In [10]:
```
group_num = 9
import os
import json
import subprocess
from collections import defaultdict
import random
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# same function from HW1 to download files

def download_files(group_num, folders = None):
    node_names = {}
    root_folder = f"Group{group_num}"
    if not os.path.exists(root_folder):
        os.mkdir(root_folder)
    else:
        print(f"Folder {root_folder} already exists")
        return
    if folders is None:
        folders = ['Facebook-Ego', 'Twitter-Ego']
    for folder in folders:
        os.mkdir(os.path.join(root_folder, folder))
        res = subprocess.run(["curl", "-s", f"https://api.github.com/repos/1250326/exercise_c
        for file_info in json.loads(res.stdout):
            os.system(f"wget -O {os.path.join(root_folder, folder, (fname:=file_info['name'])
            print(f"Downloaded file: {fname}")
            if '.' in fname:
                node_names[folder.split('-')[0]] = fname.split('.')[0]
        print(f"Downloaded folder: {folder}")
    return node_names

!rm -rf Group$group_num
node_names = download_files(group_num)
node_names
```

```
Downloaded file: 3437_2.edges
Downloaded file: 3437_2.egofeat
Downloaded file: 3437_2.feat
Downloaded file: 3437_2.featnames
Downloaded file: Description
Downloaded folder: Facebook-Ego
Downloaded file: 6408382.circles
Downloaded file: 6408382.edges
Downloaded file: 6408382.egofeat
Downloaded file: 6408382.feat
Downloaded file: 6408382.featnames
Downloaded file: Description
Downloaded folder: Twitter-Ego
```

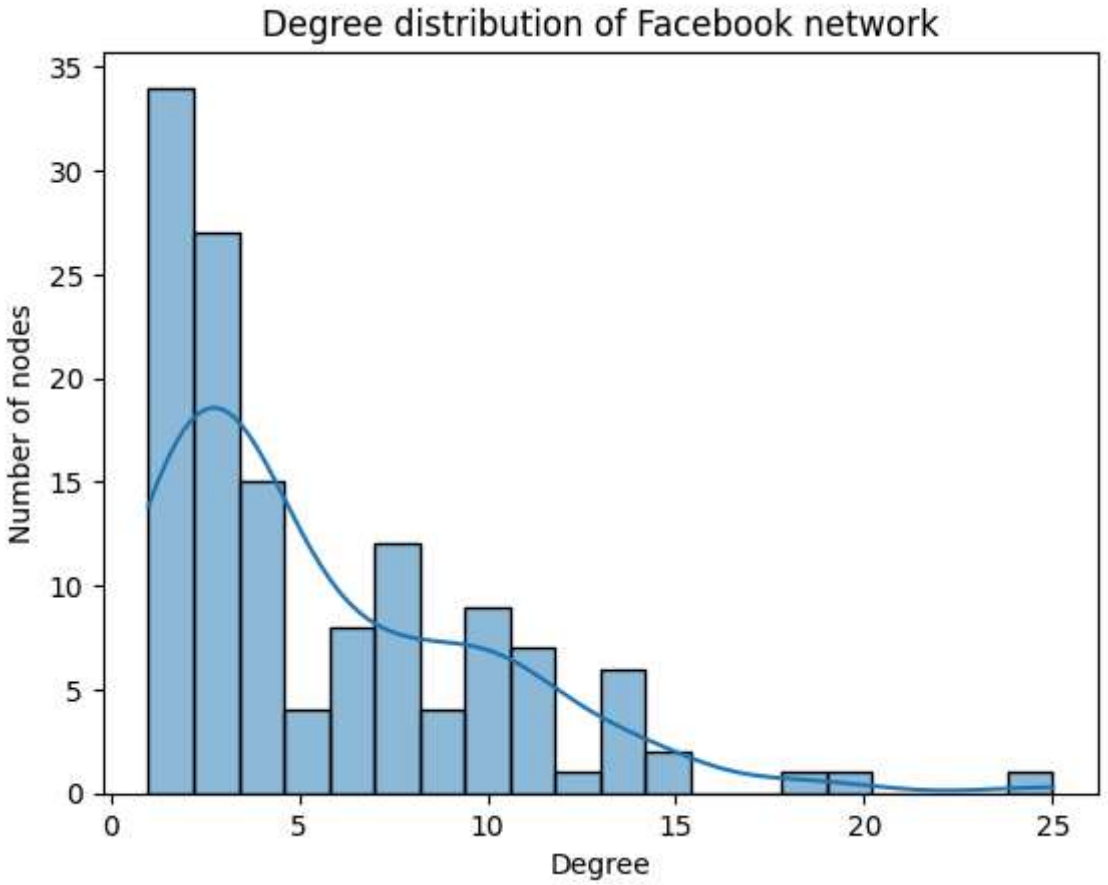
Out[2]:
```
{'Facebook': '3437_2', 'Twitter': '6408382'}
```

A

What is the degree distribution of the network? Please plot the distribution and attach it to your answer sheet.

