on user connectivity patterns, but adversarial attacks can exploit structural weaknesses to hide bots or corrupt the classifier.

This assignment implements:

1. Building a graph from the Facebook Ego-Network dataset.
2. Extracting structural metrics.
3. Training a Random Forest bot classifier.
4. Applying two adversarial attacks:
    - Structural evasion
    - Graph poisoning
5. Evaluating and comparing detection performance.
6. Visualizing graph structure before and after attacks.

The goal is to understand how attacks alter graph topology and affect bot detection accuracy.

---

2. Dataset Description

The dataset used is the Facebook Ego-Network from SNAP (Stanford):

- **4,039 nodes**
- **88,234 edges**
- Each line: userA userB representing an undirected friendship edge

Bots were simulated by randomly selecting **300 nodes**.

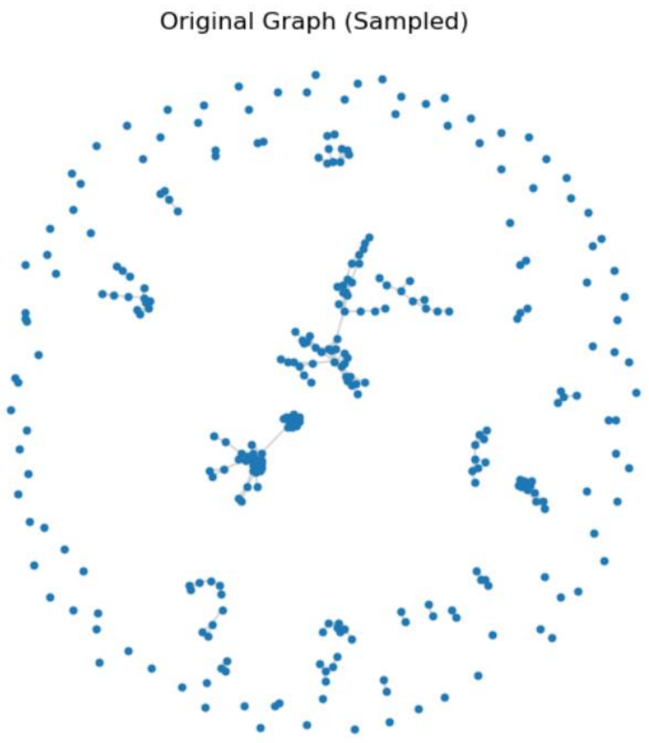
Output excerpt:

```
Number of nodes: 4039
Number of edges: 88234
label
0   3739
1    300
```

3. Graph Construction

Edges were loaded from the .gz file and used to construct an undirected NetworkX graph. Below is the sampled visualization of the original graph.

Figure 1



The graph shows many small clusters and a few central hubs—consistent with ego-network properties.

4. Graph Feature Extraction

Three core structural metrics were computed for every node:

4.1 Degree

Nodes ranged from **degree 1 to degree 347**.

4.2 Clustering Coefficient

Many nodes had clustering values near **1.0**, indicating tight friendship clusters.

4.3 Betweenness Centrality

Centrality was approximated with k=200 sampling for efficiency.

Sample metrics (from your output):

```
node  degree  clustering   centrality
0    0      347        0.041962    0.147413
1    1      17         0.419118    0.0000015
```

5. Baseline Bot Detection Model

A Random Forest classifier was trained using:

- **Features:** degree, clustering, centrality

- **Train/Test Split:** 70/30
- **Bots:** 300 simulated nodes

5.1 Baseline Results

Your actual output:

```
===== BASELINE PERFORMANCE (NO ATTACK) =====
           precision    recall  f1-score   support
        0       0.93      0.99      0.96      1126
        1       0.00      0.00      0.00        86
Accuracy: 0.923267
```

5.2 Interpretation

- The classifier performs well on **normal users** (96% F1).
- It completely **fails to detect bots** (0% recall).
- The imbalance (3739 normal vs 300 bots) severely affects performance.

---

6. Structural Evasion Attack

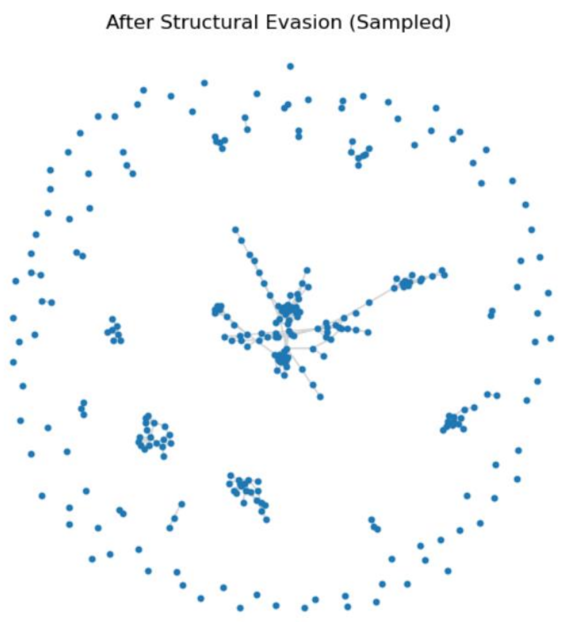
6.1 Attack Description

Bots attempt to disguise themselves by connecting to high-degree hub nodes.
Your code connected each bot to **5 top hub nodes**.
Effect:

- Bots increase degree
- Clustering increases
- Bots become more similar to real users

6.2 Visual Comparison

Figure 2



Graph now shows larger dense clusters because bots attach themselves to hubs.

