Network Analysis & Modeling

Final project

By Changlong JI

Submission date：23/02/2024

Github link: https://github.com/CharlieChee/TSP_network_analysis/

Question 1: Reading

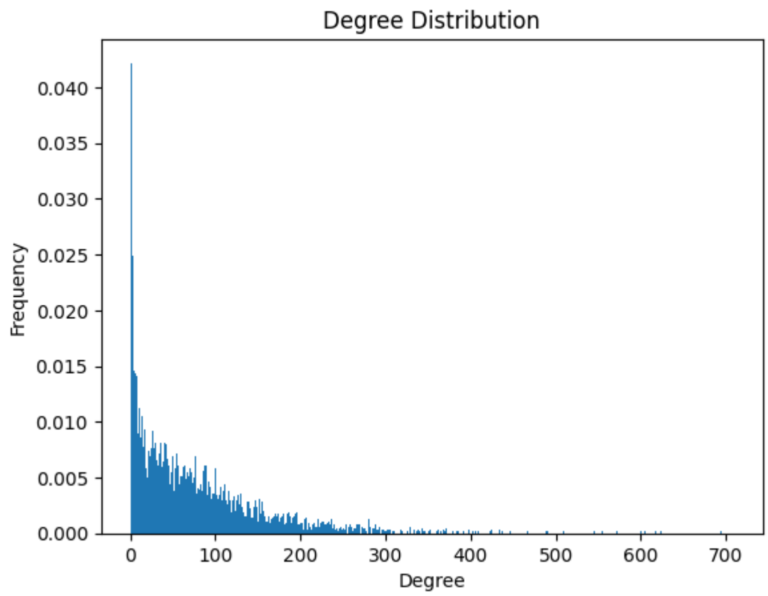
Read the following documents [3, 2, 1]

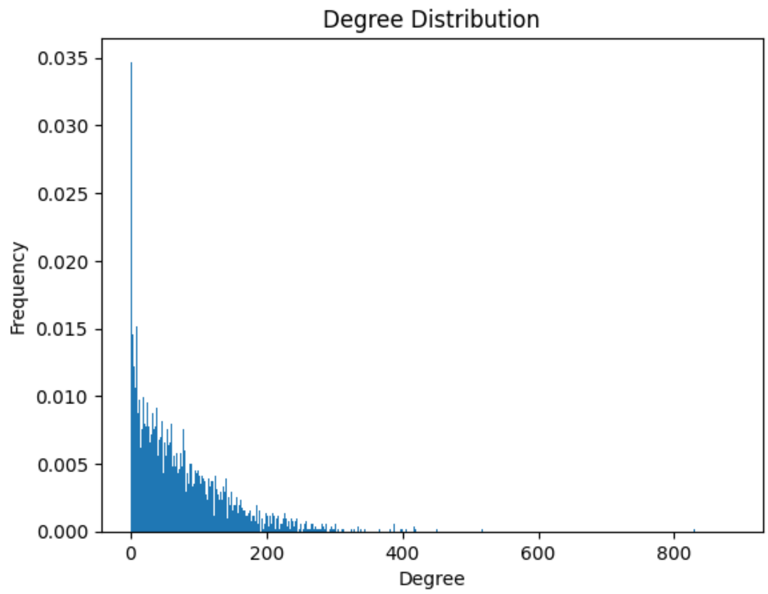
Question 2: Social Network Analysis with the Facebook100 Dataset

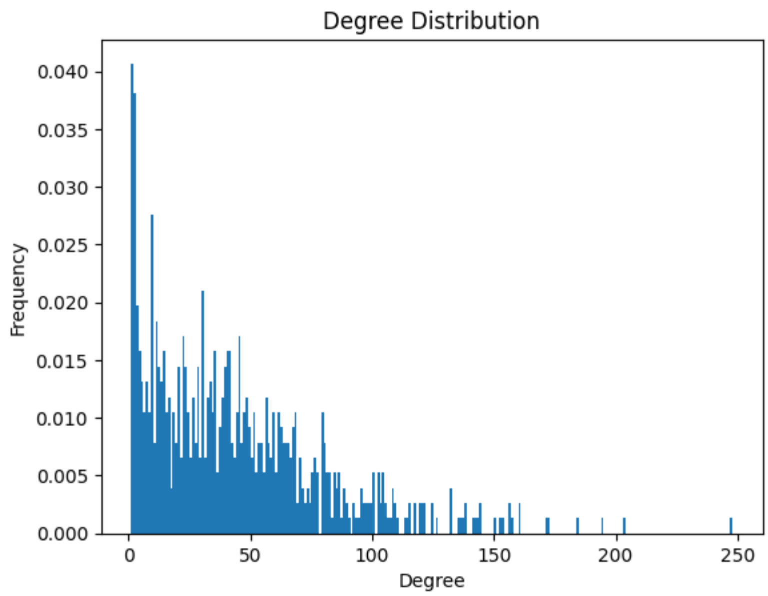
The smallest network (Caltech) has 762 nodes in the largest connected component (LCC), and the largest has more than 40000 nodes in the LCC.

Lets use three networks from the FB100: Caltech (with 762 nodes in the LCC), MIT (which has 6402 nodes in the LCC), and Johns Hopkins (which has 5157 nodes in the LCC).

1. (a) For these three networks plot the degree distribution for each of the three net- works that you downloaded. What are you able to conclude from these degree distributions?







Heavy-tailed Distributions: Both plots show a distribution where a small number of nodes have a high degree, and a large number of nodes have a low degree. This is characteristic of heavy-tailed distributions, often found in real-world networks such as social networks or the internet.

Scale-free Properties: The distributions appear to follow a power law, at least in the tail (for the higher degrees).

Possible Outliers: In the first plot, there is a sharp peak at the lower end of the degree spectrum, which might suggest the presence of outliers or a large number of nodes with a very specific degree, which could indicate a common pattern or structure within the network.

Heterogeneity of Node Connectivity: Both plots suggest that the networks are heterogeneous in terms of node connectivity, meaning not every node has the same number of links, which is often the case in non-random networks.

Sparse Connectivity: Given the rapid drop-off in frequency as degree increases, most nodes have a relatively small number of connections, indicating that the networks are sparse rather than densely connected.

2. (b) Compute the global clustering coefficient and mean local clustering coefficient for each of the 3 networks. In addition compute the edge density of each network. Should either of these networks be construed as sparse? Based on

the density information and the clustering information what can you said about the graph topology?

	global_clustering	mean_local_clustering	edge_density
Caltech	0.2912809635141533	0.409117304833461	0.05742892519512591
MIT	0.1802884023054581	0.272359996588386	0.012261341741110527
Johns Hopkins	0.19316115952994883	0.2690083618058958	0.0140335136902954

Edge Density:

The edge density of a graph is the ratio of the number of edges present to the number of possible edges. Smaller values indicate a sparser graph.

For Caltech (0.0574), MIT (0.0122), and Johns Hopkins (0.0140), the edge densities are all low, which typically suggests that these networks are sparse. In sparse networks, most node pairs are not directly connected to each other.

Global Clustering Coefficient:

The global clustering coefficient measures the overall tendency for nodes to cluster together.

Caltech has the highest global clustering coefficient (0.291), followed by Johns Hopkins (0.193) and MIT (0.180). Higher values indicate a greater prevalence of connected triples, suggesting a tendency for forming cliques or tightly knit groups within the network.

Mean Local Clustering Coefficient:

The mean local clustering coefficient is the average of the local clustering coefficients of all the nodes in the graph, indicating the average likelihood that two neighbors of a node are also neighbors of each other.

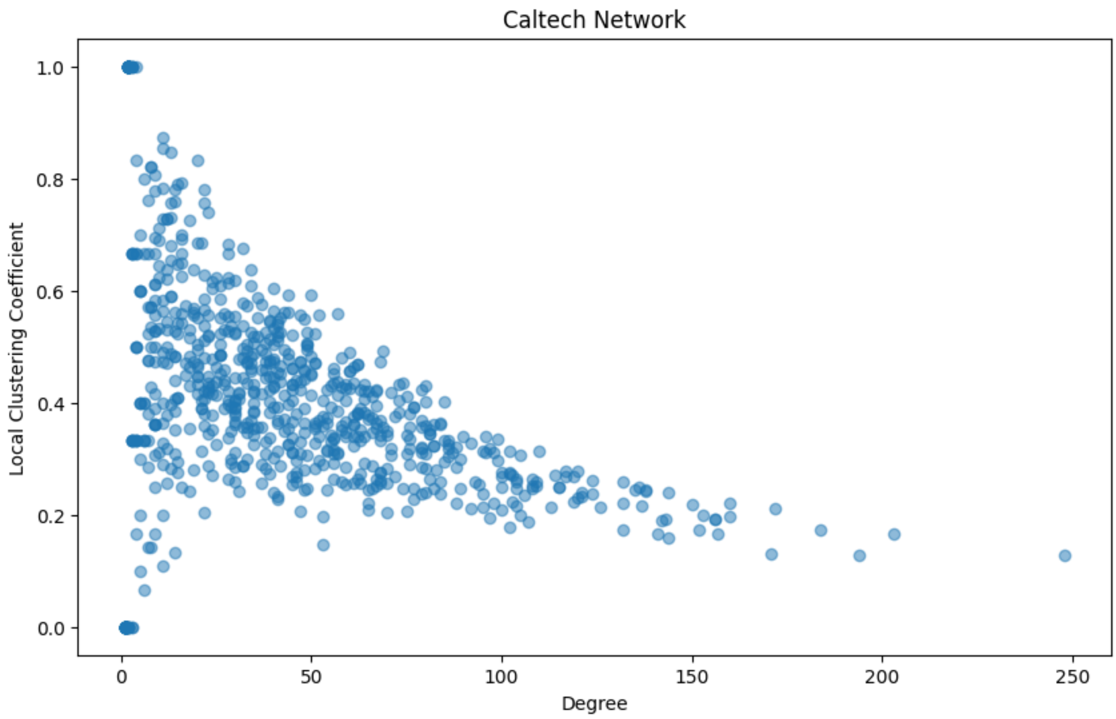
Caltech, with a mean local clustering coefficient of 0.409, suggests a high likelihood of nodes forming tightly knit groups, which is consistent with its higher global clustering coefficient. MIT and Johns Hopkins have lower mean local clustering coefficients (0.272 and 0.269, respectively), indicating less clustering locally but still significant when compared to random networks.

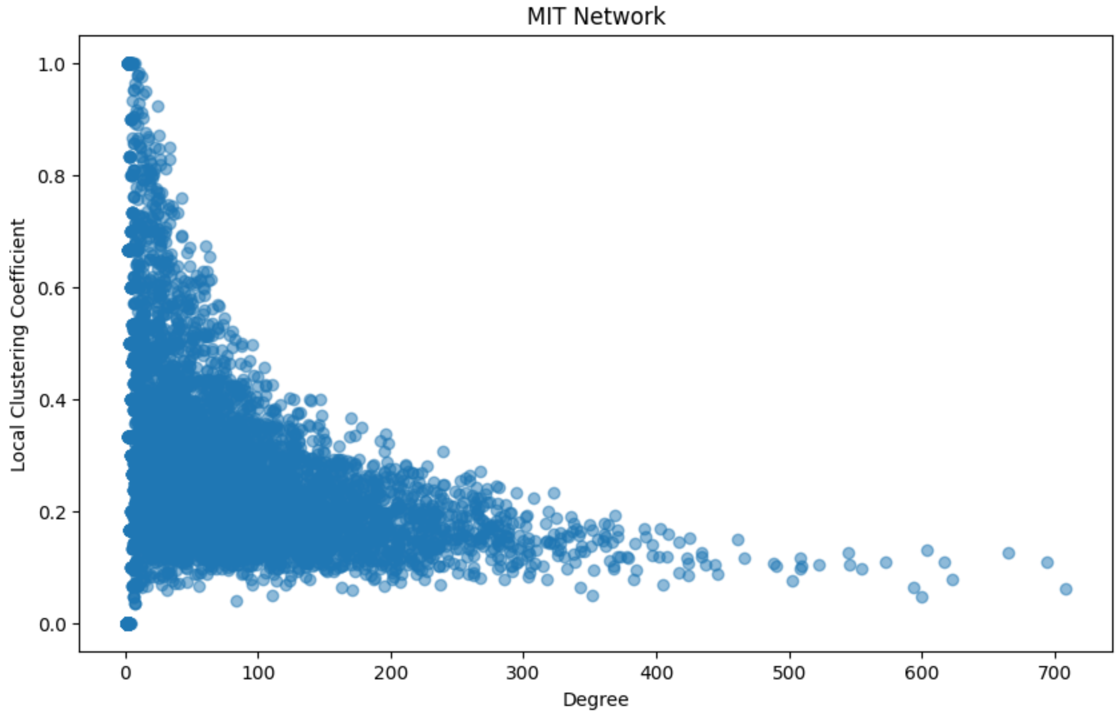
Caltech: The network topology suggests a moderately high degree of clustering with a sparse edge distribution. This could mean that while there are groups within the network that are highly interconnected, most of the nodes have few connections to other nodes outside their local cluster.