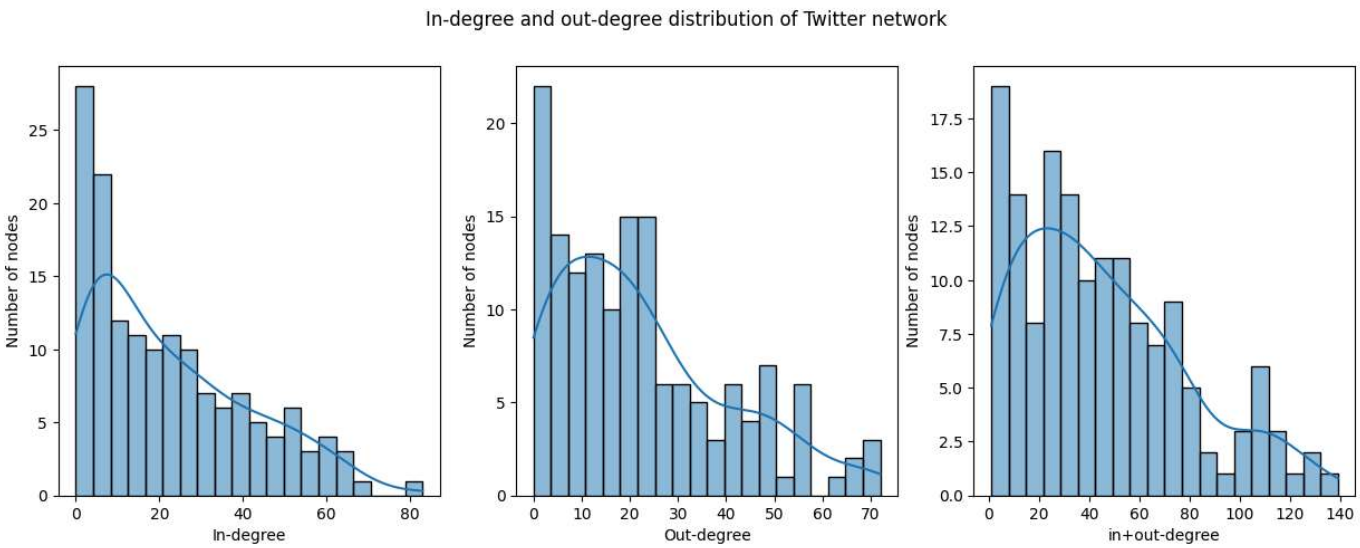
Facebook network

In [18]:
```
facebook_net = nx.read_edgelist(f"Group{group_num}/Facebook-Ego/{node_names['Facebook']}.edge
sns.histplot(dict(nx.degree(facebook_net)).values(), bins=20, kde=True, legend=False)
plt.xlabel('Degree')
plt.ylabel('Number of nodes')
plt.title('Degree distribution of Facebook network')
plt.show()
```



