Research Question

What are the attributes of social networks on Facebook, and can we detect factors that have a substantial impact in the establishment of these networks?

Introduction:

The research aims to investigate the attributes of social media networks on Facebook and discern factors that play a noteworthy role in their establishment. To accomplish this, we employ tools for analyzing social networks, specifically utilizing the NetworkX library in Python. The analysis encompasses various facets, including the structure of the network, the distribution of degrees, the analysis of connected components, the examination of paths, measures of centrality, coefficients of clustering, density, and comparisons with randomly generated, scale-free, and small-world network models.

Methodology:

1. Data Retrieval and Formulation: The data of the social media setup on Facebook is found from a tar.gz file. The process of taking out reveals the presence of related files, including "facebook_combined.txt.gz," which contains the condition of the network in the form of an edge directory.

2. Network Construction and Analysis: The Facebook network is formed using NetworkX as a guided graph. The largest weakly connected component is detected and examined for its structural assets.

3. Analysis of Degree Distribution: The division of degrees, which represents the number of connections each node retains, is envisaged through a histogram. This presents intuitions into the overall pattern of connectivity in the network.

4. Analysis of Connected Components: The number of weakly connected components in the network is revealed. This case, there is only one weakly connected component.

5. Analysis of Paths: Shortest path lengths are figured, and the average shortest path length is evaluated. This metric suggests insights into the overall availability and efficiency of the stream of data in the network.

6. Analysis of Centrality: Degree centrality, uncovering of the importance of a node founded on its connections, is calculated. The centrality of the first five nodes is shown.

7. Analysis of Clustering Coefficient and Density: The average clustering coefficient and network density are computed. These metrics provide information on the local clustering patterns and the overall size of the network.

8. Comparison with ER, BA, and WS Graphs: We conducted a comparative analysis of three distinct network models: Erds-Rényi (ER), Barabási-Albert (BA), and Watts-Strogatz (WS).

This comparative study enables a comprehensive understanding of the structural characteristics of the declared social network in relation to well-established theoretical models.

A) Erdős-Rényi (ER) Model: The ER model is a reference standard for random network formation, where connections between nodes are established independently with a fixed likelihood. Unlike the observed network, ER graphs lean to have a more homogeneous degree distribution due to their random nature. The comparison with the ER model offers insights into whether the Facebook network has a more deterministic or organized pattern in its connection.

B) Barabási-Albert (BA) Model: The BA model, renowned for preferential attachment, captures the occurrence of hubs drawing more connections, resulting in a scale-free degree distribution. In contrast to ER, BA networks exhibit a heterogeneous connectivity pattern with several highly connected nodes. Assessing the Facebook network against the BA model helps determine if the network growth follows a preferential attachment mechanism, indicating the advent of influential nodes over the course of time.

C) Watts-Strogatz (WS) Model: The WS model generates small-world networks characterized by a combination of local clustering and global connectivity. By using rewiring with a certain possibility, WS graphs maintain both local groups and efficient long-distance connections. Comparing the Facebook network to the WS model scales light on whether the examined network determines characteristics of a small-world structure, which is often observed in social networks.

9. Implications for Interpretation: Deviations or similarities in terms of edge distribution, clustering coefficients, and other relevant statistics between the Facebook network and these theoretical models provide insights into the distinct structural attributes of the social network. Such a comparative analysis aids in understanding whether the observed network aligns more closely with random, preferential attachment, or small-world models. This contextualization enriches the discussion on the real-world implications of the Facebook social network's structure, offering a deeper understanding of its formation and organizational principles.

10. Visualization: The Facebook network, along with the ER, BA, and WS models, is visually represented to facilitate a better comprehension of their structures.

11. Ego Network Analysis: An ego network is created for a randomly selected node, and its visualization provides insights into the immediate connections of that node.

Results and Discussion:

The Facebook network comprises 4039 nodes and 88234 edges within the largest weakly connected component. The degree distribution reveals a heterogeneous connectivity pattern, with some nodes having significantly higher degrees than others. The network is highly connected, with only one weakly connected component, suggesting a cohesive

structure. The average shortest path length is 4.33, indicating relatively short paths for information distribution. Degree centrality analysis focuses on nodes with differing degrees of importance, providing insight into potential influencers. The clustering coefficient is 0.30, implying a moderate level of local clustering, while the density is 0.0054, showing a sparse network. The assessment between ER, BA, and WS models shows distinct differences in terms of edges and clustering coefficients, highlighting the uniqueness of the Facebook network. Visualizations provide a clear overview of the network structures, highlighting the joined nature of the Facebook network.