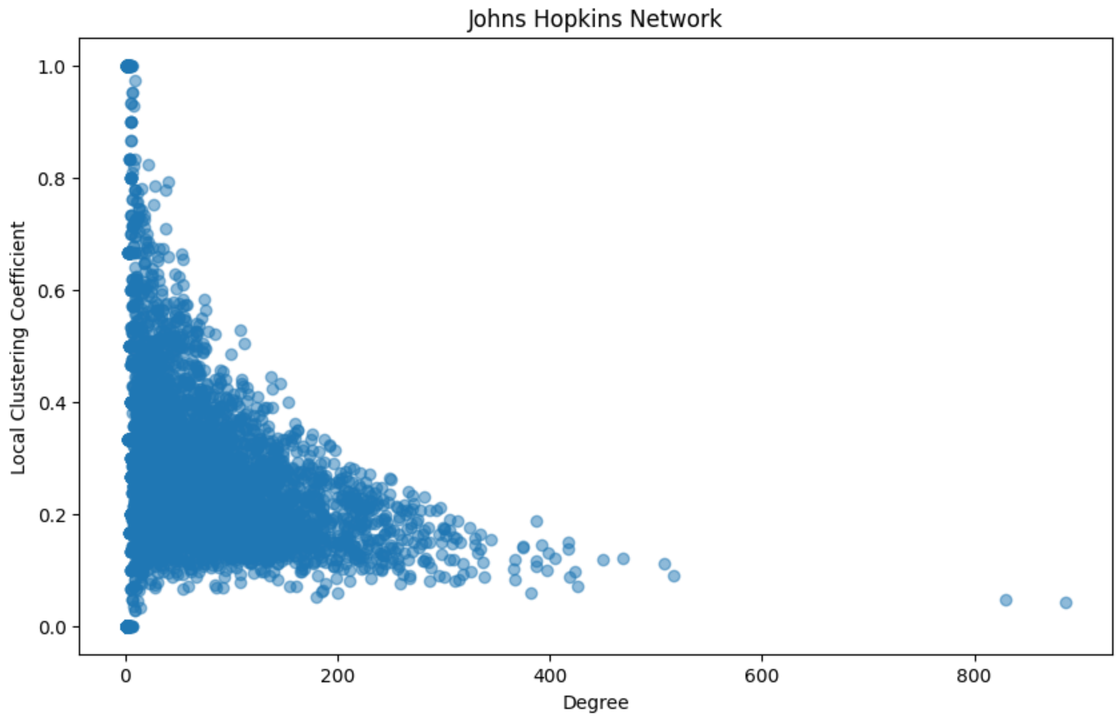
MIT and Johns Hopkins: Both have lower global and local clustering coefficients compared to Caltech, with MIT having a particularly sparse edge density. This suggests that these networks might have a more tree-like or star-like structure with less prevalence of tightly knit communities compared to Caltech.

General Topology: The combination of sparsity and non-trivial clustering coefficients indicates that the networks are neither completely random nor completely regular. They could potentially be small-world networks, where most nodes are not neighbors but can be reached from every other by a small number of steps because of the presence of clustering and the occasional bridge-like connections between clusters.

3. (c) For each network, also draw a scatter plot of the degree versus local clustering coefficient. Based on these calculations as well as your previous ones, are you able to draw any conclusions about any similarities or differences between the tree networks? What other observations can you make?







Caltech Network:
There is a wide spread in the local clustering coefficient for nodes with a lower degree, which includes some nodes with very high clustering.
For nodes with higher degrees, the local clustering coefficient tends to be lower, suggesting that more connected nodes are less likely to be part of tightly-knit cliques.

MIT Network:

The scatter plot indicates a strong negative correlation between degree and local clustering coefficient. Nodes with fewer connections tend to have higher clustering, and this trend is more pronounced compared to the Caltech network.
This could suggest that the MIT network is possibly more hierarchical or has a more pronounced core-periphery structure.

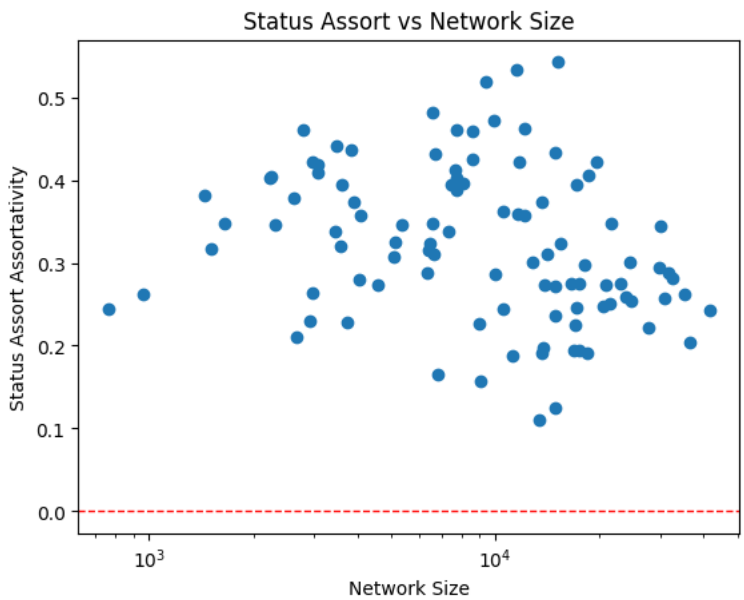
Johns Hopkins Network:
Similar to the MIT network, there is a negative correlation between degree and local clustering coefficient, but the spread of clustering coefficients for nodes with lower degrees is less than in the Caltech network.
The decline in clustering coefficient with increasing degree is quite steep, indicating a strong tendency towards non-clique-like connectivity among higher degree nodes.
Similarities and Differences:

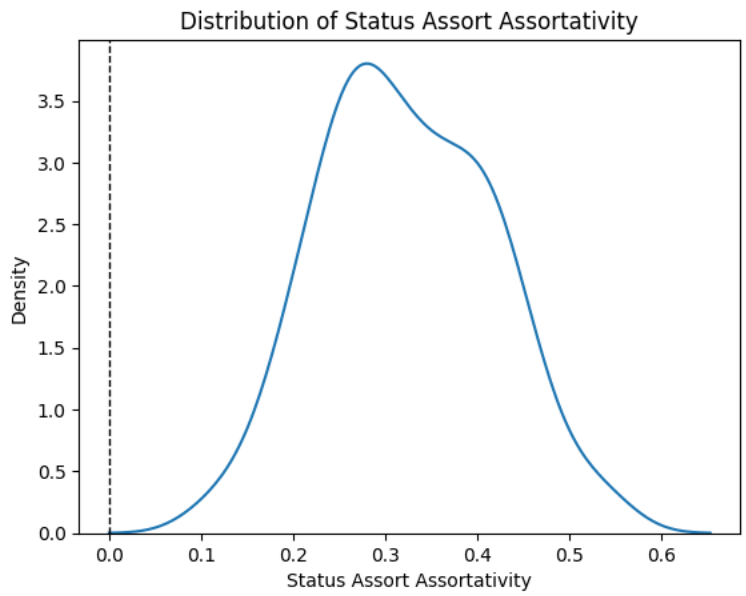
Similarities:
All three networks exhibit a negative correlation between the degree of a node and its local clustering coefficient, which is typical in many real-world networks where high-degree nodes act as bridges between different parts of the network and therefore have lower clustering.
Each network seems to display some level of hierarchical organization, with nodes of lower degree having higher potential for local clustering.
Differences:

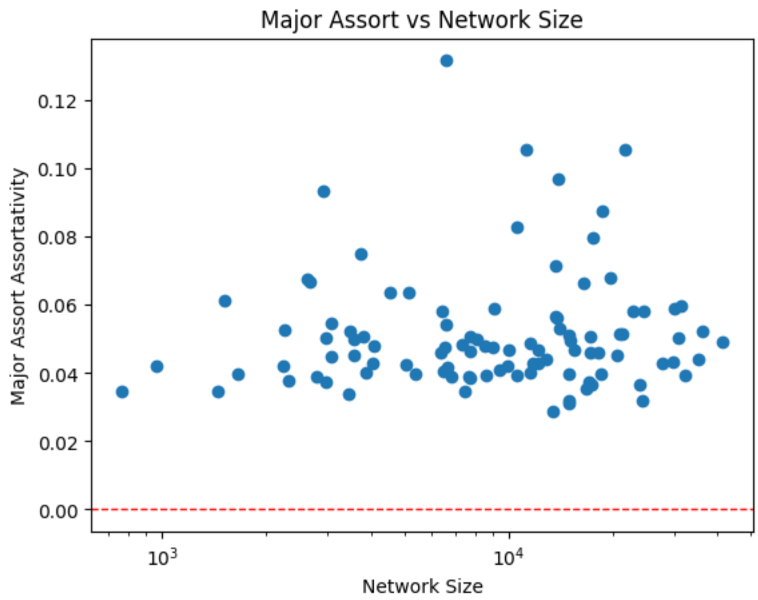
The Caltech network appears to have a higher variability in local clustering coefficients for nodes of lower degree, suggesting a more diverse connectivity pattern.
The MIT network shows a very clear and strong negative correlation, which might indicate a more strict hierarchical structure.
The Johns Hopkins network seems to have a smoother gradient in the decline of local clustering coefficient with degree, which might suggest a more uniform distribution of connectivity across different node degrees.

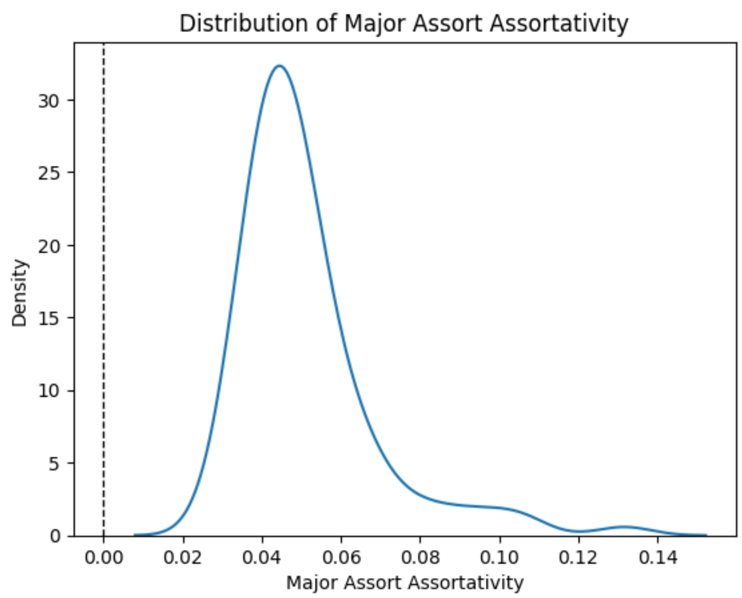
Additional Observations:
Degree of Hubs: The scatter plots indicate that the highest-degree nodes (hubs) are not the most clustered, which is typical for scale-free networks where hubs tend to connect different parts of the network.
Potential for Modularity: The presence of nodes with a high local clustering coefficient at the lower degrees may indicate the potential for modularity within the network, with possible community structures or groups.

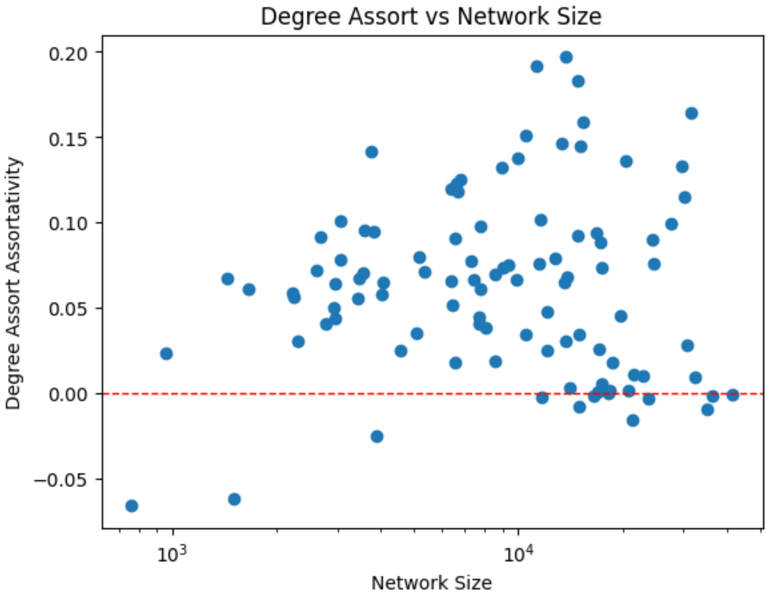
Question 3: Assortativity Analysis with the Facebook100 Dataset
In this question we expect you will compute the assortativity on a large set of graphs (if possible all the graphs).
(a) Of the FB100 networks, investigate the assortativity patterns for five vertex attributes: (i) student/faculty status, (ii) major, (iii) vertex degree, and (iiii) dorm, (iiiii) gender. Treat these networks as simple graphs in your analysis.
For each vertex attribute, make a scatter plot showing the assortativity versus network size n, with log-linear axes for all 100 networks, and a histogram or density plot showing the distribution of assortativity values. In both figures, include a line indicating no assortativity. Briefly discuss the degree to which vertices do or do not exhibit assortative mixing on each attribute, and specu- late about what kind of processes or tendencies in the formation of Facebook friendships might produce this kind of pattern. For example, below are figures for assortativity by gender on these networks. The distribution of points spans the line of no assortativity, with some values nearly as far below 0 as others are above 0. However, the gender attributes do appear to be slightly assortative in these social networks: although all values are within 6% in either direction of 0, the mean assortativity is 0.02, which is slightly above 0. This suggests a slight amount of homophily by gender (like links with like) in the way people friend each other on Facebook, although the tendency is very weak. In some schools, we see a slight tendency for heterophily (like links with dislike), as one might expect if the networks reflected heteronormative dating relationships.