Twitter network

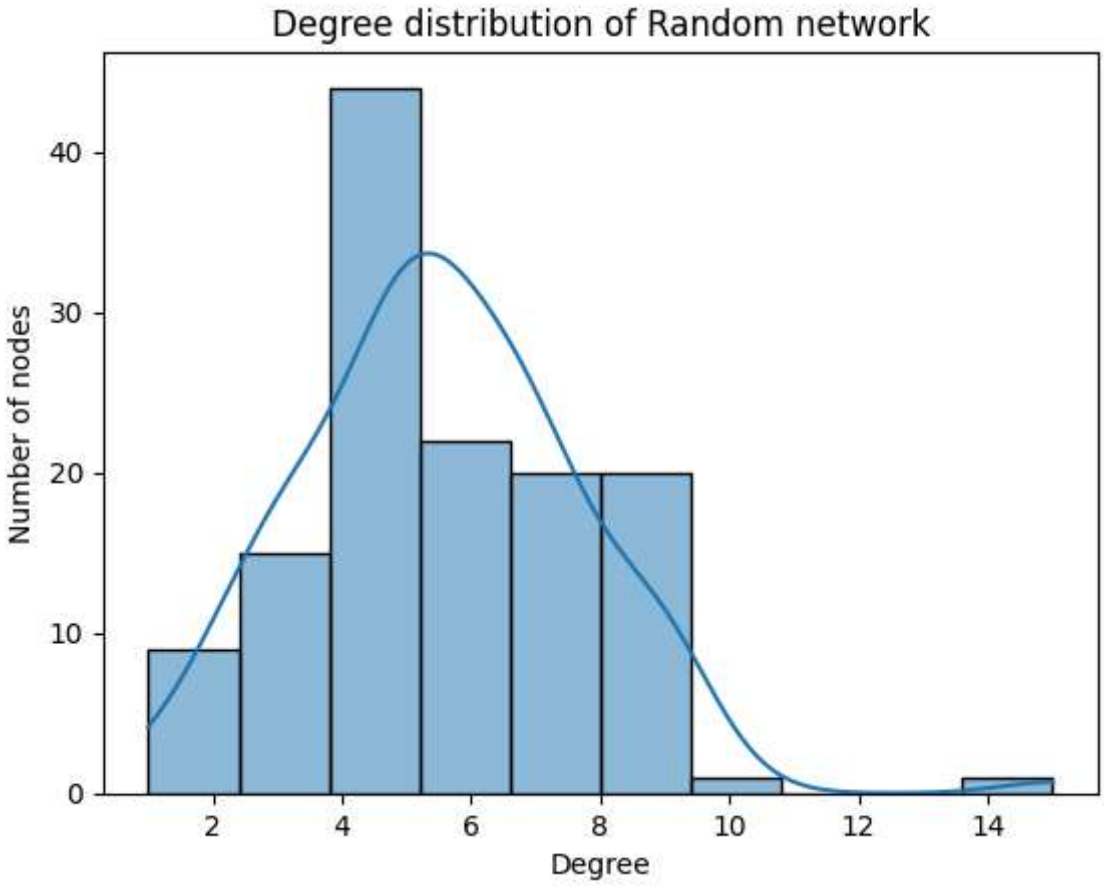
In [189…
```
twitter_net = nx.read_edgelist(f"Group{group_num}/Twitter-Ego/{node_names['Twitter']}.edges",
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))
fig.suptitle('In-degree and out-degree distribution of Twitter network')

sns.histplot(dict(twitter_net.in_degree()).values(), bins=20, kde=True, ax=ax1, legend=False)
ax1.set_xlabel('In-degree')
ax1.set_ylabel('Number of nodes')

sns.histplot(dict(twitter_net.out_degree()).values(), bins=20, kde=True, ax=ax2, legend=False
ax2.set_xlabel('Out-degree')
ax2.set_ylabel('Number of nodes')

sns.histplot(dict(twitter_net.degree()).values(), bins=20, kde=True, ax=ax3, legend=False)
ax3.set_xlabel('in+out-degree')
ax3.set_ylabel('Number of nodes')


plt.show()
```



B

According to the degree distribution, what is the main difference between the chosen network and a random network with the same number of nodes? What may cause the difference?

In [24]:
```
# Lets create a random network with the same number of nodes and edges as the facebook networ
random_net = nx.gnm_random_graph(n=facebook_net.number_of_nodes(), m=facebook_net.number_of_e
sns.histplot(dict(nx.degree(random_net)).values(), bins=10, kde=True, legend=False)
plt.xlabel('Degree')
plt.ylabel('Number of nodes')
plt.title('Degree distribution of Random network')
plt.show()
```



At first glance, it is evident that the selected network contains many nodes with a low degree, while the random network features more nodes with a medium degree. More precisely, the degree distribution of the random network closely resembles a normal distribution, whereas the degree distribution of the selected network aligns more with a power-law distribution.

C

What is the average path length of the network?

Facebook network

In [85]:
```
len(list(nx.connected_components(facebook_net)))
```

Out[85]:
```
10
```

As we can see, there are 10 connected components in the Facebook network. So, we can compute the average path length for each connected component.

In [93]:
```
for connected_component in nx.connected_components(facebook_net):
    if len(connected_component) == 1:
        continue
    subgraph = facebook_net.subgraph(connected_component)
    avg_path = sum([sum(path.values()) for _, path in nx.all_pairs_shortest_path_length(subgr
    assert (nx_avg_path :=nx.average_shortest_path_length(subgraph)) == avg_path, f"Expected
    print(f"Average shortest path length for connected component with {n} nodes: {avg_path}")
```

```
Average shortest path length for connected component with 97 nodes: 4.625429553264605
Average shortest path length for connected component with 5 nodes: 1.3
Average shortest path length for connected component with 5 nodes: 1.1
Average shortest path length for connected component with 4 nodes: 1.0
Average shortest path length for connected component with 4 nodes: 1.3333333333333333
Average shortest path length for connected component with 7 nodes: 2.380952380952381
Average shortest path length for connected component with 3 nodes: 1.3333333333333333
Average shortest path length for connected component with 2 nodes: 1.0
Average shortest path length for connected component with 3 nodes: 1.3333333333333333
Average shortest path length for connected component with 2 nodes: 1.0
```

Twitter network

In [95]:
```
twitter_net.is_directed()
```

Out[95]:
```
True
```

In [94]:
```
for strongly_cc in nx.strongly_connected_components(twitter_net):
    if len(strongly_cc) == 1:
        continue
    subgraph = twitter_net.subgraph(strongly_cc)
    avg_path = sum([sum(path.values()) for _, path in nx.all_pairs_shortest_path_length(subgr
    assert (nx_avg_path :=nx.average_shortest_path_length(subgraph)) == avg_path, f"Expected
    print(f"Average shortest path length for strongly connected component with {n} nodes: {av
```

```
Average shortest path length for strongly connected component with 141 nodes: 2.13191489361702
14
```

D

Please randomly choose 5 nodes from the network, and calculate the clustering coefficients of these nodes, respectively.