Conclusion:

The analysis gives a comprehensive overview of the attributes of the Facebook social network. The findings suggest that a heterogeneous structure with significant nodes, efficient information flow, and distinct clustering patterns. The comparison with artificial models underscores the distinct characteristics of the observed network.

Recommendations for Future Work:

Future research could explore temporal aspects, believing the evolution of the network over time. Additionally, integrating node attributes and content analysis might provide deeper perceptions into the factors influencing network subtleties. Furthermore, research on community detection algorithms and their application to recognize subgroups within the network would improve our understanding of social structures on Facebook.

Code for Analysis

In [ ]:
```
# Required Libraries Installation
pip install networkx
```

In [ ]:
```
# Required Libraries Installation
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\pmyls\miniconda3\envs\keras-tf\lib
\site-packages (1.26.2)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
# Required Libraries Installation
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\pmyls\miniconda3\envs\keras-tf
\lib\site-packages (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\pmyls\miniconda3\envs\ke
ras-tf\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\pmyls\miniconda3\envs\keras-
tf\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\pmyls\miniconda3\envs\k
eras-tf\lib\site-packages (from matplotlib) (4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\pmyls\miniconda3\envs\k
eras-tf\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\pmyls\miniconda3\envs\kera
s-tf\lib\site-packages (from matplotlib) (1.26.2)
Requirement already satisfied: packaging>=20.0 in c:\users\pmyls\miniconda3\envs\ker
as-tf\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\pmyls\miniconda3\envs\keras-tf
\lib\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\pmyls\miniconda3\envs\ke
ras-tf\lib\site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\pmyls\miniconda3\env
s\keras-tf\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\pmyls\miniconda3\envs\keras-tf\l
ib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```python
# Importing Libraries
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import random
import tarfile
import os
```

In [ ]:
```python
# Data Extraction from Tar.gz File

# Specify the path to tar.gz file
tar_file_path = "facebook.tar.gz"

# Specify the extraction directory
extraction_dir = "."

# Extract the contents of the tar.gz file
with tarfile.open(tar_file_path, 'r:gz') as tar:
    tar.extractall(extraction_dir)

# List the files in the extraction directory
extracted_files = os.listdir(extraction_dir)
print("Extracted files:", extracted_files)
```

```
Extracted files: ['a', 'facebook', 'facebook.tar.gz', 'facebook_combined.txt.gz', 'N
etwork code.ipynb', 'New folder', 'twitter', 'twitter.tar.gz', 'twitter_combined.tx
t.gz', 'Untitled-1.ipynb', 'wikipedia']
```

