THE UNIVERSITY OF DODOMA

COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE TITLE: Wireless Security

COURSE CODE: IA 423

GROUP 9

NO.	NAME	REG. NO
1	RAHIM SELEMANI	T21-03-03077
2	DAVID G. MUSA	T21-03-09156
3	WILLIAM JOHN	T21-03-15139
4	GODLOVE HIPOLITE	T21-03-01282
5	ISSA HEMED	T21-03-13932

Social Network Analysis

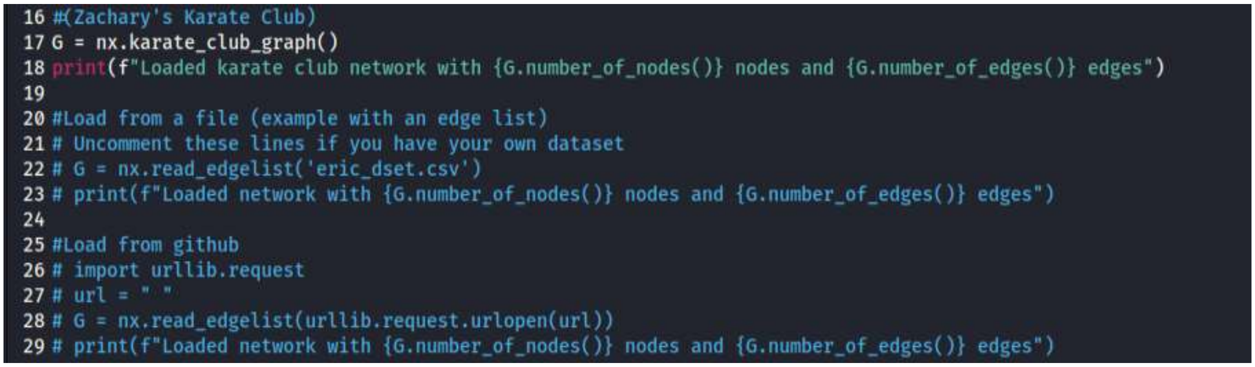
i. Set up your environment

A new Python environment is created and the necessary packages are installed.

ii. Finding datasets with relationship data (social networks)

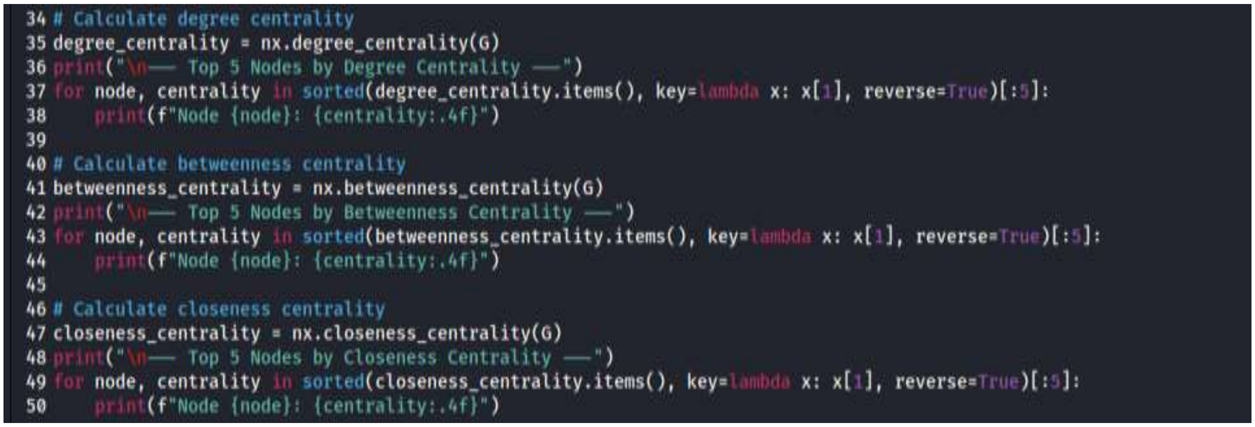
Social network dataset used was Zachary's Karate Club dataset which is built in **networkx**. External datasets can also be used.