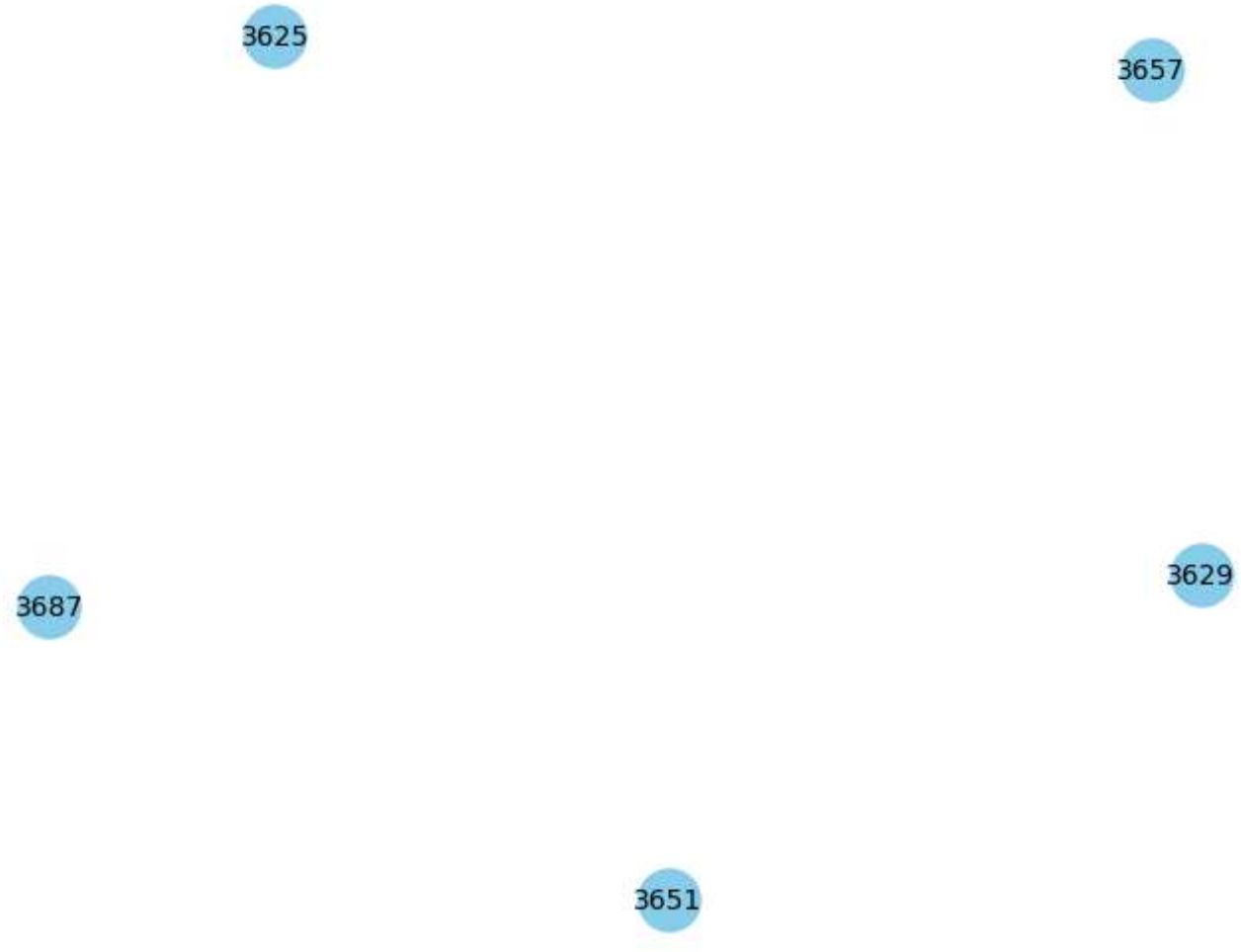
Facebook network

In [105…
```
facebook_net.nodes()
```

Out[105…
```
NodeView(('3710', '3713', '3633', '3596', '3702', '3627', '3718', '3584', '3716', '3616', '37
30', '3635', '3683', '3692', '3667', '3593', '3644', '3698', '3677', '3592', '3615', '3599',
'3656', '3715', '3684', '3586', '3693', '3697', '3639', '3613', '3595', '3628', '3649', '364
8', '3643', '3636', '3724', '3617', '3605', '3707', '3731', '3642', '3728', '3608', '3719',
'3645', '3601', '3652', '3676', '3682', '3662', '3666', '3618', '3711', '3725', '3590', '364
0', '3641', '3655', '3625', '3604', '3646', '3659', '3663', '3594', '3705', '3714', '3629',
'3687', '3658', '3624', '3587', '3696', '3680', '3611', '3722', '3670', '3672', '3706', '359
1', '3623', '3685', '3679', '3674', '3609', '3712', '3585', '3721', '3602', '3634', '3620',