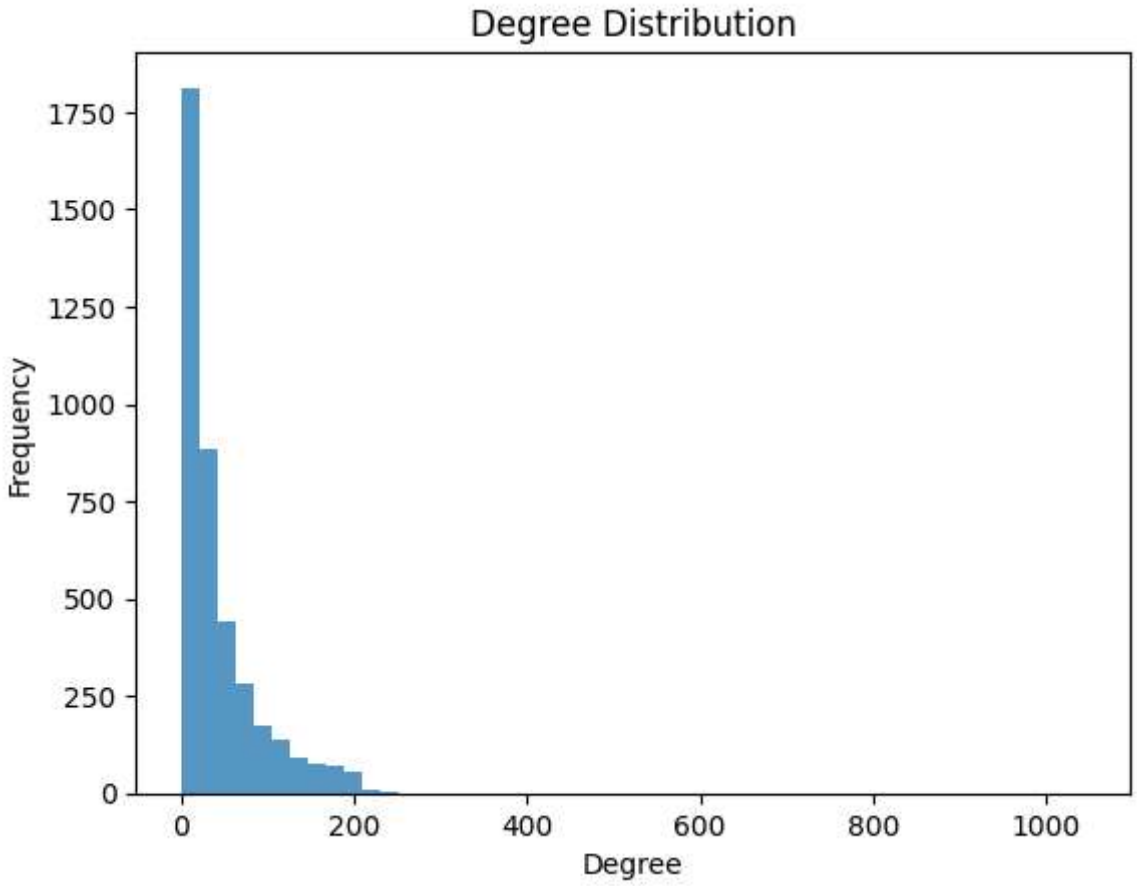
In [ ]:
```python
# Graph Construction and Analysis
file_path = "facebook_combined.txt.gz"
# Reading the edgelist file into a directed graph
G = nx.read_edgelist(file_path, create_using=nx.DiGraph)
```

In [ ]:
```
# Finding the largest weakly connected component
largest_wcc = max(nx.weakly_connected_components(G), key=len)
# Extracting the largest weakly connected component
G_largest_wcc = G.subgraph(largest_wcc).copy()
```

In [ ]:
```
# Visualizing nodes and edges
print("Number of nodes:", G_largest_wcc.number_of_nodes())
print("Number of edges:", G_largest_wcc.number_of_edges())
```

```
Number of nodes: 4039
Number of edges: 88234
```

In [ ]:
```
# Network Analysis
# Degree Distribution Analysis
# Extracting the degree sequence of nodes
degree_sequence = [d for n, d in G_largest_wcc.degree()]
# Extracting the degree sequence of nodes
plt.hist(degree_sequence, bins=50, alpha=0.75)
plt.title("Degree Distribution")
plt.xlabel("Degree")
plt.ylabel("Frequency")
plt.show()
```



In [ ]:
```
# Connected Components Analysis
# Finding the number of weakly connected components
connected_components = nx.number_weakly_connected_components(G)
print("Number of connected components:", connected_components)
```

```
Number of connected components: 1
```

In [ ]:
```
# Path Analysis
# Calculating shortest path lengths
shortest_path_lengths = nx.shortest_path_length(G)
# Extracting all path lengths
all_path_lengths = [length for source_paths in shortest_path_lengths for length in
# Calculating the average shortest path length
average_shortest_path_length = np.mean(all_path_lengths)
print("Average Shortest Path Length:", average_shortest_path_length)
```

```
Average Shortest Path Length: 4.330770048337255
```

In [ ]:
```
# Centrality Analysis
# Calculating degree centrality
degree_centrality = nx.degree_centrality(G)
print("Degree Centrality of the first 5 nodes:", {node: degree_centrality[node] for
```

```
Degree Centrality of the first 5 nodes: {'0': 0.08593363051015354, '1': 0.0042100049
52947003, '2': 0.002476473501733 5313, '3': 0.004210004952947003, '4': 0.002476473501
7335313}
```