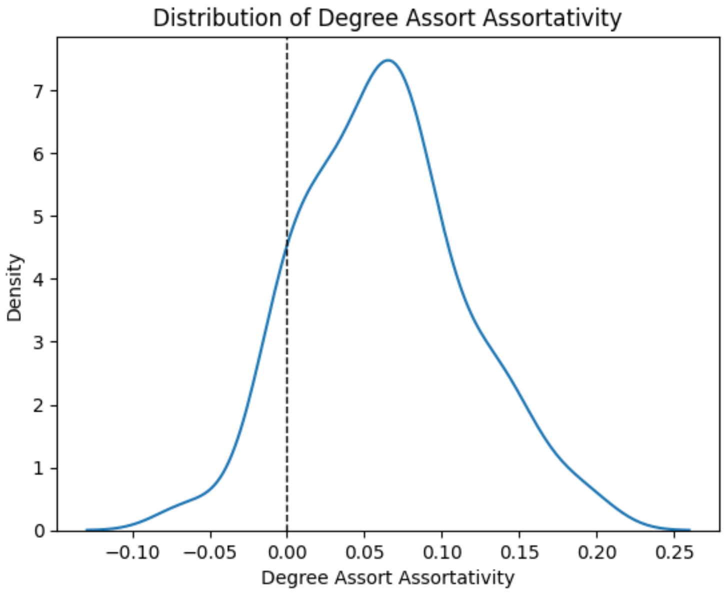


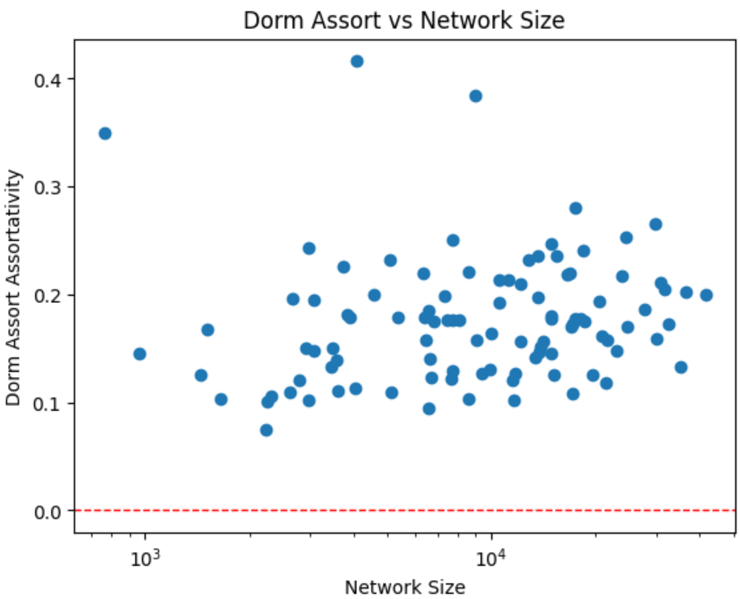

Distribution of Status Assort Assortativity
Density
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Status Assort Assortativity

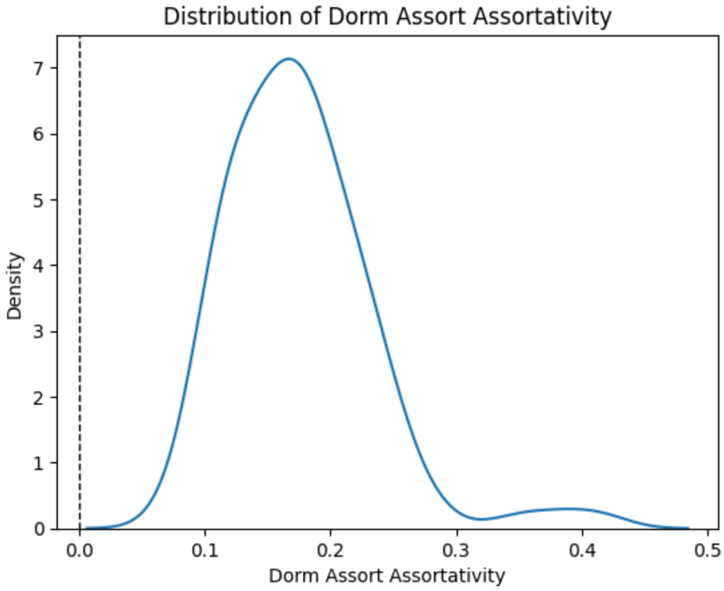

Major Assort vs Network Size
Major Assort Assortativity
0.12
0.10
0.08
0.06
0.04
0.02
0.00
10^3
10^4
Network Size

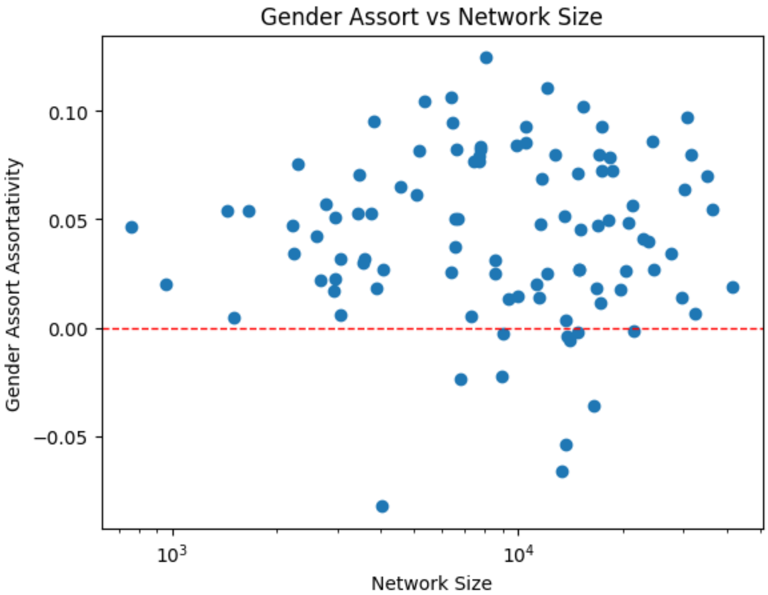

Distribution of Major Assort Assortativity
Density
30
25
20
15
10
5
0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Major Assort Assortativity

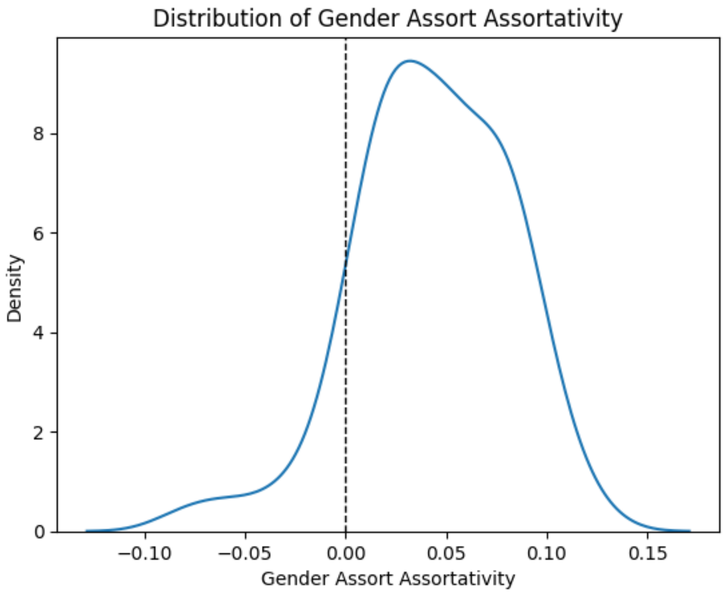

Degree Assort vs Network Size
Degree Assort Assortativity
Network Size
0.20
0.15
0.10
0.05
0.00
−0.05
10^3
10^4


Distribution of Degree Assort Assortativity
Density
Degree Assort Assortativity
0
1
2
3
4
5
6
7
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
0.25


Dorm Assort vs Network Size
Dorm Assort Assortativity
Network Size
0.4
0.3
0.2
0.1
0.0
10^3
10^4


Distribution of Dorm Assort Assortativity
Density
Dorm Assort Assortativity
0.0
0.1
0.2
0.3
0.4
0.5
0
1
2
3
4
5
6
7


Gender Assort vs Network Size
Gender Assort Assortativity
Network Size
10^3
10^4
−0.05
0.00
0.05
0.10


Distribution of Gender Assort Assortativity
Density
Gender Assort Assortativity
−0.10
−0.05
0.00
0.05
0.10
0.15
0
2
4
6
8

Student/Faculty Status:
Assortativity appears positive across networks, indicating a tendency for students to connect with students and faculty with faculty. This likely reflects social circles within universities, where interactions are more frequent within these distinct groups.
Major:
The assortativity values are generally low, but positive, suggesting slight homophily based on major. Students may have more interactions within their own fields of study, leading to a greater likelihood of forming friendships.
Vertex Degree:
There is a range of assortativity values for vertex degree, with some networks showing negative assortativity, which could indicate a tendency for well-connected individuals to befriend less connected individuals, possibly due to the bridging role that popular nodes play in expanding the network.
Dorm:
The assortativity by dorm shows significant positive values, reflecting strong homophily. This is expected since dormitory residence facilitates close and frequent interactions, leading to the formation of friendships.
Gender:
There is slight positive assortativity by gender, indicating a weak tendency for individuals of the same gender to befriend each other more than expected by chance. This could be due to shared interests or social norms guiding friendship formation. However, the presence of some networks with negative assortativity by gender might suggest influences of romantic relationships, where individuals of different genders are connecting.
General Observations:
The scatter plots with log-linear axes allow for the visualization of trends across networks of varying sizes, and the density plots provide an overview of how common certain assortativity values are.
The presence of assortativity or disassortativity in these attributes indicates the influence of social processes such as selection (people forming ties with similar others), influence (people becoming more similar to their friends over time), and the structural organization of the university environment on the formation of social ties.
The overall positive assortativity for most attributes except vertex degree suggests that homophily is a common tendency in these networks, with individuals connecting with others who are similar to themselves in status, major, and dorm residence.
The variation in assortativity across different networks suggests that there may be unique cultural or institutional factors at play in different universities that affect how friendships are formed.

Question 4: Link prediction

In this question we expect you will compute the link prediction algorithms on a large set of graphs (> 10).

(a) Read the following documents [4].

(b) Implement the following link prediction metrics: common neighbors, jaccard, Adamic/Adar. We use the scikit-learn2 API as an example for our implementation of the link prediction metrics. Please use the implementation (in listing. 1) as an example. Your implementation should inherit from the class LinkPrediction defined in listing. 1. You should implement yourself the given metrics, don't used the ones defined in Networkx

```
import networkx as nx
import numpy as np
from abc import ABC, abstractmethod
import progressbar
class LinkPrediction(ABC):
    def __init__(self, graph):
        self.graph = graph

    @abstractmethod
    def fit(self, u, v):
        raise NotImplementedError("Fit must be implemented")

class CommonNeighbors(LinkPrediction):
    def __init__(self, graph):
        super(CommonNeighbors , self).__init__(graph)
    def fit(self, u, v):
        neighbors_u = set(self.graph.neighbors(u))
        neighbors_v = set(self.graph.neighbors(v))
        return len(neighbors_u & neighbors_v)

class JaccardCoefficient(LinkPrediction):
    def __init__(self, graph):
        super(JaccardCoefficient , self).__init__(graph)
    def fit(self, u, v):
        neighbors_u = set(self.graph.neighbors(u))
        neighbors_v = set(self.graph.neighbors(v))
        union = neighbors_u | neighbors_v
        intersection = neighbors_u & neighbors_v
        return len(intersection) / len(union) if union else 0

class AdamicAdarIndex(LinkPrediction):
    def __init__(self, graph):
        super(AdamicAdarIndex , self).__init__(graph)
    def fit(self, u, v):
        neighbors_u = set(self.graph.neighbors(u))
        neighbors_v = set(self.graph.neighbors(v))
        common_neighbors = neighbors_u & neighbors_v
        return sum(1 / np.log(len(list(self.graph.neighbors(w))))
                   for w in common_neighbors if self.graph.degree(w) > 1)
```

(c) Evaluating a link predictor:
1. Select graph Gfb(V,E) in the Facebook100 dataset
2. randomly remove a given fraction f ∈ [0.05, 0.1, 0.15, 0.2] of edges Eremoved from the original graph Gfb.
3. For each node pair in the graph |V |×|V |, for each node pair compute the link predictor metrics of interest p, these are the predicted ”friendship” Epredict.
4. Sort in decreasing order of confidence as a function p from the node pair
Epredict and then we take the first k pairs of nodes E(top@k). predict Compute the size of the intersection between the edge set of removed edges
and the edge set of predicted node |Eremoved ∩ E(top@k)| . Then compute the predict top@k, recall@k and precision@k (for k = 50, 100, 200, · · · , 400) using the k best scored edges provided by link predictor algorithm (see [5] for more information). Where the top@k predictive rate is the percentage of correctly classified positive samples among the top k instances in the ranking produced by a link predictor P.