'3654', '3689', '3621', '3703', '3632', '3720', '3681', '3622', '3657', '3626', '3647', '363
1', '3603', '3638', '3695', '3651', '3665', '3690', '3606', '3610', '3664', '3607', '3732',
'3675', '3708', '3598', '3612', '3673', '3660', '3694', '3589', '3600', '3699', '3653', '363
0', '3686', '3614', '3700', '3669', '3727', '3637'))
```

In [112…
```
selected_nodes = random.sample(list(facebook_net.nodes()), 5)
print(f"Selected nodes: {selected_nodes}")
nx.draw(facebook_net.subgraph(selected_nodes), with_labels=True
        , node_color='skyblue', node_size=500, font_size=10, font_color='black')
```

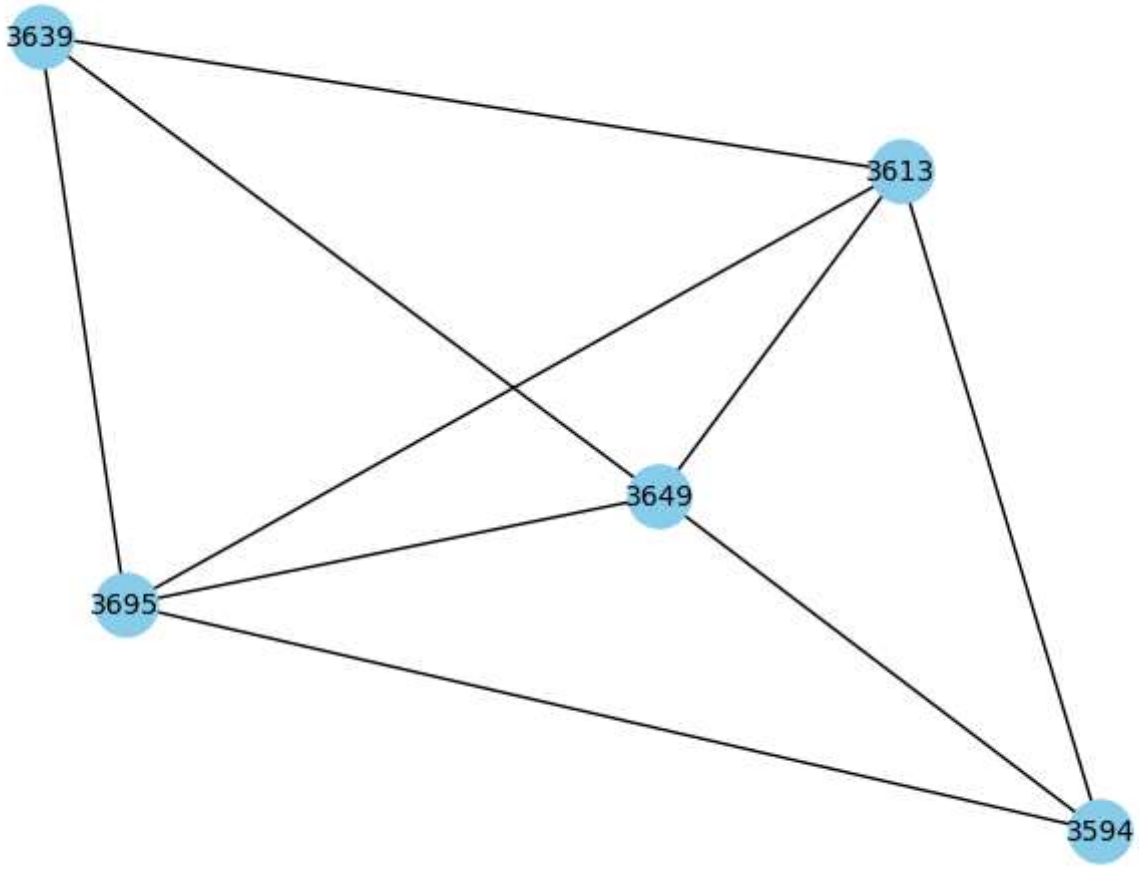
```
Selected nodes: ['3629', '3657', '3687', '3651', '3625']
```



as we can see the clustring coefficient of the nodes are zero, to have a better result let's choose a connected component with 5 nodes.