In [ ]:
```
# Clustering Coefficient, Density Analysis
# Calculating the average clustering coefficient
clustering_coefficient = nx.average_clustering(G)
# Calculating the graph density
density = nx.density(G)
print("Clustering Coefficient:", clustering_coefficient)
print("Density:", density)
```

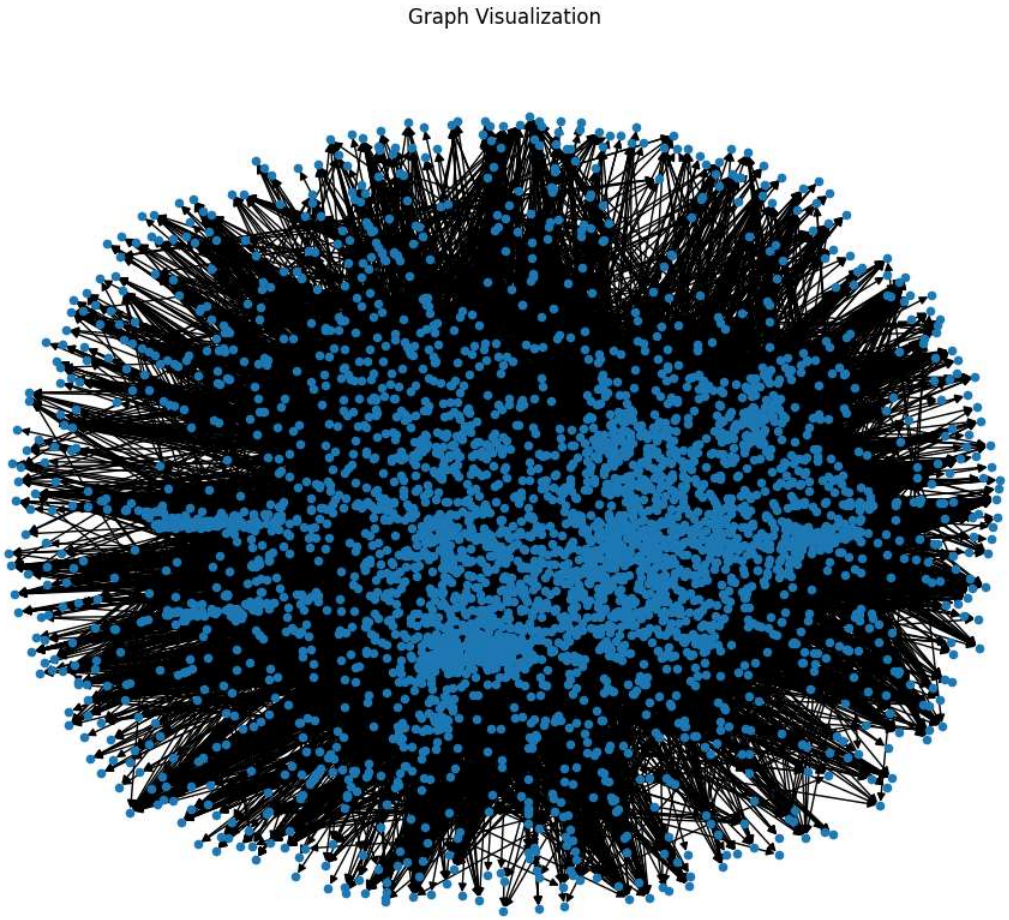
```
Clustering Coefficient: 0.3027733593100438
Density: 0.0054099817517196435
```

In [ ]:
```
# Comparison with ER, BA, and WS Graphs
# Generate ER, BA, and WS graphs with a similar number of nodes and edges
# Generating an Erdos-Renyi graph
random_graph = nx.erdos_renyi_graph(G.number_of_nodes(), p=0.01)
# Generating a Barabasi-Albert graph
ba_graph = nx.barabasi_albert_graph(G.number_of_nodes(), m=5)
# Generating a Watts-Strogatz graph
ws_graph = nx.watts_strogatz_graph(G.number_of_nodes(), k=4, p=0.05)
```