$$Precision = \frac{|TP|}{|TP| + |FP|}$$

$$Recall = \frac{|TP|}{|TP| + |FN|}$$

negatives, FP stands for false positives, and FN stands for false negatives.

```
g_list = g_all_list

fractions = [0.05, 0.1, 0.15, 0.2]

methods = ['CommonNeighbors','JaccardCoefficient','AdamicAdarIndex']

k = [50,100,200,400]


results = []


for g in g_list:

    for fraction in fractions:

        E_removed, removed_edges = remove_random_edges(g, fraction)


        for method in methods:

            pre = calculate_link_prediction_metrics(E_removed, g)


            for current_k in k:
```

```
                predictions = np.array(pre[method])


                top_k_indices = np.argsort(predictions)[-
current_k:][::-1]
                top_k_node_pairs = [pre['node_pairs'][i] for i in
top_k_indices]


                precision, recall, topk =
compute_precision_recall_top_k(top_k_node_pairs, removed_edges,
current_k)
                print(fraction,method, current_k )


                results.append([(precision, recall, topk), pre])
```

Len(fractions) * len(methods) * len(k) = 3*4*4 = 48 (times/graph)
Total results = 48 times/graph * 15 graph = 720 times
One of the results:

```
[Graph with 762 nodes and 16651 edges] [================================] 100%
0.05 CommonNeighbors 50

precision:  0.2
recall:  0.01201923076923077
topk:  10
0.05 CommonNeighbors 100

precision:  0.13
recall:  0.015625
topk:  13
0.05 CommonNeighbors 200

precision:  0.11
recall:  0.026442307692307692
topk:  22
0.05 CommonNeighbors 400

precision:  0.0925
recall:  0.04447115384615385
topk:  37
[Graph with 762 nodes and 16651 edges] [================================] 100%
0.05 JaccardCoefficient 50

precision:  0.12
recall:  0.007211538461538462
topk:  6
0.05 JaccardCoefficient 100

precision:  0.12
recall:  0.014423076923076924
topk:  12
0.05 JaccardCoefficient 200

precision:  0.11
```

```
recall:  0.026442307692307692
topk:  22
0.05 JaccardCoefficient 400

precision:  0.11
recall:  0.052884615384615384
topk:  44
[Graph with 762 nodes and 16651 edges] [==================================] 100%
0.05 AdamicAdarIndex 50

precision:  0.18
recall:  0.010817307692307692
topk:  9
0.05 AdamicAdarIndex 100

precision:  0.13
recall:  0.015625
topk:  13
0.05 AdamicAdarIndex 200

precision:  0.115
recall:  0.027644230769230768
topk:  23
0.05 AdamicAdarIndex 400

precision:  0.105
recall:  0.05048076923076923
topk:  42
[Graph with 762 nodes and 16651 edges] [==================================] 100%
[Graph with 762 nodes and 16651 edges] [                                  ]   0%
0.1 CommonNeighbors 50

precision:  0.28
recall:  0.008408408408408409
topk:  14
0.1 CommonNeighbors 100

precision:  0.3
recall:  0.018018018018018018
topk:  30
0.1 CommonNeighbors 200

precision:  0.275
recall:  0.03303303303303303
topk:  55
0.1 CommonNeighbors 400

precision:  0.215
recall:  0.05165165165165165
topk:  86
[Graph with 762 nodes and 16651 edges] [==================================] 100%
0.1 JaccardCoefficient 50

precision:  0.22
recall:  0.006606606606606606
topk:  11
0.1 JaccardCoefficient 100

precision:  0.2
```

```
recall:  0.012012012012012012
topk:  20
0.1 JaccardCoefficient 200

precision:  0.175
recall:  0.021021021021021023
topk:  35
0.1 JaccardCoefficient 400

precision:  0.1825
recall:  0.04384384384384384
topk:  73
[Graph with 762 nodes and 16651 edges] [==================================] 100%
0.1 AdamicAdarIndex 50

precision:  0.32
recall:  0.00960960960960961
topk:  16
0.1 AdamicAdarIndex 100

precision:  0.31
recall:  0.018618618618618618
topk:  31
0.1 AdamicAdarIndex 200

precision:  0.285
recall:  0.03423423423423423
topk:  57
0.1 AdamicAdarIndex 400

precision:  0.2225
recall:  0.05345345345345345
topk:  89
[Graph with 762 nodes and 16651 edges] [==================================] 100%
[Graph with 762 nodes and 16651 edges] [                                  ]   0%
0.15 CommonNeighbors 50

precision:  0.26
recall:  0.005206247496996396
topk:  13
0.15 CommonNeighbors 100

precision:  0.28
recall:  0.011213456147376852
topk:  28
0.15 CommonNeighbors 200

precision:  0.245
recall:  0.01962354825790949
topk:  49
0.15 CommonNeighbors 400

precision:  0.2275
recall:  0.03644373247897477
topk:  91
[Graph with 762 nodes and 16651 edges] [==================================] 100%
0.15 JaccardCoefficient 50

precision:  0.18
```

```
recall:  0.003604325190228274
topk:  9
0.15 JaccardCoefficient 100

precision:  0.21
recall:  0.00841009211053264
topk:  21
0.15 JaccardCoefficient 200

precision:  0.185
recall:  0.014817781337605127
topk:  37
0.15 JaccardCoefficient 400

precision:  0.21
recall:  0.03364036844213056
topk:  84
[Graph with 762 nodes and 16651 edges] [===================================] 100%
0.15 AdamicAdarIndex 50

precision:  0.24
recall:  0.004805766920304365
topk:  12
0.15 AdamicAdarIndex 100

precision:  0.27
recall:  0.010812975570684821
topk:  27
0.15 AdamicAdarIndex 200

precision:  0.26
recall:  0.020824989987985584
topk:  52
0.15 AdamicAdarIndex 400

precision:  0.23
recall:  0.0368442130556668
topk:  92
[Graph with 762 nodes and 16651 edges] [===================================] 100%
[Graph with 762 nodes and 16651 edges] [                                   ]   0%
0.2 CommonNeighbors 50

precision:  0.36
recall:  0.005405405405405406
topk:  18
0.2 CommonNeighbors 100

precision:  0.32
recall:  0.00960960960960961
topk:  32
0.2 CommonNeighbors 200

precision:  0.305
recall:  0.01831831831831832
topk:  61
0.2 CommonNeighbors 400

precision:  0.2575
recall:  0.030930930930930932
```

```
topk:  103
[Graph with 762 nodes and 16651 edges] [================================] 100%
0.2 JaccardCoefficient 50

precision:  0.08
recall:  0.0012012012012012011
topk:  4
0.2 JaccardCoefficient 100

precision:  0.15
recall:  0.0045045045045045045
topk:  15
0.2 JaccardCoefficient 200

precision:  0.17
recall:  0.01021021021021021
topk:  34
0.2 JaccardCoefficient 400

precision:  0.2175
recall:  0.026126126126126126
topk:  87
[Graph with 762 nodes and 16651 edges] [================================] 100%
0.2 AdamicAdarIndex 50

precision:  0.32
recall:  0.004804804804804805
topk:  16
0.2 AdamicAdarIndex 100

precision:  0.33
recall:  0.00990990990990991
topk:  33
0.2 AdamicAdarIndex 200

precision:  0.305
recall:  0.01831831831831832
topk:  61
0.2 AdamicAdarIndex 400

precision:  0.265
recall:  0.03183183183183183
topk:  106
```

Algorithm Performance:
Across different thresholds and top-k values, the Adamic-Adar Index and Common Neighbors tend to yield higher precision compared to the Jaccard Coefficient. This indicates that for this particular network, the weighting scheme of Adamic-Adar and the simplicity of Common Neighbors are more suited for predicting links.
Impact of Top-k:
For all three algorithms, as the top-k increases, there is a trend of decreasing precision and increasing recall. This suggests that while the algorithms are capable of identifying more true positives as they are allowed to make more predictions (higher top-k), the rate of false positives also increases, hence the drop in precision.
Recall Increases with Top-k:
Recall tends to increase as the top-k value rises. This is expected because as you predict more links (a higher top-k), the chance of capturing the actual removed links increases.

Precision at Different Thresholds:
At lower thresholds (0.05, 0.1), precision is generally higher, indicating that the algorithms perform better when fewer links are removed from the network. As the threshold increases (0.15, 0.2), the precision tends to decrease, suggesting that the task becomes more difficult as more links are removed.
General Observations:
The precision values are never exceedingly high, which might imply that the network has a complex structure that simple similarity indices struggle to fully capture.
The recall values are generally low across all configurations, indicating that a large portion of actual links are not being captured within the top-k predictions. This could be due to the network's complexity or the inherent limitations of the algorithms used.
Conclusions:
The link prediction algorithms tested here have room for improvement, especially in increasing recall without significantly sacrificing precision.
Considering the moderate success of the Adamic-Adar and Common Neighbors algorithms, a combination or ensemble of these methods with other techniques could potentially improve overall performance.
The findings suggest that while simple local similarity indices can provide some insight into link formation, they may not be sufficient to capture all the nuances in a complex network such as the one analyzed. Additional features, possibly incorporating node attributes or global network properties, may be required to enhance the prediction capabilities.

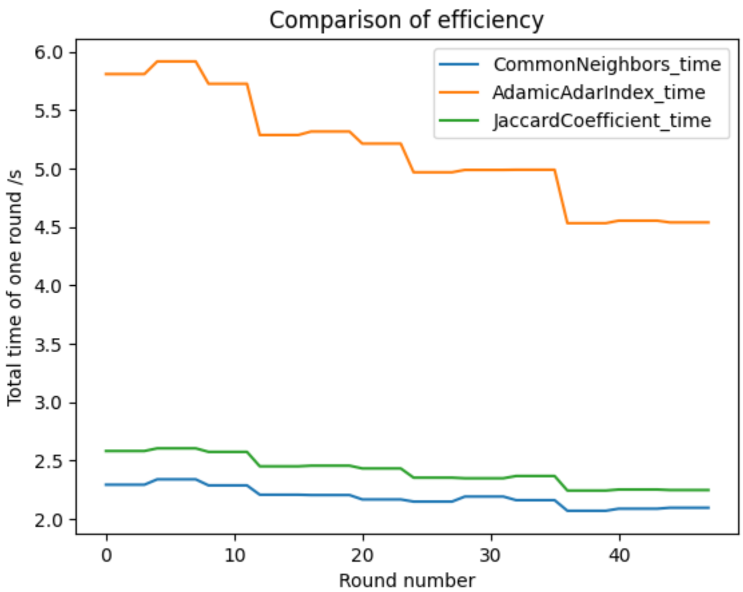
(d) Choose a couple of graphs in the facebook100 dataset run and evaluate each link predictor on them, and conclude on the efficiency of the following metrics: common neighbors, jaccard, Adamic/Adar.

```
def calculate_link_prediction_metrics(graph, graph_name):
    node_pairs = list(nx.non_edges(graph))
    results = {
        'node_pairs': [],
        'CommonNeighbors': [],
        'JaccardCoefficient': [],
        'AdamicAdarIndex': [],
        'CommonNeighbors_time': [],
        'JaccardCoefficient_time': [],
        'AdamicAdarIndex_time': []
    }

    bar = progressbar.ProgressBar(maxval=len(node_pairs),
                                  widgets=[progressbar.Bar('=',
f'[{graph_name}] [', ']'), ' ', progressbar.Percentage()])
    bar.start()

    for i, (u, v) in enumerate(node_pairs):

        start_time = time.time()
        cn = CommonNeighbors(graph)
        cn_result = cn.fit(u, v)
        end_time = time.time()
```

```
        cn_time = end_time - start_time


        start_time = time.time()
        jc = JaccardCoefficient(graph)
        jc_result = jc.fit(u, v)
        end_time = time.time()
        jc_time = end_time - start_time


        start_time = time.time()
        aa = AdamicAdarIndex(graph)
        aa_result = aa.fit(u, v)
        end_time = time.time()
        aa_time = end_time - start_time


        results['node_pairs'].append((u, v))
        results['CommonNeighbors'].append(cn_result)
        results['JaccardCoefficient'].append(jc_result)
        results['AdamicAdarIndex'].append(aa_result)
        results['CommonNeighbors_time'].append(cn_time)
        results['JaccardCoefficient_time'].append(jc_time)
        results['AdamicAdarIndex_time'].append(aa_time)

        bar.update(i+1)

    bar.finish()
    return results
```

```
c_total = []
a_total = []
j_total = []
for i in range(len(results)):
  c = sum(results[i][-1]['CommonNeighbors_time'])
  a = sum(results[i][-1]['AdamicAdarIndex_time'])
  j = sum(results[i][-1]['JaccardCoefficient_time'])
  c_total.append(c)
  a_total.append(a)
  j_total.append(j)


plt.plot(c_total, label='CommonNeighbors_time')
plt.plot(a_total, label='AdamicAdarIndex_time')
plt.plot(j_total, label='JaccardCoefficient_time')
```

```
plt.legend()


plt.title('Comparison of efficiency')
plt.xlabel('Round number')
plt.ylabel('Total time of one round /s')


plt.show()
```



Key observations from the graph:

Adamic-Adar Index Time:
The Adamic-Adar Index (a_total) shows a significantly higher computation time compared to the other two algorithms. This could be due to the more complex calculations involved in the Adamic-Adar Index, which takes into account the logarithm of the degree of common neighbors.
Jaccard Coefficient Time:
The Jaccard Coefficient (j_total) also shows variability in computation time but remains consistently lower than the Adamic-Adar Index. Its computation involves the ratio of the size of the intersection to the size of the union of the sets of neighbors, which might be less computationally intensive than the Adamic-Adar Index.
Common Neighbors Time:
The Common Neighbors algorithm (c_total) appears to have the lowest and most stable computation time across all rounds. This is expected because it relies on a simple count of common neighbors between two nodes, which is a straightforward calculation.
Efficiency Over Rounds:
For the Common Neighbors and Jaccard Coefficient, the computation time is relatively stable across rounds, indicating consistent performance.
The Adamic-Adar Index time decreases after the initial rounds and then stabilizes. The initial higher values could be due to startup overhead or the algorithm warming up cache and memory locations, which is optimized in later rounds.
Conclusions:

The Common Neighbors algorithm is the most efficient in terms of computation time, making it a good choice for larger networks or when computation resources are limited.
The Adamic-Adar Index, while potentially offering better precision in link prediction as seen in previous data, requires significantly more computation time and would benefit from optimization if it is to be used in time-sensitive applications.
The Jaccard Coefficient offers a middle ground in terms of computation time but may not provide as much precision in link prediction as the Adamic-Adar Index.
The choice of algorithm should consider both prediction performance and computational efficiency, depending on the specific requirements of the task at hand.