In [140…
```
selected_nodes = [i for i in nx.connected_components(facebook_net) if len(i) == 5][1]
print(f"Selected nodes: {selected_nodes}")
nx.draw((sg:=facebook_net.subgraph(selected_nodes)), with_labels=True
        , node_color='skyblue', node_size=500, font_size=10, font_color='black')
```

```
Selected nodes: {'3639', '3613', '3695', '3649', '3594'}
```

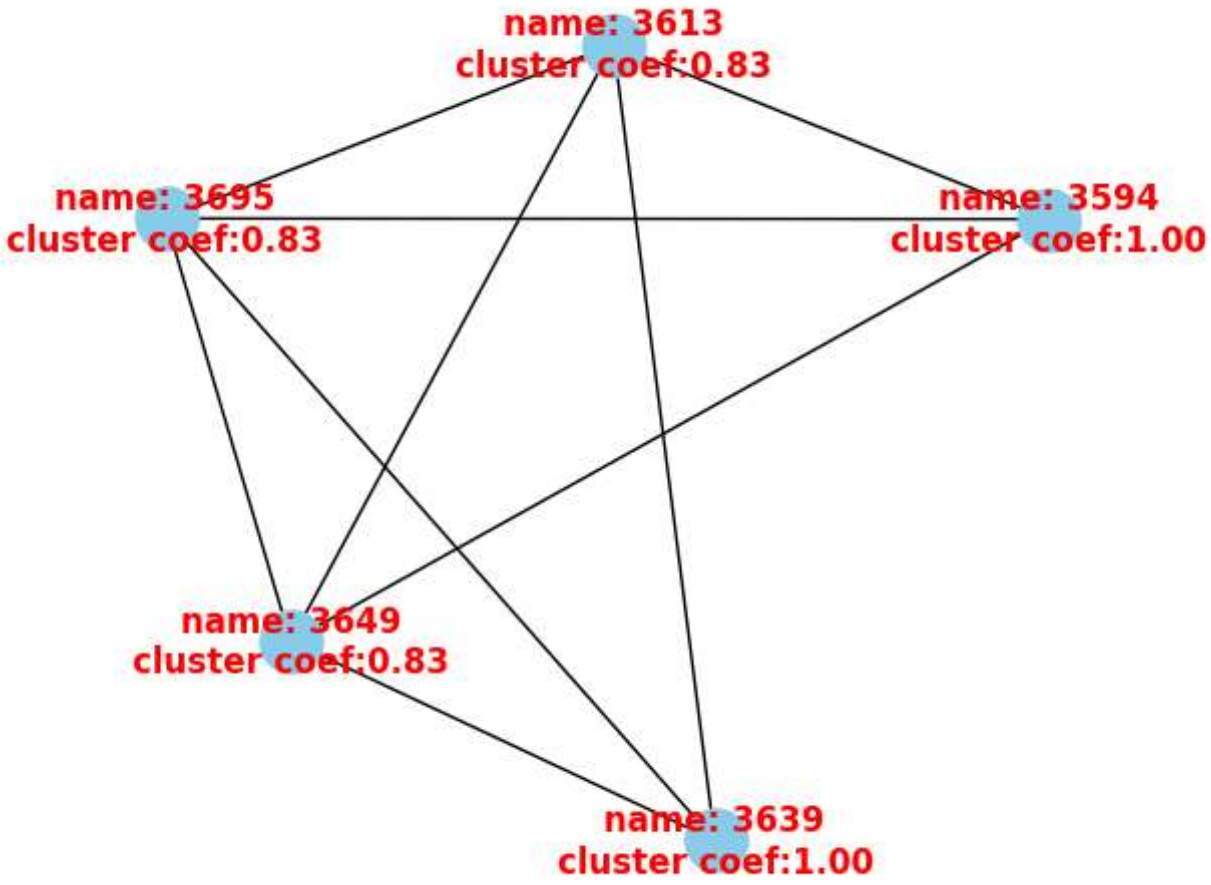


In [141…
```
clusters = nx.clustering(sg)
clusters
```

Out[141…
```
{'3639': 1.0,
 '3613': 0.8333333333333334,
 '3695': 0.8333333333333334,
 '3649': 0.8333333333333334,
 '3594': 1.0}
```

In [148…
```
nx.draw(sg, with_labels=True, labels = {node: f"name: {node}\ncluster coef:{clusters[node]:.2
        node_color='skyblue', node_size=500, font_size=12, font_color='red', font_weight='bol
plt.xlim(plt.xlim()[0]*1.2, plt.xlim()[1]*1.2)
```