iii. Load the network data

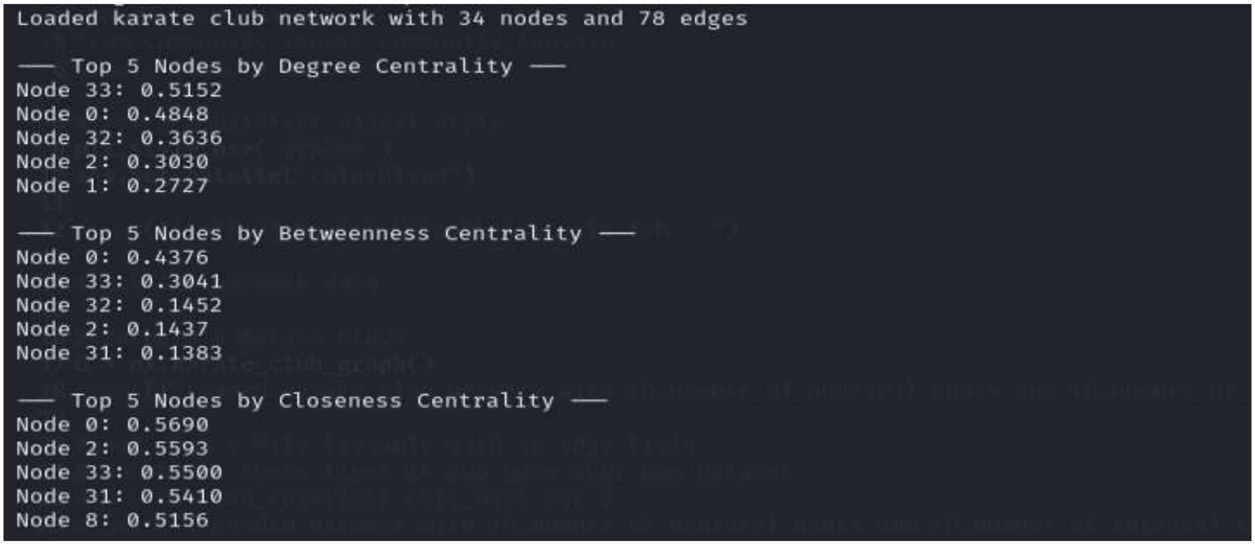


```python
#(Zachary's Karate Club)
G = nx.karate_club_graph()
print(f"Loaded karate club network with {G.number_of_nodes()} nodes and {G.number_of_edges()} edges")

#Load from a file (example with an edge list)
# Uncomment these lines if you have your own dataset
# G = nx.read_edgelist('eric_dset.csv')
# print(f"Loaded network with {G.number_of_nodes()} nodes and {G.number_of_edges()} edges")

#Load from github
# import urllib.request
# url = " "
# G = nx.read_edgelist(urllib.request.urlopen(url))
# print(f"Loaded network with {G.number_of_nodes()} nodes and {G.number_of_edges()} edges")
```