Question 5: Find missing labels with the label propagation algorithms
In this question we expect you will compute the label propagation algorithm on a large set of graphs (> 10). We studied in class two algorithms with the name ”label propagation” that have different objective, choose wisely the one to implement.
(a) Read the following document [6].
(b) Implement in python the label propagation algorithm [6], please consider pyt-orch3 and networkx4 for the development of your algorithm.

```
import torch
import torch.nn.functional as F
import networkx as nx
import numpy as np
from scipy import sparse

def label_propagation(graph, attribute, alpha=0.1, max_iter=1000,
tol=1e-3, device='cuda'):

    is_labeled = {node: data.get(attribute) is not None for node, data
in graph.nodes(data=True)}


    labels = {node: data[attribute] for node, data in
graph.nodes(data=True) if is_labeled[node]}


    unique_labels = list(set(labels.values()))
    label_mapping = {label: index for index, label in
enumerate(unique_labels)}


    num_nodes = graph.number_of_nodes()
    num_labels = len(unique_labels)
    labels_pred = torch.zeros(num_nodes, num_labels, device=device)


    for node, label in labels.items():
        node_index = list(graph.nodes()).index(node)
        labels_pred[node_index, label_mapping[label]] = 1


    A = nx.adjacency_matrix(graph, nodelist=graph.nodes())
    D = sparse.diags(1.0 / np.array(A.sum(axis=1)).flatten(), 0)
    T = D.dot(A)


    T_coo = T.tocoo()
    values = T_coo.data
    indices = np.vstack((T_coo.row, T_coo.col))
```

```
    i = torch.LongTensor(indices).to(device)
    v = torch.FloatTensor(values).to(device)
    shape = T_coo.shape
    T = torch.sparse.FloatTensor(i, v, torch.Size(shape)).to(device)



    F = labels_pred.clone()
    for iteration in range(max_iter):
        F_prev = F.clone()
        F = alpha * torch.sparse.mm(T, F) + (1 - alpha) * labels_pred

        if torch.norm(F - F_prev) < tol:
            break


    labels_pred = torch.argmax(F, dim=1).cpu().numpy()
    predicted_labels = {node: unique_labels[labels_pred[i]] for i, node
in enumerate(graph.nodes()) if not is_labeled[node]}

    return predicted_labels
```

(c) Choose a network from The Facebook100 dataset and randomly select 10%, 20%, and 30% of of the node attributes of the network to be removed. Use the label propagation algorithm you implemented to recover the missing attributes.
Perform this operation for "gender".

```
Processing graph: Caltech36.gml
Processing graph: Reed98.gml
Processing graph: Haverford76.gml
Processing graph: Simmons81.gml
Processing graph: Swarthmore42.gml
Processing graph: Amherst41.gml
Processing graph: Bowdoin47.gml
Processing graph: Hamilton46.gml
Processing graph: Trinity100.gml
Processing graph: USFCA72.gml
Processing graph: Williams40.gml
Processing graph: Oberlin44.gml
Processing graph: Wellesley22.gml
Processing graph: Smith60.gml
Processing graph: Vassar85.gml
```

```
Accuracy of the label propagation algorithm
Attribute 0.1       0.2       0.3
Caltech36.gml
dorm      0.684     0.697     0.759
major_index0.118     0.217     0.219
gender    0.592     0.559     0.610

Reed98.gml
dorm      0.625     0.630     0.615
```

```
major_index0.260      0.286      0.257
gender     0.552      0.521      0.552

Haverford76.gml
dorm       0.583      0.536      0.510
major_index0.347      0.298      0.316
gender     0.597      0.640      0.605

Simmons81.gml
dorm       0.543      0.523      0.512
major_index0.272      0.278      0.280
gender     0.921      0.917      0.945

Swarthmore42.gml
dorm       0.509      0.523      0.517
major_index0.309      0.266      0.268
gender     0.564      0.577      0.592

Amherst41.gml
dorm       0.417      0.459      0.452
major_index0.274      0.293      0.266
gender     0.484      0.548      0.567

Bowdoin47.gml
dorm       0.440      0.431      0.458
major_index0.284      0.289      0.286
gender     0.538      0.562      0.553

Hamilton46.gml
dorm       0.437      0.448      0.430
major_index0.255      0.227      0.255
gender     0.606      0.634      0.610

Trinity100.gml
dorm       0.452      0.441      0.448
major_index0.375      0.322      0.305
gender     0.548      0.569      0.587

USFCA72.gml
dorm       0.640      0.635      0.618
major_index0.348      0.331      0.326
gender     0.603      0.624      0.630

Williams40.gml
dorm       0.446      0.476      0.523
major_index0.291      0.264      0.273
gender     0.601      0.592      0.579

Oberlin44.gml
dorm       0.545      0.550      0.565
major_index0.281      0.291      0.283
gender     0.565      0.555      0.508

Wellesley22.gml
dorm       0.451      0.497      0.489
major_index0.286      0.219      0.259
gender     0.879      0.902      0.901
```

```
Smith60.gml
dorm         0.680      0.684      0.691
major_index0.273      0.271      0.240
gender      0.886      0.862      0.870
```

```
Vassar85.gml
dorm         0.627      0.615      0.680
major_index0.320      0.336      0.307
gender      0.516      0.520      0.511
```

(d) For each case of the following percentage of missing attributes: 10%, 20% and 30% and for each of the following attributes: the ”dorm”, ”major”, ”gender” show the mean absolute error and accuracy score (as defined in eq. 1) of the label propagation algorithm as in the example provided in Table 1 for the Duke University Facebook network. Note we can use the formula eq. 1 for computing the accuracy. However, a better approach would have been to compute the F1-score [7].

```python
def calculate_weighted_f1_score(original_results):
    f1_scores = {}

    for graph_name, graph_data in original_results.items():
        f1_scores[graph_name] = {}
        for attribute, values in graph_data.items():

            f1_scores_attribute = []
            for true_values, predictions in values:

                y_true = list(true_values.values())
                y_pred = list(predictions.values())

                f1 = f1_score(y_true, y_pred, average='weighted')
                f1_scores_attribute.append(f1)

            f1_scores[graph_name][attribute] = f1_scores_attribute

    return f1_scores
f1_scores = calculate_weighted_f1_score(original_results)
```

```python
def print_f1_scores_table(f1_scores, fraction_of_nodes_list,
miss_attribute_list):
    print("Table: Weighted F1 Score of the label propagation
algorithm")
    header = "Attribute" + " " * (15 - len("Attribute"))
    for fraction in fraction_of_nodes_list:
        header += f"{fraction:<15}"
    print(header)

    for graph_name, graph_data in f1_scores.items():
        print(f"{graph_name}")
        for attribute in miss_attribute_list:
            attr_scores = graph_data.get(attribute, [])
```

```
            row = f"{attribute:<15}"
            for score in attr_scores:
                row += f"{score:.3f}{' ' * 10}"
            print(row)
        print()
```

```
Weighted F1 Score of the label propagation algorithm
Attribute      0.1            0.2            0.3
Caltech36.gml
dorm           0.146          0.131          0.130
major_index    0.030          0.069          0.071
gender         0.525          0.423          0.558

Reed98.gml
dorm           0.325          0.289          0.305
major_index    0.039          0.052          0.095
gender         0.454          0.415          0.441

Haverford76.gml
dorm           0.161          0.174          0.198
major_index    0.151          0.096          0.113
gender         0.387          0.444          0.457

Simmons81.gml
dorm           0.227          0.215          0.244
major_index    0.067          0.064          0.080
gender         0.882          0.881          0.919

Swarthmore42.gml
dorm           0.122          0.212          0.155
major_index    0.101          0.079          0.092
gender         0.462          0.423          0.441

Amherst41.gml
dorm           0.234          0.250          0.264
major_index    0.101          0.140          0.131
gender         0.428          0.393          0.451

Bowdoin47.gml
dorm           0.152          0.150          0.168
major_index    0.130          0.116          0.129
gender         0.407          0.451          0.396

Hamilton46.gml
dorm           0.137          0.163          0.149
major_index    0.106          0.103          0.102
gender         0.422          0.467          0.449

Trinity100.gml
dorm           0.162          0.146          0.155
major_index    0.129          0.120          0.117
gender         0.397          0.376          0.456

USFCA72.gml
dorm           0.271          0.270          0.300
major_index    0.122          0.106          0.109
gender         0.504          0.535          0.509
```

Williams40.gml			
dorm	0.180	0.158	0.189
major_index	0.131	0.100	0.115
gender	0.446	0.440	0.436

Oberlin44.gml			
dorm	0.259	0.267	0.261
major_index	0.050	0.073	0.061
gender	0.437	0.468	0.389

Wellesley22.gml			
dorm	0.153	0.177	0.150
major_index	0.069	0.080	0.081
gender	0.825	0.853	0.849

Smith60.gml			
dorm	0.061	0.043	0.046
major_index	0.062	0.069	0.064
gender	0.838	0.799	0.812

Vassar85.gml			
dorm	0.125	0.106	0.135
major_index	0.118	0.095	0.097
gender	0.504	0.450	0.441

(e) Conclude on the accuracy of the label propagation algorithm for different labels, could you explain why is there such difference in the accuracy between each type of label ?

Performance Across Different Labels:
Dorm: The F1 Scores for predicting dorm affiliation are generally low across all networks, which may indicate that dormitory assignments are less indicative of the types of connections that individuals make within these networks.
Major_Index: Similar to dorms, the F1 Scores for major_index are also relatively low, but slightly higher than dorms in some cases. This suggests some level of homophily based on major but not as strong as one might expect.
Gender: The F1 Scores for gender are consistently higher than for dorm and major_index. This suggests that gender may be a more significant factor in the formation of connections within these networks, which could reflect social tendencies or cultural norms in friendship formations.

Variation in Accuracy Between Labels:
The difference in accuracy between dorm, major_index, and gender is likely due to the different nature of these attributes and their relevance to social network formation. Gender may be a more fundamental social category that influences network formation more significantly than dorm or major, which might be more circumstantial or subject to change. Dorm and major are more specific and can be less consistent across individuals' university experience, while gender is a more general and socially significant category that affects a wide range of interactions.

Accuracy Trend With Removal Ratio:
For some networks, the accuracy increases as the removal ratio increases. This could happen if the label propagation algorithm performs better when there's less noise in the data. When

more labels are removed (up to a certain point), the algorithm might be better able to identify and propagate the most reliable signals, leading to improved accuracy.
Another possibility is that the removal of certain nodes may disrupt cliques or tightly-knit groups that were previously causing the algorithm to incorrectly label members of different communities.

Overall Accuracy:
The algorithm shows varying degrees of accuracy across different schools and different attributes. This variability can be attributed to the unique social structures and the prevalence of certain attributes within each network.
In networks where the F1 Score is particularly high for gender (e.g., Simmons81.gml, Wellesley22.gml, Smith60.gml), it might indicate a more pronounced gender-based grouping or the influence of gender in social organization within those schools.
The accuracy of the label propagation algorithm for different labels reflects the importance of those attributes in the social structure of the networks. Gender appears to be a strong factor in the formation of social ties across these networks, while dorm and major are less so. The improvement in accuracy with the increase in the removal ratio could be due to the reduction of noise and less complex structures for the algorithm to handle, which allows it to propagate labels more effectively. However, this trend may not hold consistently across all networks and attributes, as local social dynamics can significantly influence the results.

Question 6: Communities detection with the FB100 datasets
Formulate a research question about group formation among students in the FB100 dataset. To validate your hypothesis, use only a few universities and a community detection algorithm of your choice to extract the different groups of students. To help you formulate a research question, some of the following references might be useful [8, 9] and section 3.4 in [2].
(a) Formulate a research question about group formation in FB100 and explain your hypothesis.
Hypothesis:
The primary driving force behind group formation in social networks among students in the FB100 dataset is gender.
Reasons for the Hypothesis:

Higher Assortativity for Gender:
The data from the label propagation algorithm indicates higher F1 Scores for gender compared to other attributes like dorm and major. This suggests that gender may play a more significant role in how students form groups.