Out[148…
```
(-1.1856240265328641, 1.3806383195323937)
```

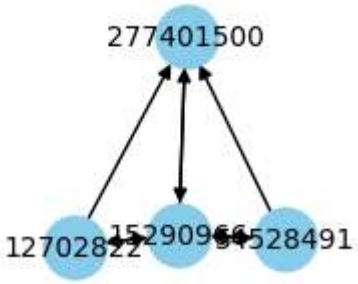


Twitter network

In [156…

```
random.seed(7)
twitter_sample = random.sample(list(twitter_net.nodes()), 5)
print(f"Selected nodes: {twitter_sample}")

nx.draw((twitter_sample:=twitter_net.subgraph(twitter_sample)), with_labels=True,
        node_color='skyblue', node_size=500, font_size=10, font_color='black')
```

```
Selected nodes: [54528491, 15290966, 12702822, 277401500, 86120067]
```

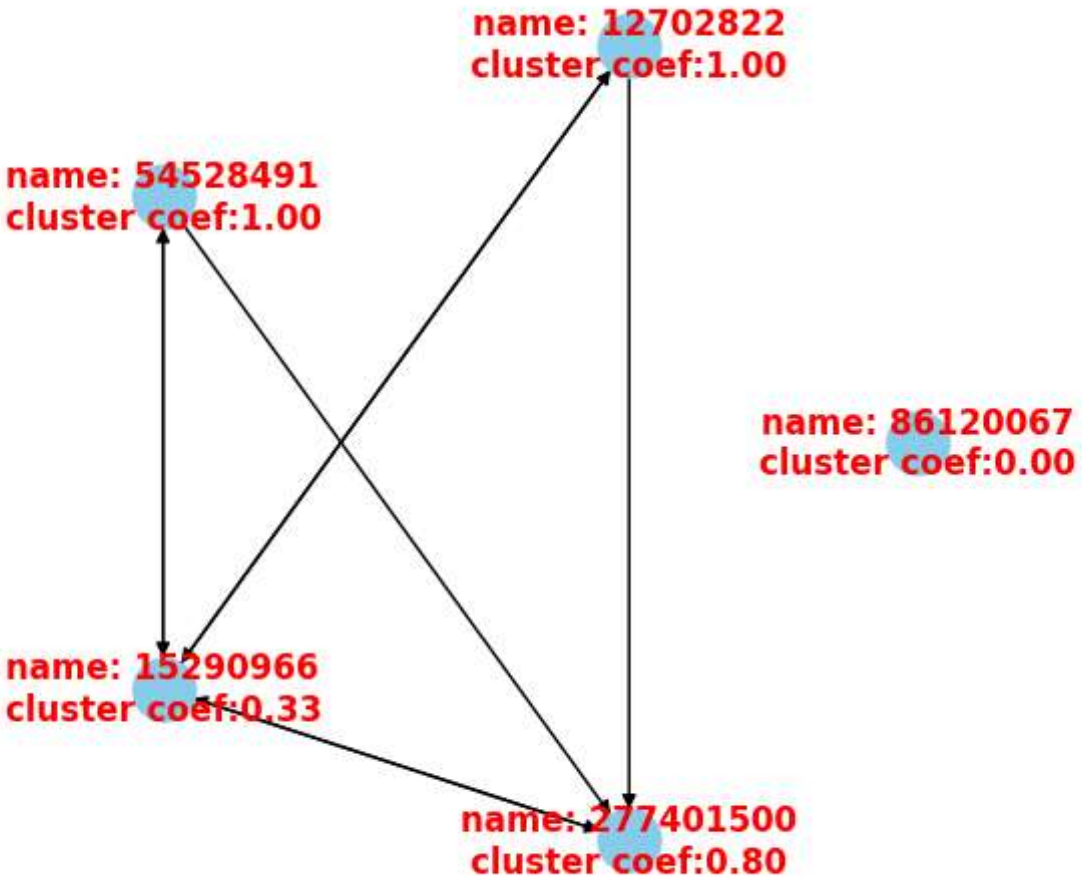


In [161…
```
clusters = nx.clustering(twitter_sample)
clusters
```

Out[161…
```
{86120067: 0,
 12702822: 1.0,
 54528491: 1.0,
 15290966: 0.3333333333333333,
 277401500: 0.8}
```

In [169…
```
nx.draw(twitter_sample, with_labels=True, labels = {node: f"name: {node}\ncluster coef:{clust
        node_color='skyblue', node_size=500, font_size=12, font_color='red', font_weight='bol

plt.xlim(plt.xlim()[0]*1.5, plt.xlim()[1]*1.5)
```

Out[169…
```
(-1.437247053898783, 1.6398689073285135)
```



E

What is the average clustering coefficient?

Facebook network

In [171…
```
nx.average_clustering(facebook_net)
```

Out[171…
```
0.5209621418860091
```

In [174…
```
sum(nx.clustering(facebook_net).values())/facebook_net.number_of_nodes()
```

Out[174…
```
0.5209621418860091
```

Twitter network

In [175…
```
nx.average_clustering(twitter_net)
```

Out[175…
```
0.4548006068015104
```

In [177…
```
sum(nx.clustering(twitter_net).values())/twitter_net.number_of_nodes()
```

Out[177…
```
0.4548006068015104
```

F

Please list the nodes which compose the largest component in the network.