iv. Calculate network metrics

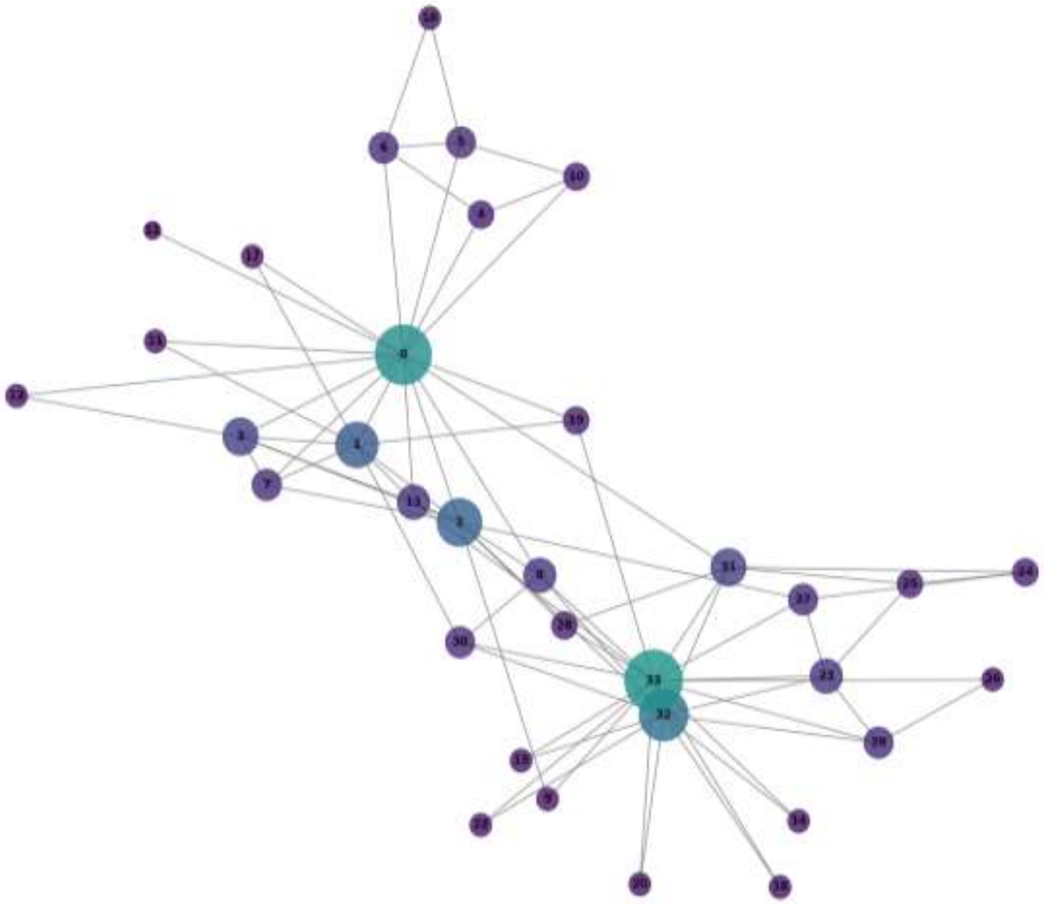


```python
# Calculate degree centrality
degree_centrality = nx.degree_centrality(G)
print("\n--- Top 5 Nodes by Degree Centrality ---")
for node, centrality in sorted(degree_centrality.items(), key=lambda x: x[1], reverse=True)[:5]:
    print(f"Node {node}: {centrality:.4f}")

# Calculate betweenness centrality
betweenness_centrality = nx.betweenness_centrality(G)
print("\n--- Top 5 Nodes by Betweenness Centrality ---")
for node, centrality in sorted(betweenness_centrality.items(), key=lambda x: x[1], reverse=True)[:5]:
    print(f"Node {node}: {centrality:.4f}")

# Calculate closeness centrality
closeness_centrality = nx.closeness_centrality(G)
print("\n--- Top 5 Nodes by Closeness Centrality ---")
for node, centrality in sorted(closeness_centrality.items(), key=lambda x: x[1], reverse=True)[:5]:
    print(f"Node {node}: {centrality:.4f}")
```