Social and Cultural Factors:
Gender is a fundamental social category that often guides social interactions and affiliations. It can influence friendship patterns, social circles, and community formation due to shared experiences, interests, and cultural norms.

Gender-Specific Groups and Societies:
Universities often have gender-specific groups, societies, and activities that could foster tighter networks among students of the same gender, influencing the overall structure of the social network.

Homophily in Social Networks:
Homophily, the tendency for individuals to associate with similar others, is a well-documented phenomenon in social networks. Given that gender is a distinct and recognizable attribute, it may serve as a strong basis for homophily in these university networks.

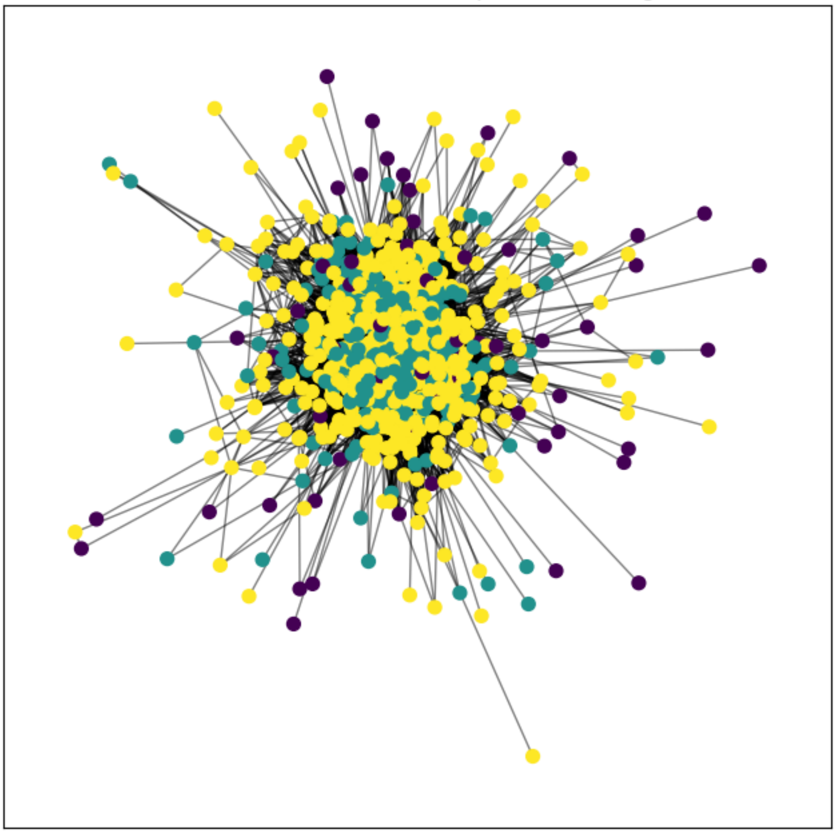
Visibility of Gender in Social Platforms:
On many social platforms, gender is a visible attribute, which can impact how individuals connect and form groups, especially in environments where gender-related discussions and events are prominent.

Consistency Across Different Universities:
The pattern of gender influencing group formation is consistent across multiple universities in the dataset, suggesting that this is not an isolated phenomenon but rather a widespread tendency.
(b) Write the code to validate your research question and show the result using a few selected community detection algorithms and graphs.

```
import networkx as nx
import matplotlib.pyplot as plt
import community.community_louvain as cl

total_community = []
for i, G in enumerate(g_all_list):
```

```
    pos = nx.spring_layout(G)  # Fruchterman-Reingold 算法
    plt.figure(figsize=(8, 8))

    node_colors = [G.nodes[node]['gender'] for node in G.nodes()]
    nx.draw_networkx_nodes(G, pos, node_color=node_colors,
cmap=plt.get_cmap('viridis'), node_size=50)
    nx.draw_networkx_edges(G, pos, alpha=0.5)
    plt.title(f'Network Visualization of Campus {file_names_list[i]}')
    plt.show()


    partition = cl.best_partition(G)


    community_gender = {}
    for node, com_id in partition.items():
        gender = G.nodes[node]['gender']
        community_gender.setdefault(com_id, []).append(gender)

    total_community.append(community_gender)

    for com_id, genders in community_gender.items():
        print(f"Campus {file_names_list[i]}, Community {com_id}:")
        print(f"Gender distribution: {genders.count(0)} Unknown,
{genders.count(1)} Male, {genders.count(2)} Female")
        print(f"Total members: {len(genders)}")
        print(f"Proportion of Unknown:
{genders.count(0)/len(genders):.2f}")
        print(f"Proportion of Male:
{genders.count(1)/len(genders):.2f}")
        print(f"Proportion of Female:
{genders.count(2)/len(genders):.2f}\n")
```

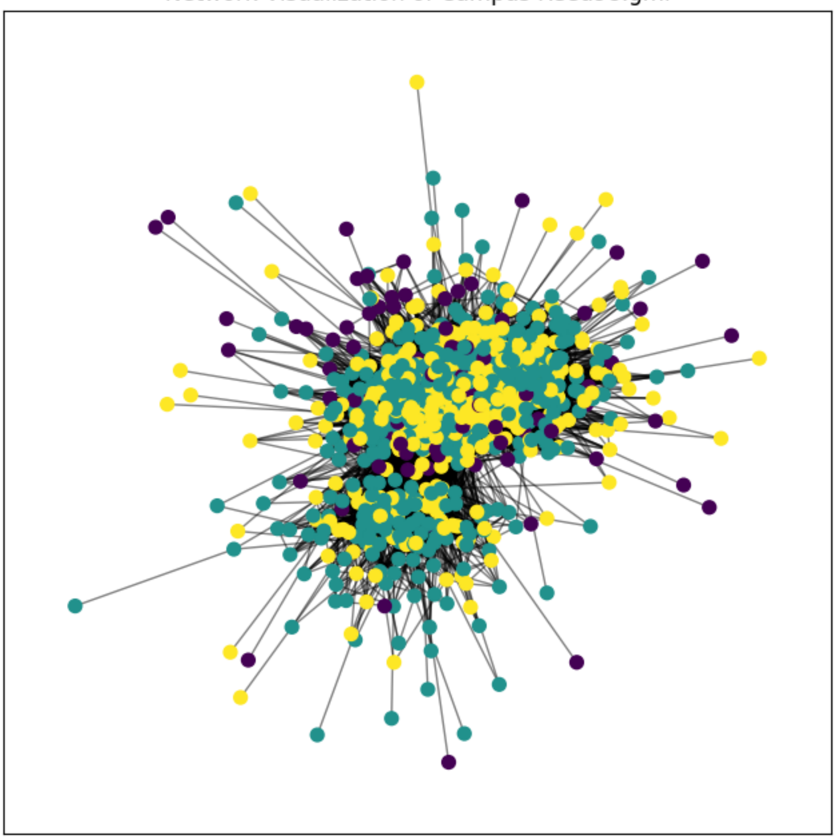
Network Visualization of Campus Caltech36.gml



```
Campus Caltech36.gml, Community 0:
Gender distribution: 6 Unknown, 42 Male, 71 Female
Total members: 119
Proportion of Unknown: 0.05
Proportion of Male: 0.35
Proportion of Female: 0.60

Campus Caltech36.gml, Community 2:
Gender distribution: 9 Unknown, 22 Male, 49 Female
Total members: 80
Proportion of Unknown: 0.11
Proportion of Male: 0.28
Proportion of Female: 0.61

Campus Caltech36.gml, Community 5:
Gender distribution: 9 Unknown, 47 Male, 88 Female
Total members: 144
Proportion of Unknown: 0.06
Proportion of Male: 0.33
Proportion of Female: 0.61

Campus Caltech36.gml, Community 3:
Gender distribution: 6 Unknown, 22 Male, 50 Female
Total members: 78
Proportion of Unknown: 0.08
Proportion of Male: 0.28
Proportion of Female: 0.64

Campus Caltech36.gml, Community 1:
Gender distribution: 14 Unknown, 22 Male, 80 Female
Total members: 116
```

```
Proportion of Unknown: 0.12
Proportion of Male: 0.19
Proportion of Female: 0.69

Campus Caltech36.gml, Community 6:
Gender distribution: 8 Unknown, 39 Male, 45 Female
Total members: 92
Proportion of Unknown: 0.09
Proportion of Male: 0.42
Proportion of Female: 0.49

Campus Caltech36.gml, Community 7:
Gender distribution: 0 Unknown, 5 Male, 12 Female
Total members: 17
Proportion of Unknown: 0.00
Proportion of Male: 0.29
Proportion of Female: 0.71

Campus Caltech36.gml, Community 4:
Gender distribution: 11 Unknown, 28 Male, 77 Female
Total members: 116
Proportion of Unknown: 0.09
Proportion of Male: 0.24
Proportion of Female: 0.66
```

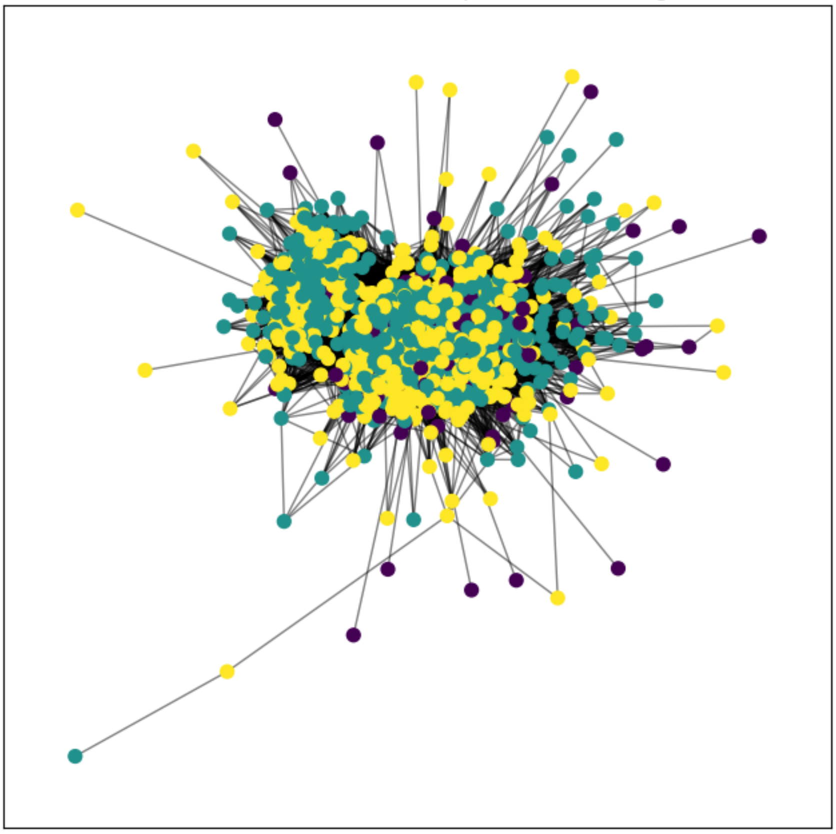
Network Visualization of Campus Reed98.gml



```
Campus Reed98.gml, Community 4:
Gender distribution: 23 Unknown, 123 Male, 89 Female
Total members: 235
Proportion of Unknown: 0.10
Proportion of Male: 0.52
Proportion of Female: 0.38
```

```
Campus Reed98.gml, Community 1:
Gender distribution: 24 Unknown, 105 Male, 68 Female
Total members: 197
Proportion of Unknown: 0.12
Proportion of Male: 0.53
Proportion of Female: 0.35

Campus Reed98.gml, Community 2:
Gender distribution: 19 Unknown, 53 Male, 70 Female
Total members: 142
Proportion of Unknown: 0.13
Proportion of Male: 0.37
Proportion of Female: 0.49

Campus Reed98.gml, Community 3:
Gender distribution: 6 Unknown, 114 Male, 58 Female
Total members: 178
Proportion of Unknown: 0.03
Proportion of Male: 0.64
Proportion of Female: 0.33

Campus Reed98.gml, Community 5:
Gender distribution: 7 Unknown, 33 Male, 20 Female
Total members: 60
Proportion of Unknown: 0.12
Proportion of Male: 0.55
Proportion of Female: 0.33

Campus Reed98.gml, Community 0:
Gender distribution: 18 Unknown, 76 Male, 56 Female
Total members: 150
Proportion of Unknown: 0.12
Proportion of Male: 0.51
Proportion of Female: 0.37
```