Facebook network

In [181…
```
max_cluster_coef = max((fcl:=nx.clustering(facebook_net)).values())
print("\n".join([f"Node {node} has the highest clustering coefficient of {max_cluster_coef:.2
```

```
Node 3716 has the highest clustering coefficient of 1.00
Node 3715 has the highest clustering coefficient of 1.00
Node 3639 has the highest clustering coefficient of 1.00
Node 3595 has the highest clustering coefficient of 1.00
Node 3628 has the highest clustering coefficient of 1.00
Node 3676 has the highest clustering coefficient of 1.00
Node 3666 has the highest clustering coefficient of 1.00
Node 3725 has the highest clustering coefficient of 1.00
Node 3641 has the highest clustering coefficient of 1.00
Node 3655 has the highest clustering coefficient of 1.00
Node 3663 has the highest clustering coefficient of 1.00
Node 3594 has the highest clustering coefficient of 1.00
Node 3587 has the highest clustering coefficient of 1.00
Node 3591 has the highest clustering coefficient of 1.00
Node 3712 has the highest clustering coefficient of 1.00
Node 3602 has the highest clustering coefficient of 1.00
Node 3689 has the highest clustering coefficient of 1.00
Node 3657 has the highest clustering coefficient of 1.00
Node 3647 has the highest clustering coefficient of 1.00
Node 3638 has the highest clustering coefficient of 1.00
Node 3665 has the highest clustering coefficient of 1.00
Node 3607 has the highest clustering coefficient of 1.00
Node 3598 has the highest clustering coefficient of 1.00
Node 3630 has the highest clustering coefficient of 1.00
Node 3686 has the highest clustering coefficient of 1.00
```

In [185…
```
sorted(nx.clustering(facebook_net).items(), key=lambda x: x[1], reverse=True)[:30]
```

Out[185…
```
[('3716', 1.0),
 ('3715', 1.0),
 ('3639', 1.0),
 ('3595', 1.0),
 ('3628', 1.0),
 ('3676', 1.0),
 ('3666', 1.0),
 ('3725', 1.0),
 ('3641', 1.0),
 ('3655', 1.0),
 ('3663', 1.0),
 ('3594', 1.0),
 ('3587', 1.0),
 ('3591', 1.0),
 ('3712', 1.0),
 ('3602', 1.0),
 ('3689', 1.0),
 ('3657', 1.0),
 ('3647', 1.0),
 ('3638', 1.0),
 ('3665', 1.0),
 ('3607', 1.0),
 ('3598', 1.0),
 ('3630', 1.0),
 ('3686', 1.0),
 ('3693', 0.9523809523809523),
 ('3722', 0.8888888888888888),
 ('3613', 0.8333333333333334),
 ('3649', 0.8333333333333334),
 ('3658', 0.8333333333333334)]
```

Twitter network

In [182…
```
max_cluster_coef = max((tcl:=nx.clustering(twitter_net)).values())
print("\n".join([f"Node {node} has the highest clustering coefficient of {max_cluster_coef:.2
```

```
Node 18158500 has the highest clustering coefficient of 1.00
Node 74880095 has the highest clustering coefficient of 1.00
```

In [186…
```
sorted(nx.clustering(twitter_net).items(), key=lambda x: x[1], reverse=True)[:30]
```

Out[186…
```
[(18158500, 1.0),
 (74880095, 1.0),
 (468447294, 0.96),
 (358114153, 0.9),
 (226272886, 0.8770491803278688),
 (15747590, 0.8666666666666667),
 (431669343, 0.8170731707317073),
 (18880017, 0.7982456140350878),
 (153133767, 0.7629427792915532),
 (373422039, 0.7307692307692307),
 (276597679, 0.7086038961038961),
 (263684365, 0.6829268292682927),
 (162671556, 0.6785714285714286),
 (14801264, 0.6700507614213198),
 (143156249, 0.6644295302013423),
 (18940840, 0.6173421300659755),
 (102350867, 0.6071428571428571),
 (739773, 0.6),
 (246691076, 0.5919540229885057),
 (7623482, 0.5909090909090909),
 (1249751, 0.586720867208672l),
 (14456798, 0.5740740740740741),
 (86120067, 0.5721533258173619),
 (4494, 0.5714285714285714),
 (53609015, 0.5673076923076923),
 (22712077, 0.5666666666666667),
 (46422814, 0.5662768031189084),
 (142450248, 0.5648330058939096),
 (284468794, 0.5627802690582959),
 (15039741, 0.5627413127413128)]
```