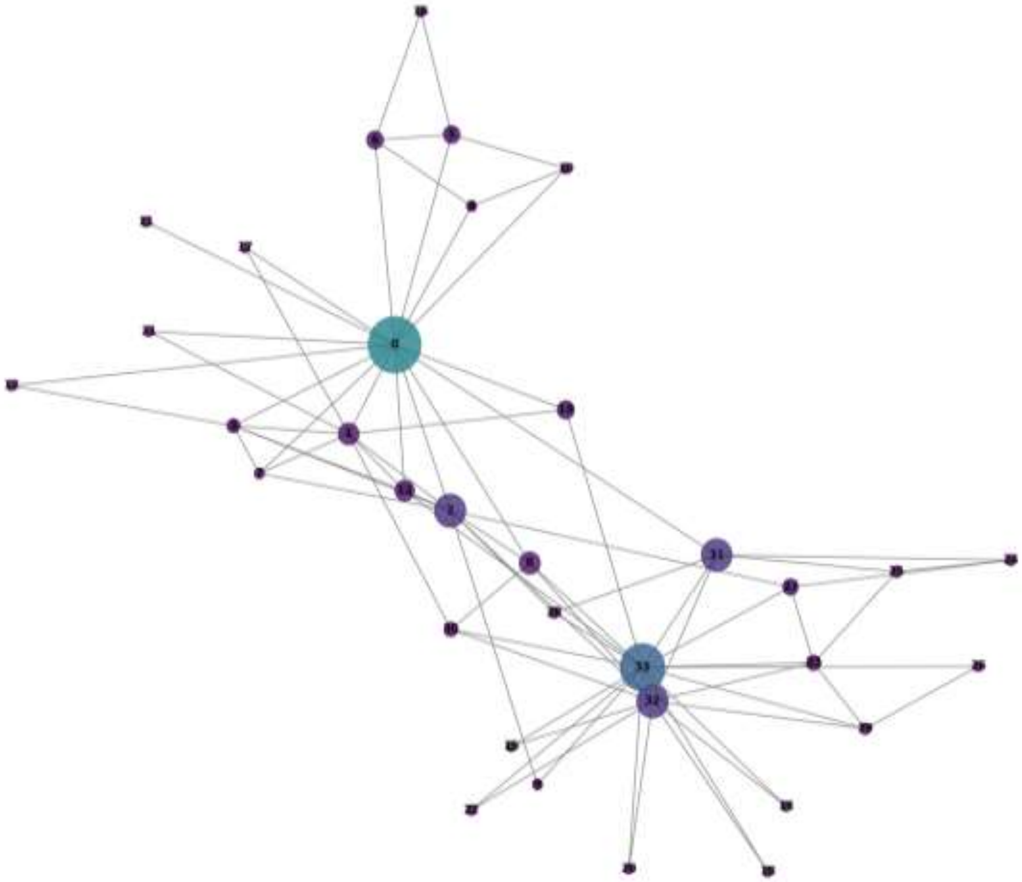


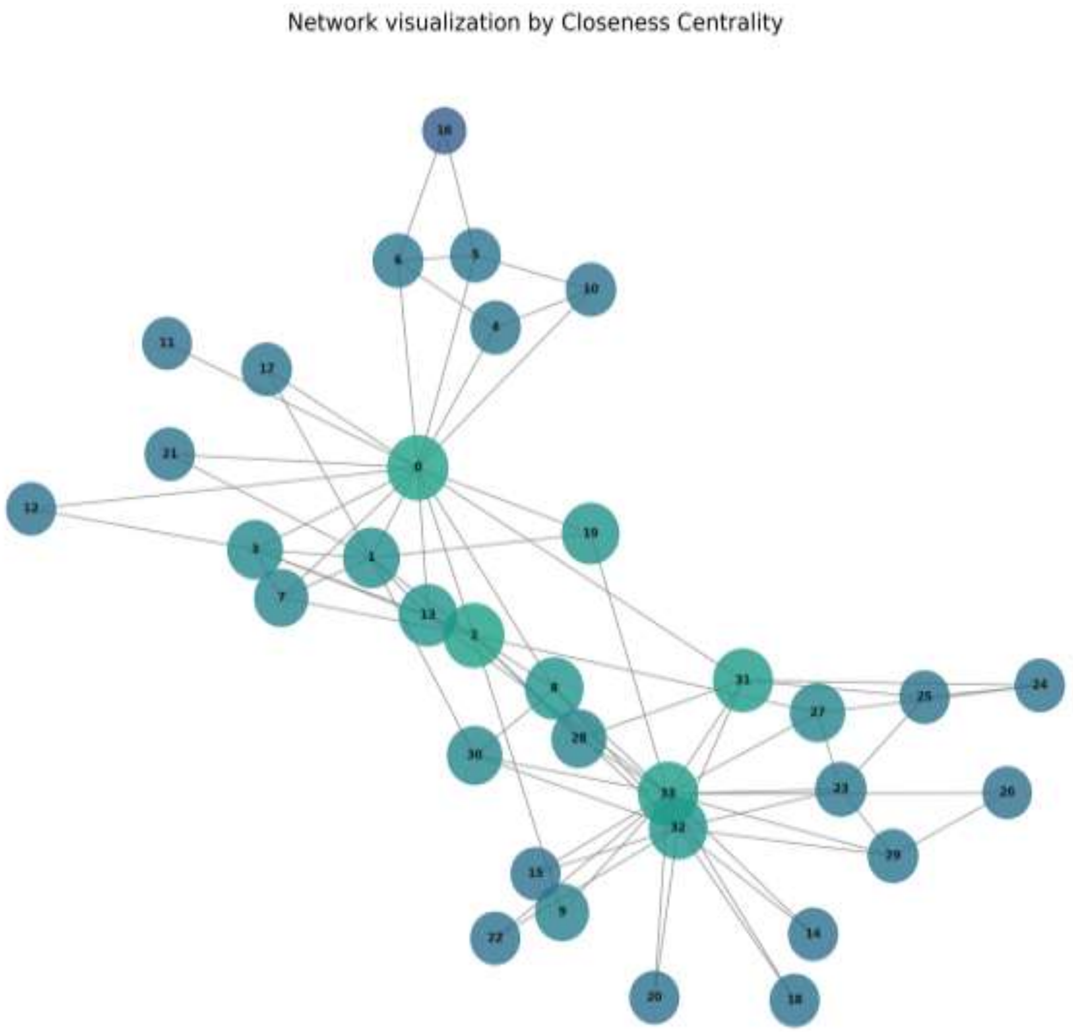
```
Loaded karate club network with 34 nodes and 78 edges

—— Top 5 Nodes by Degree Centrality ——
Node 33: 0.5152
Node 0: 0.4848
Node 32: 0.3636
Node 2: 0.3030
Node 1: 0.2727

—— Top 5 Nodes by Betweenness Centrality ——
Node 0: 0.4376
Node 33: 0.3041
Node 32: 0.1452
Node 2: 0.1437
Node 31: 0.1383

—— Top 5 Nodes by Closeness Centrality ——
Node 0: 0.5690
Node 2: 0.5593
Node 33: 0.5500
Node 31: 0.5410
Node 8: 0.5156
```