In [ ]:
```
# Compute and compare relevant statistics
print("Comparison with ER Graph:")
print("ER Graph Number of edges:", random_graph.number_of_edges())
print("ER Graph Clustering Coefficient:", nx.average_clustering(random_graph.to_und

print("Comparison with BA Graph:")
print("BA Graph Number of edges:", ba_graph.number_of_edges())
print("BA Graph Clustering Coefficient:", nx.average_clustering(ba_graph.to_undirec

print("Comparison with WS Graph:")
print("WS Graph Number of edges:", ws_graph.number_of_edges())
print("WS Graph Clustering Coefficient:", nx.average_clustering(ws_graph.to_undirec
```

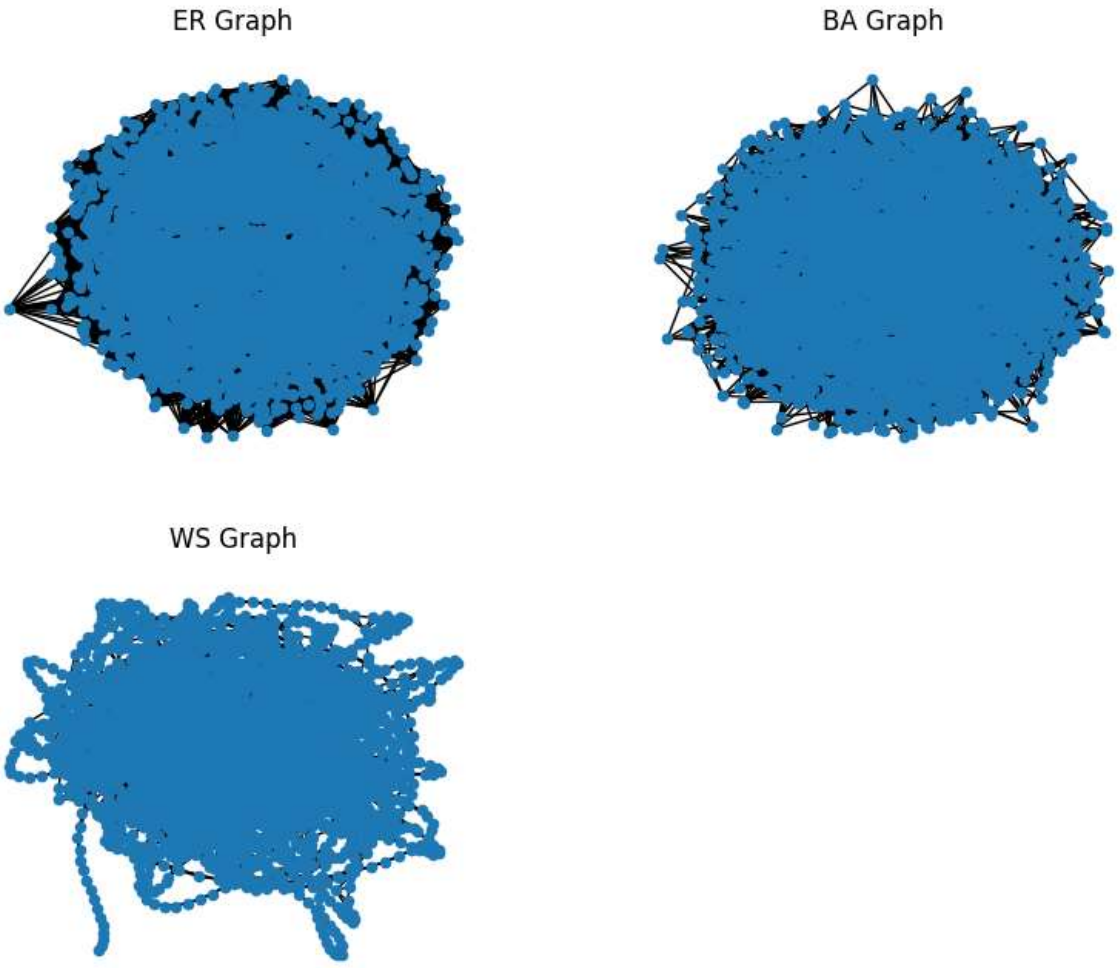
```
Comparison with ER Graph:
ER Graph Number of edges: 81792
ER Graph Clustering Coefficient: 0.010087708137532294
Comparison with BA Graph:
BA Graph Number of edges: 20170
BA Graph Clustering Coefficient: 0.014098638245716215
Comparison with WS Graph:
WS Graph Number of edges: 8078
WS Graph Clustering Coefficient: 0.43913509944705426
```