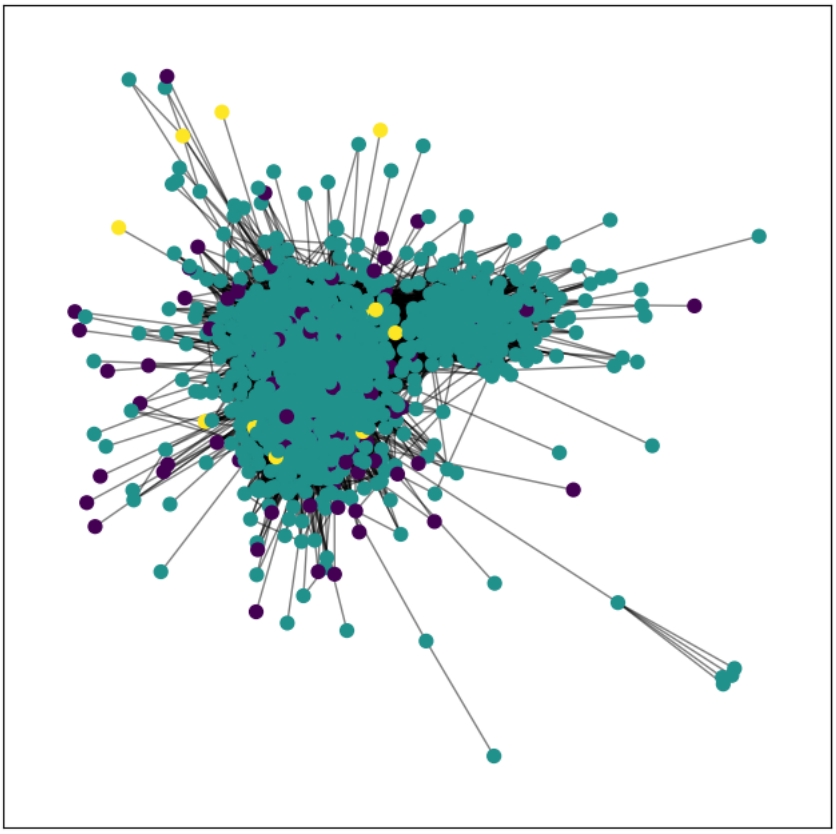
Network Visualization of Campus Haverford76.gml



```
Campus Haverford76.gml, Community 0:
Gender distribution: 39 Unknown, 197 Male, 135 Female
Total members: 371
Proportion of Unknown: 0.11
Proportion of Male: 0.53
Proportion of Female: 0.36

Campus Haverford76.gml, Community 1:
Gender distribution: 23 Unknown, 175 Male, 126 Female
Total members: 324
Proportion of Unknown: 0.07
Proportion of Male: 0.54
Proportion of Female: 0.39

Campus Haverford76.gml, Community 2:
Gender distribution: 18 Unknown, 135 Male, 115 Female
Total members: 268
Proportion of Unknown: 0.07
Proportion of Male: 0.50
Proportion of Female: 0.43

Campus Haverford76.gml, Community 3:
Gender distribution: 2 Unknown, 159 Male, 130 Female
Total members: 291
Proportion of Unknown: 0.01
Proportion of Male: 0.55
Proportion of Female: 0.45

Campus Haverford76.gml, Community 4:
Gender distribution: 14 Unknown, 66 Male, 112 Female
Total members: 192
```

```
Proportion of Unknown: 0.07
Proportion of Male: 0.34
Proportion of Female: 0.58
```

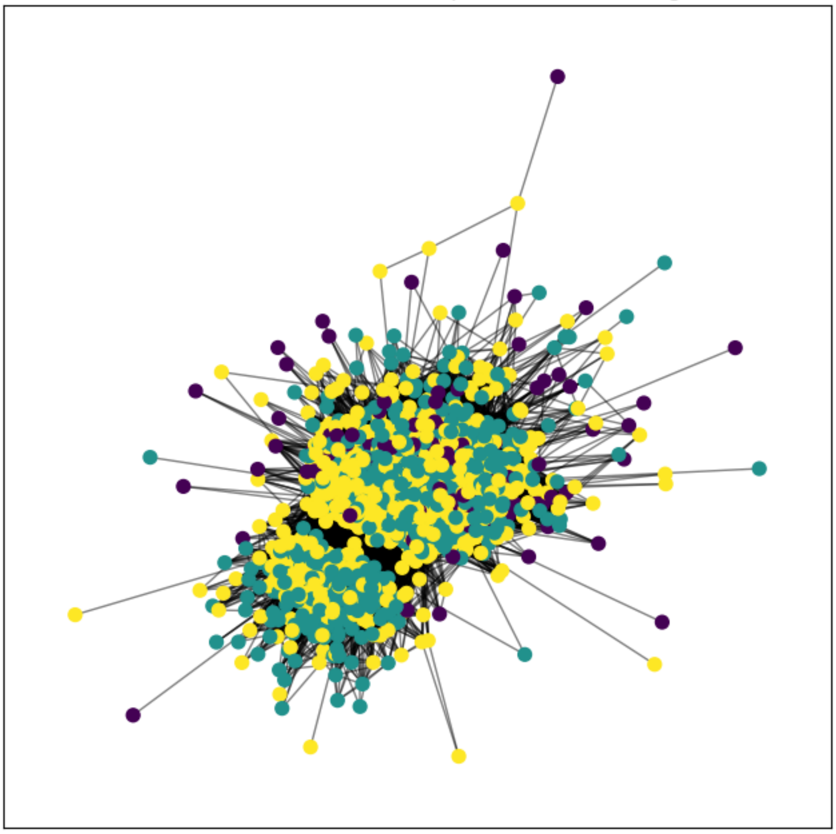
Network Visualization of Campus Simmons81.gml



```
Campus Simmons81.gml, Community 0:
Gender distribution: 26 Unknown, 319 Male, 6 Female
Total members: 351
Proportion of Unknown: 0.07
Proportion of Male: 0.91
Proportion of Female: 0.02

Campus Simmons81.gml, Community 1:
Gender distribution: 41 Unknown, 374 Male, 4 Female
Total members: 419
Proportion of Unknown: 0.10
Proportion of Male: 0.89
Proportion of Female: 0.01

Campus Simmons81.gml, Community 2:
Gender distribution: 20 Unknown, 363 Male, 1 Female
Total members: 384
Proportion of Unknown: 0.05
Proportion of Male: 0.95
Proportion of Female: 0.00

Campus Simmons81.gml, Community 3:
Gender distribution: 5 Unknown, 307 Male, 0 Female
Total members: 312
Proportion of Unknown: 0.02
Proportion of Male: 0.98
```

```
Proportion of Female: 0.00

Campus Simmons81.gml, Community 5:
Gender distribution: 2 Unknown, 11 Male, 3 Female
Total members: 16
Proportion of Unknown: 0.12
Proportion of Male: 0.69
Proportion of Female: 0.19

Campus Simmons81.gml, Community 6:
Gender distribution: 1 Unknown, 22 Male, 0 Female
Total members: 23
Proportion of Unknown: 0.04
Proportion of Male: 0.96
Proportion of Female: 0.00

Campus Simmons81.gml, Community 4:
Gender distribution: 0 Unknown, 5 Male, 0 Female
Total members: 5
Proportion of Unknown: 0.00
Proportion of Male: 1.00
Proportion of Female: 0.00
```

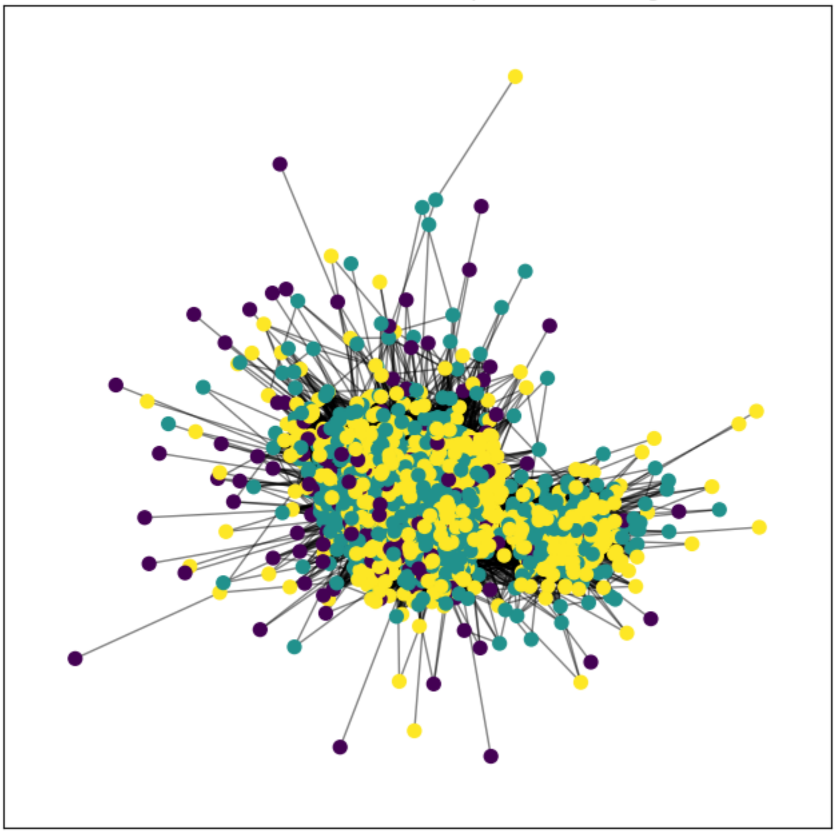
Network Visualization of Campus Swarthmore42.gml



```
Campus Swarthmore42.gml, Community 0:
Gender distribution: 28 Unknown, 108 Male, 94 Female
Total members: 230
Proportion of Unknown: 0.12
Proportion of Male: 0.47
```

```
Proportion of Female: 0.41

Campus Swarthmore42.gml, Community 1:
Gender distribution: 20 Unknown, 136 Male, 137 Female
Total members: 293
Proportion of Unknown: 0.07
Proportion of Male: 0.46
Proportion of Female: 0.47

Campus Swarthmore42.gml, Community 2:
Gender distribution: 33 Unknown, 108 Male, 168 Female
Total members: 309
Proportion of Unknown: 0.11
Proportion of Male: 0.35
Proportion of Female: 0.54

Campus Swarthmore42.gml, Community 3:
Gender distribution: 3 Unknown, 162 Male, 137 Female
Total members: 302
Proportion of Unknown: 0.01
Proportion of Male: 0.54
Proportion of Female: 0.45

Campus Swarthmore42.gml, Community 4:
Gender distribution: 16 Unknown, 72 Male, 84 Female
Total members: 172
Proportion of Unknown: 0.09
Proportion of Male: 0.42
Proportion of Female: 0.49

Campus Swarthmore42.gml, Community 5:
Gender distribution: 25 Unknown, 116 Male, 82 Female
Total members: 223
Proportion of Unknown: 0.11
Proportion of Male: 0.52
Proportion of Female: 0.37

Campus Swarthmore42.gml, Community 6:
Gender distribution: 13 Unknown, 69 Male, 46 Female
Total members: 128
Proportion of Unknown: 0.10
Proportion of Male: 0.54
Proportion of Female: 0.36
```

Network Visualization of Campus Amherst41.gml

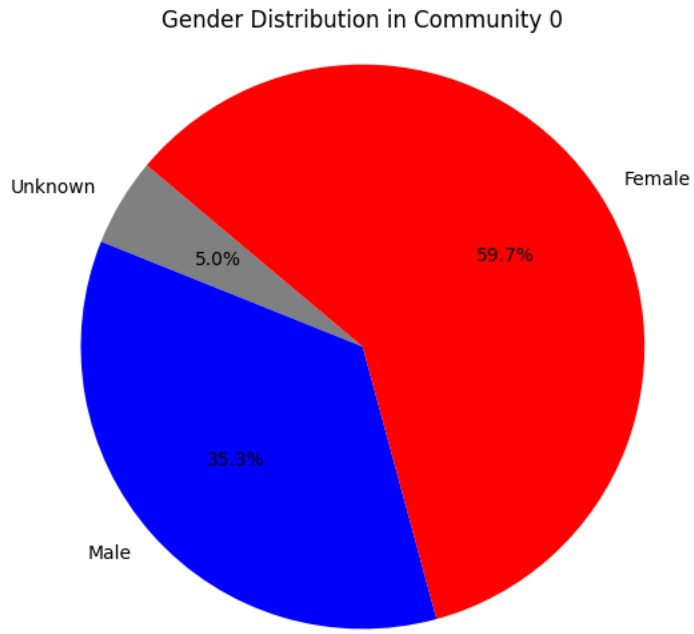


```
Campus Amherst41.gml, Community 0:
Gender distribution: 27 Unknown, 177 Male, 183 Female
Total members: 387
Proportion of Unknown: 0.07
Proportion of Male: 0.46
Proportion of Female: 0.47

Campus Amherst41.gml, Community 1:
Gender distribution: 18 Unknown, 104 Male, 112 Female
Total members: 234
Proportion of Unknown: 0.08
Proportion of Male: 0.44
Proportion of Female: 0.48

Campus Amherst41.gml, Community 2:
Gender distribution: 46 Unknown, 152 Male, 206 Female
Total members: 404
Proportion of Unknown: 0.11
Proportion of Male: 0.38
Proportion of Female: 0.51

Campus Amherst41.gml, Community 3:
Gender distribution: 7 Unknown, 195 Male, 184 Female
Total members: 386
Proportion of Unknown: 0.02
Proportion of Male: 0.51
Proportion of Female: 0.48

Campus Amherst41.gml, Community 6:
Gender distribution: 86 Unknown, 293 Male, 258 Female
Total members: 637
```