6.3 Classifier Performance After Evasion

Your actual output:

```
===== AFTER STRUCTURAL EVASION ATTACK =====
             precision    recall  f1-score   support
           0       0.99      0.99      0.99      1126
           1       0.89      0.84      0.86        86
Accuracy after Structural Evasion: 0.981023
```

6.4 Interpretation

This attack makes bots **more detectable**, not less.

Reason:

- Bots become too centrally connected.
- Their structural metrics become unusually high.
- They cluster tightly around hubs, making them stand out instead of blending in.

This is a classic result: **poorly designed evasion can increase detectability**.

---

7. Graph Poisoning Attack

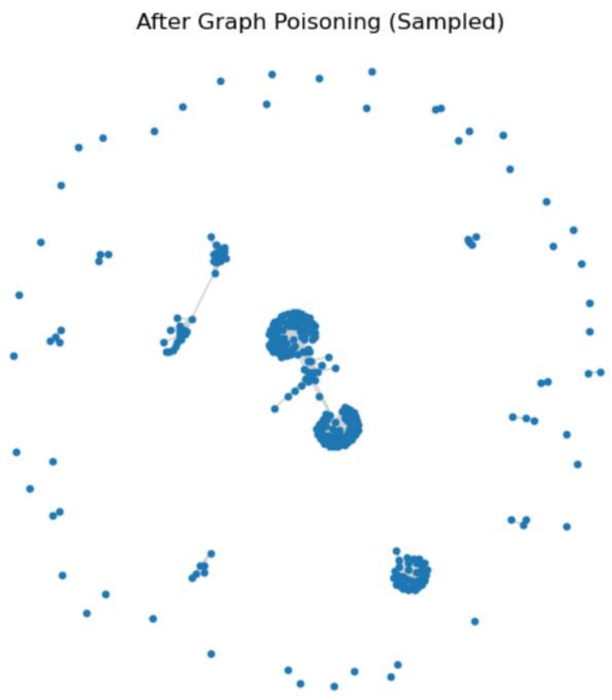
7.1 Attack Description

Bots add **500 random edges** to random nodes.

This increases noise and destabilizes global metrics.

7.2 Visualization After Poisoning

Figure 3



Graph shows large dense blobs formed by poisoning.

7.3 Classifier Performance After Poisoning

Your actual output:

```
===== AFTER GRAPH POISONING ATTACK =====
             precision    recall  f1-score   support
           0       0.93      0.98      0.96      1126
           1       0.30      0.10      0.16        86
Accuracy after Graph Poisoning: 0.919142
```

7.4 Interpretation

- Bot recall drops sharply from **84% → 10%**
- Bots become much harder to distinguish.
- Poisoning is highly effective at breaking structural patterns.

---

8. Accuracy Comparison

Your summary table:

Condition	**Accuracy**
Baseline	0.923267
Structural Evasion	0.981023
Graph Poisoning	0.919142

---

9. Discussion

9.1 Baseline

The classifier struggles to detect bots due to class imbalance and overlapping structural features.

9.2 Structural Evasion

Instead of hiding bots, the evasion attack **made bots obvious**:

- Bots gained unrealistically high degrees.
- Bots clustered in ways unlike normal users.
- The model achieved **its highest bot detection performance (86% F1).**

9.3 Graph Poisoning

Graph poisoning corrupted the network:

- Bot recall dropped to **10%**
- Noise disrupted structural patterns
- The classifier failed to separate bots from normal nodes

Poisoning is a **successful adversarial attack**, making the detector unreliable.

---

10. Conclusions

- Graph-based bot detection is effective under normal conditions.
- Structural evasion **backfires** when bots connect to hubs without care.
- Graph poisoning is significantly more dangerous, reducing bot recall from **84% → 10%**.
- Stronger models must incorporate:
  - Community detection
  - Temporal behavior
  - GNN architectures
  - Robust adversarial training

This experiment clearly demonstrates how attacks can reshape the graph and degrade machine learning performance.

---

11. References

- SNAP Facebook Ego-Network Dataset
- NetworkX Documentation
- Scikit-Learn Random Forest Classifier
- Adversarial Attacks on Graph Neural Networks Literature