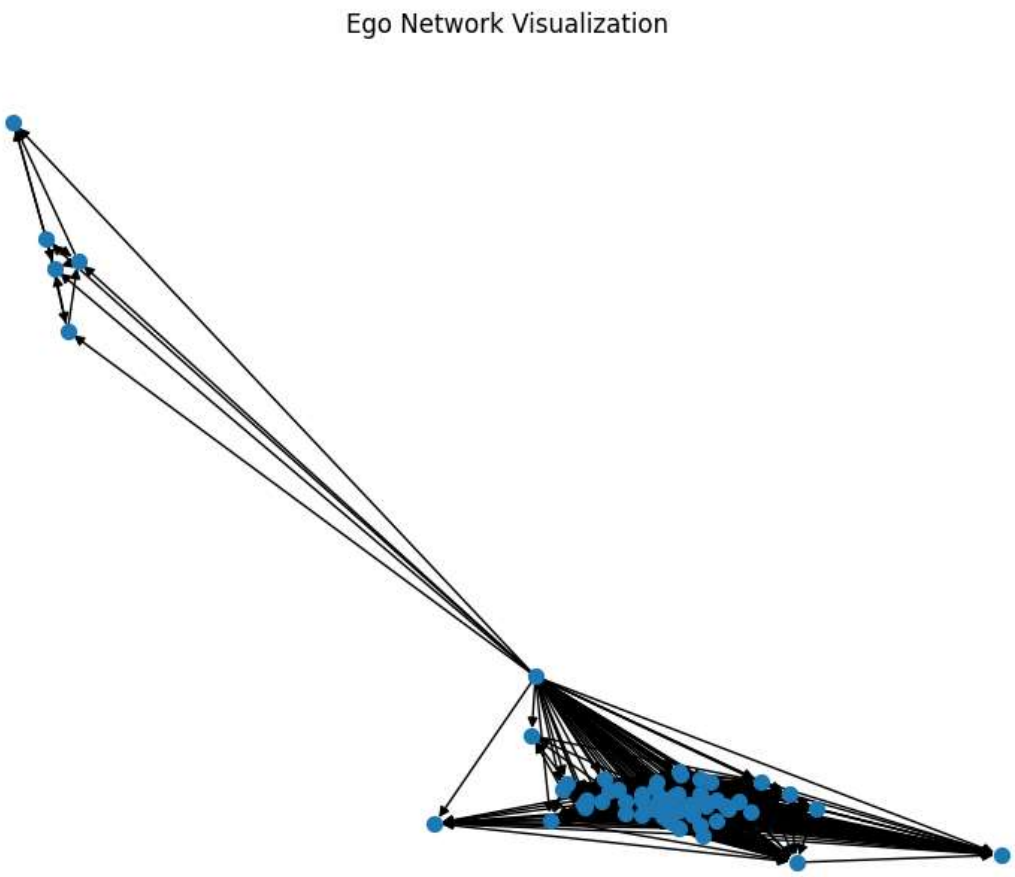
In [ ]:
```
# Visualization of the Graph
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=False, node_size=20)
plt.title("Graph Visualization")
plt.show()
```



In [ ]:
```
# Visualization of ER, BA, and WS Graphs
plt.figure(figsize=(10, 8))

plt.subplot(2, 2, 1)
nx.draw(random_graph, with_labels=False, node_size=20)
plt.title("ER Graph")

plt.subplot(2, 2, 2)
nx.draw(ba_graph, with_labels=False, node_size=20)
```

```
plt.title("BA Graph")

plt.subplot(2, 2, 3)
nx.draw(ws_graph, with_labels=False, node_size=20)
plt.title("WS Graph")

plt.show()
```





In [ ]:
```
# Ego Network Analysis
ego_node = random.choice(list(G.nodes()))
ego_network = nx.ego_graph(G, ego_node, radius=1)
```

In [ ]:
```
# Visualization of the Ego Network
plt.figure(figsize=(8, 6))
pos_ego = nx.spring_layout(ego_network)
nx.draw(ego_network, pos_ego, with_labels=False, node_size=50)
plt.title("Ego Network Visualization")
plt.show()
```

Github Repository Link:(https://github.com/MuhammadAhmadJamil18/B107_Data-Decision-.git)