Network visualization by Degree Centrality



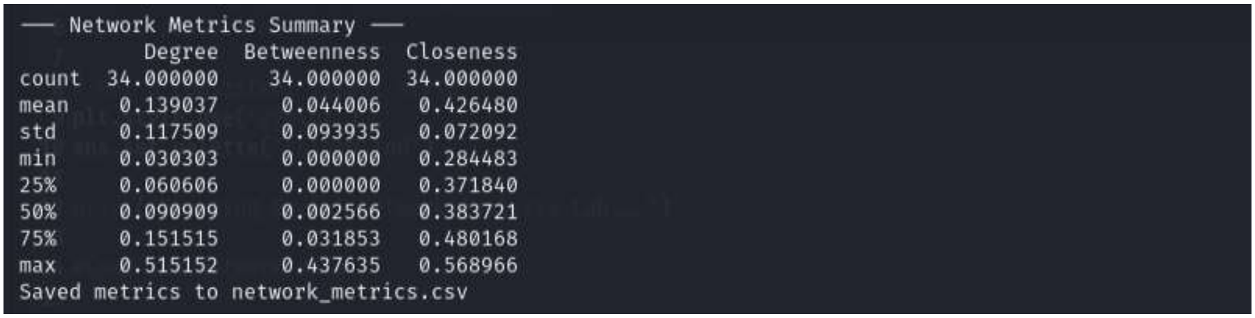
Network visualization by Betweenness Centrality





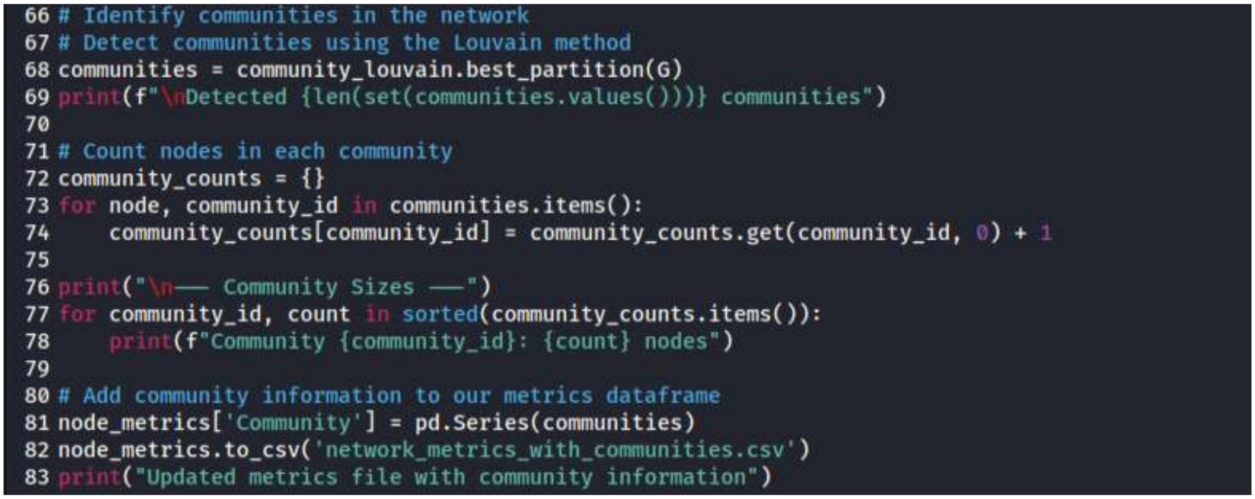
Network visualization by Closeness Centrality