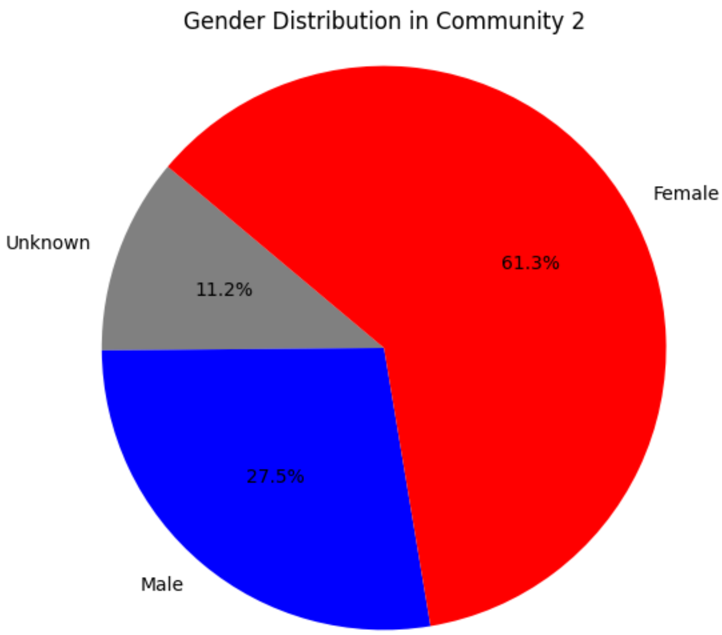
```
Proportion of Unknown: 0.14
Proportion of Male: 0.46
Proportion of Female: 0.41

Campus Amherst41.gml, Community 4:
Gender distribution: 19 Unknown, 92 Male, 72 Female
Total members: 183
Proportion of Unknown: 0.10
Proportion of Male: 0.50
Proportion of Female: 0.39

Campus Amherst41.gml, Community 5:
Gender distribution: 0 Unknown, 2 Male, 2 Female
Total members: 4
Proportion of Unknown: 0.00
Proportion of Male: 0.50
Proportion of Female: 0.50
```

……

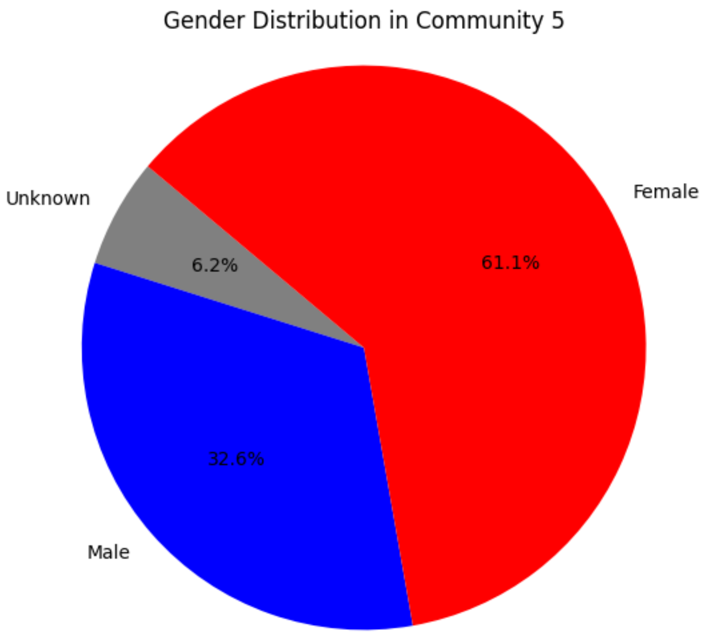
```
Campus Caltech36.gml
Community 0:
Gender distribution: Unknown=6, Male=42, Female=71
Chi-squared: 53.46, p-value: 0.0000
Significant gender distribution found.
```



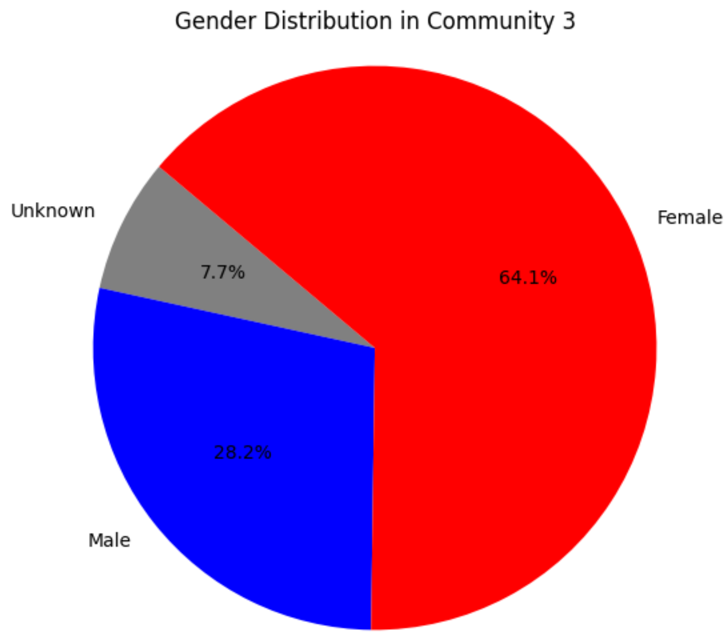

```
Community 2:
Gender distribution: Unknown=9, Male=22, Female=49
Chi-squared: 31.22, p-value: 0.0000
Significant gender distribution found.
```



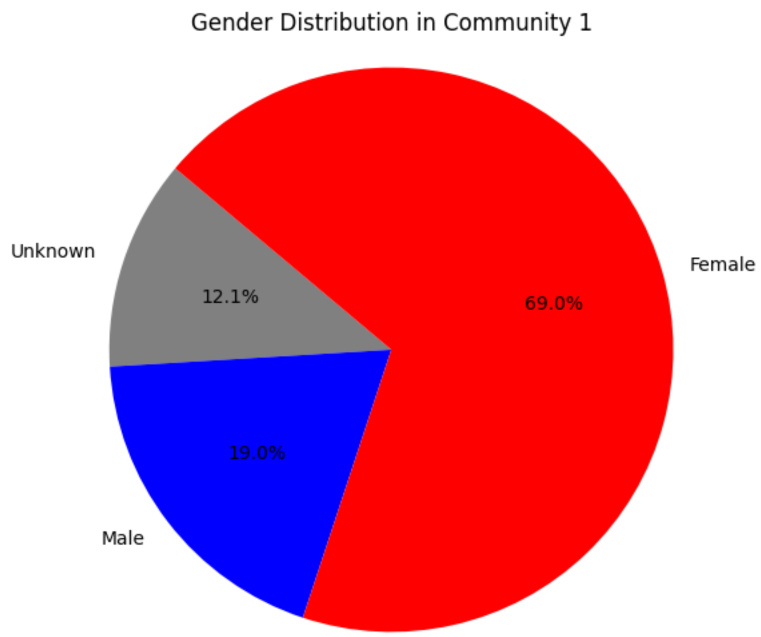

```
Community 5:
Gender distribution: Unknown=9, Male=47, Female=88
Chi-squared: 65.04, p-value: 0.0000
Significant gender distribution found.
```



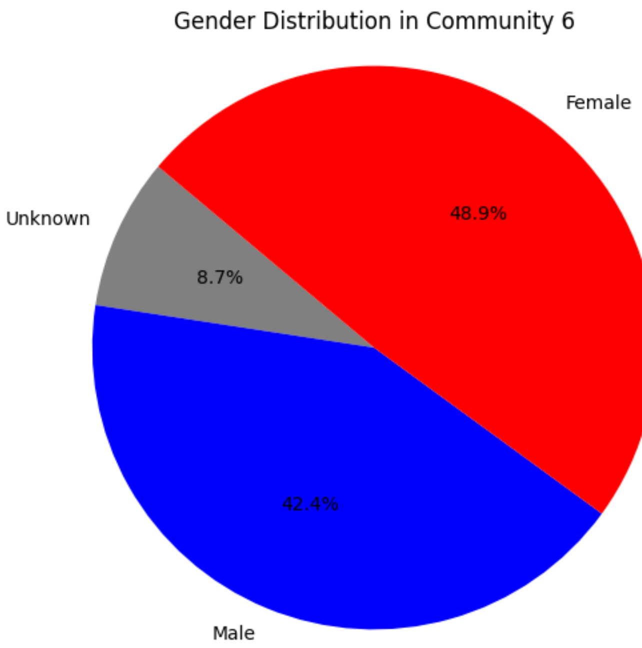

```
Community 3:
Gender distribution: Unknown=6, Male=22, Female=50
Chi-squared: 38.15, p-value: 0.0000
Significant gender distribution found.
```



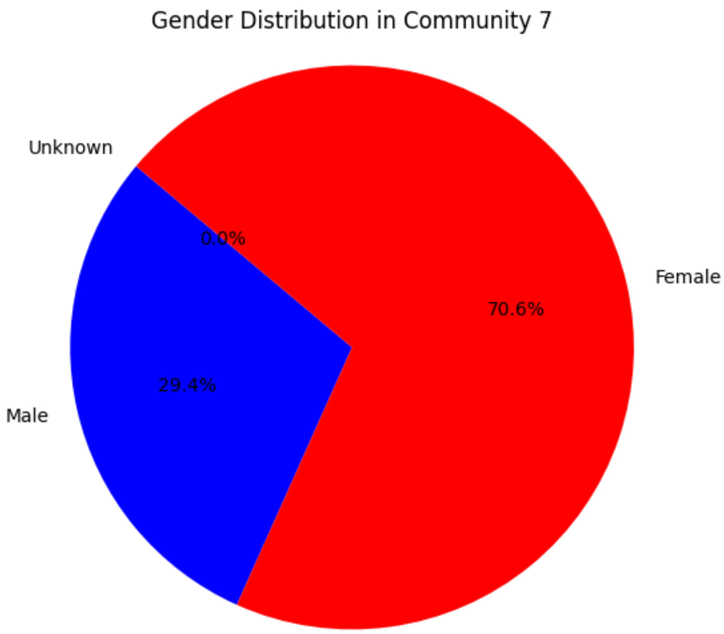

```
Community 1:
Gender distribution: Unknown=14, Male=22, Female=80
Chi-squared: 67.10, p-value: 0.0000
Significant gender distribution found.
```



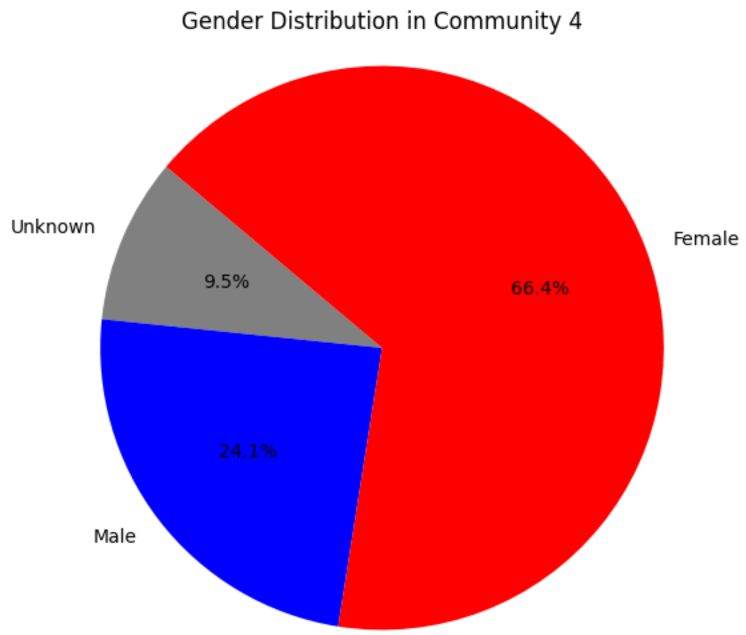

```
Community 6:
Gender distribution: Unknown=8, Male=39, Female=45
Chi-squared: 25.72, p-value: 0.0000
Significant gender distribution found.
```




```
Community 7:
Gender distribution: Unknown=0, Male=5, Female=12
Chi-squared: 12.82, p-value: 0.0016
Significant gender distribution found.
```




```
Community 4:
Gender distribution: Unknown=11, Male=28, Female=77
Chi-squared: 60.74, p-value: 0.0000
Significant gender distribution found.
```



……

(c) Explain the results and conclude whether your experiment confirms your initial hypothesis.

Gender Distribution in Communities:
The charts indicate a significant presence of single-gender dominance within several communities. For instance, in some communities, the proportion of females is much higher than that of males and vice versa. This suggests that gender could indeed be a significant factor in how these communities are formed.
Statistical Significance:

The provided p-values from the Chi-squared tests are very low ($p < 0.05$), indicating that the gender distributions in these communities are unlikely to be due to random chance. This supports the hypothesis of gender-based group formation.
Consistency Across Communities:

If most communities consistently show a higher proportion of one gender, it would support the hypothesis. However, the data shows some communities with a more balanced gender distribution, indicating that while gender may be a factor, it might not be the sole driver of group formation.

Community Sizes:
The size of the communities varies, and it seems that both larger and smaller communities can have a skewed gender distribution. This suggests that the influence of gender on group formation does not depend on the size of the community.
Unknown Gender Proportions:

Some communities have a significant proportion of members with an unknown gender. The presence of this category can affect the accuracy of conclusions regarding the impact of gender on group formation.

In conclusion, while the evidence points towards gender being an important factor in group formation, it is not definitive proof that it is the primary driver. The variations in gender distribution across communities suggest that other factors may also play significant roles. Moreover, the existence of mixed-gender communities implies that while gender may influence group formation, it does not create entirely homogenous groups.