- ✓ Create a summary dataframe of all metrics
- ✓ Save metrics to CSV

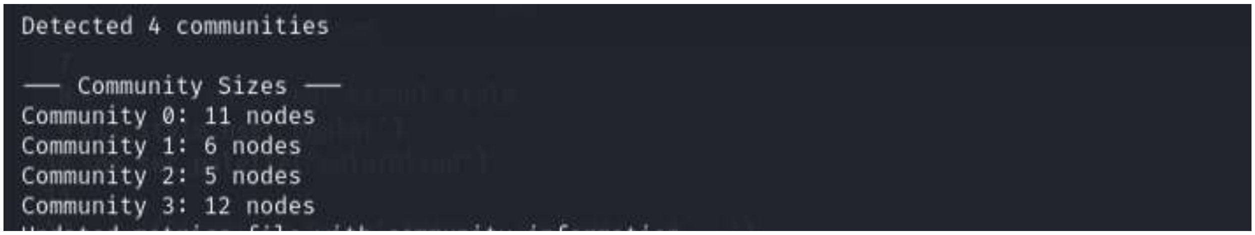


```
── Network Metrics Summary ──
         Degree  Betweenness  Closeness
count  34.000000    34.000000  34.000000
mean    0.139037     0.044006   0.426480
std     0.117509     0.093935   0.072092
min     0.030303     0.000000   0.284483
25%     0.060606     0.000000   0.371840
50%     0.090909     0.002566   0.383721
75%     0.151515     0.031853   0.480168
max     0.515152     0.437635   0.568966
Saved metrics to network_metrics.csv
```

v. Identify communities in the network

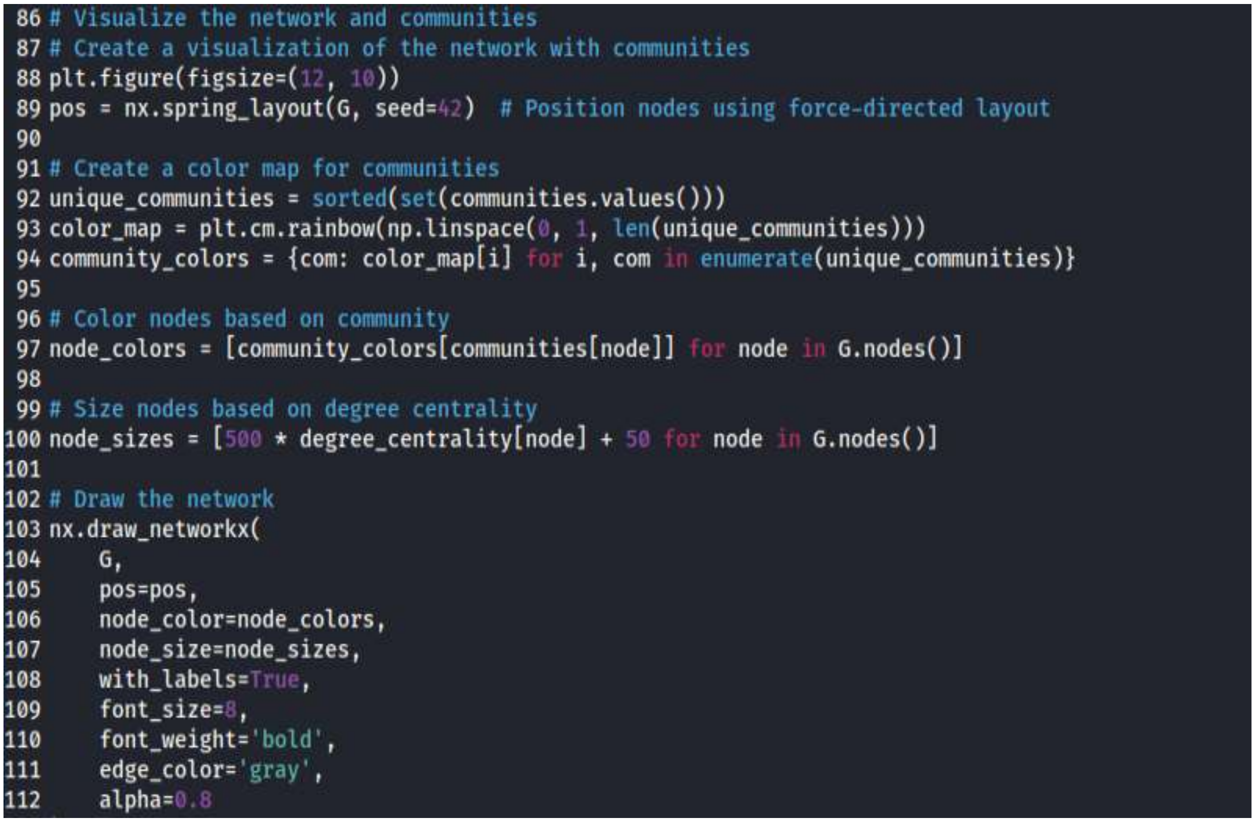


```python
# Identify communities in the network
# Detect communities using the Louvain method
communities = community_louvain.best_partition(G)
print(f"\nDetected {len(set(communities.values()))} communities")

# Count nodes in each community
community_counts = {}
for node, community_id in communities.items():
    community_counts[community_id] = community_counts.get(community_id, 0) + 1

print("\n—— Community Sizes ——")
for community_id, count in sorted(community_counts.items()):
    print(f"Community {community_id}: {count} nodes")

# Add community information to our metrics dataframe
node_metrics['Community'] = pd.Series(communities)
node_metrics.to_csv('network_metrics_with_communities.csv')
print("Updated metrics file with community information")
```

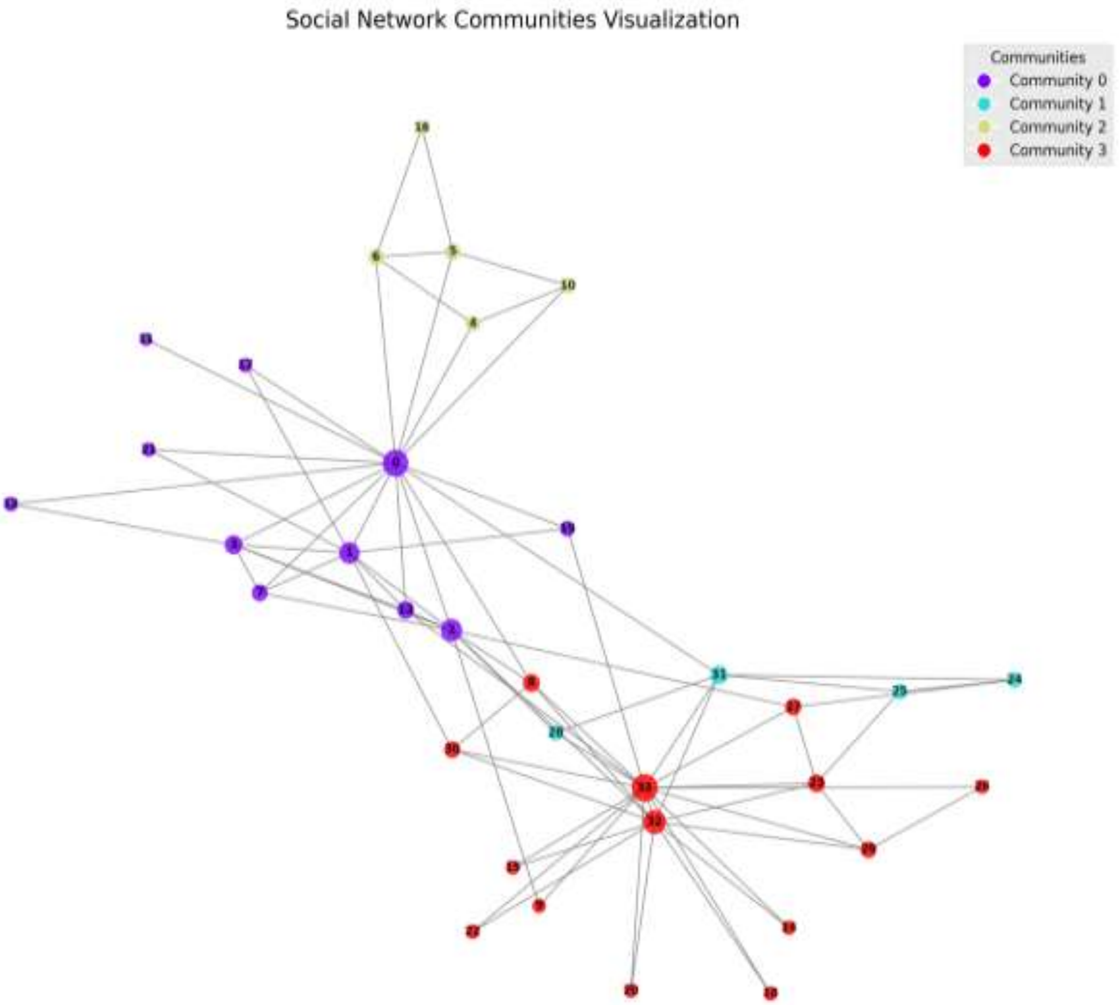


```
Detected 4 communities

—— Community Sizes ——
Community 0: 11 nodes
Community 1: 6 nodes
Community 2: 5 nodes
Community 3: 12 nodes
```

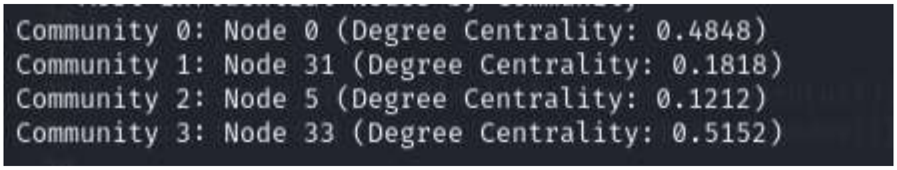
vi. Visualize the network and communities



```python
# Visualize the network and communities
# Create a visualization of the network with communities
plt.figure(figsize=(12, 10))
pos = nx.spring_layout(G, seed=42)  # Position nodes using force-directed layout

# Create a color map for communities
unique_communities = sorted(set(communities.values()))
color_map = plt.cm.rainbow(np.linspace(0, 1, len(unique_communities)))
community_colors = {com: color_map[i] for i, com in enumerate(unique_communities)}

# Color nodes based on community
node_colors = [community_colors[communities[node]] for node in G.nodes()]

# Size nodes based on degree centrality
node_sizes = [500 * degree_centrality[node] + 50 for node in G.nodes()]

# Draw the network
nx.draw_networkx(
    G,
    pos=pos,
    node_color=node_colors,
    node_size=node_sizes,
    with_labels=True,
    font_size=8,
    font_weight='bold',
    edge_color='gray',
    alpha=0.8
```



Identification the most influential nodes in each community



```
Community 0: Node 0 (Degree Centrality: 0.4848)
Community 1: Node 31 (Degree Centrality: 0.1818)
Community 2: Node 5 (Degree Centrality: 0.1212)
Community 3: Node 33 (Degree Centrality: 0.5152)
```

Understanding The Results

1. **Key metrics**:

   a. **Degree Centrality**: Measures how many connections each node has. Higher values indicate nodes that connect to many others.

b. **Betweenness Centrality**: Measures how often a node lies on the shortest path between other nodes. High betweenness nodes act as "bridges" between different parts of the network.

c. **Closeness Centrality**: Measures how close a node is to all other nodes in the network. Nodes with high closeness can quickly reach other nodes.

2. **Interpreting Communities**:

Detection of communities is by using the Louvain method, which identifies groups of nodes that are more densely connected to each other than to the rest of the network. Each community represents a cluster of closely related nodes.

3. **Analyzing the Visualizations**:

    1. The main visualization (network_communities.png) shows the network with nodes colored by community.

    2. The metric-specific visualizations show how each centrality measure is distributed across the network.

    3. Larger nodes indicate higher values for the respective metrics.