SENA PROJECT REPORT

Problem Statement

Detecting communities from a given undirected graph and performing a detailed analysis on the communities.(Graph visualisation,Community size, Community description,Frequency plot)

Dataset Description:

- Nodes: 351
- Edges: 1150
- Type: Undirected,unweighted
- Social circles from Facebook (anonymized) source: https://snap.stanford.edu/data/

Tools Used

Micro Web Framework

- Flask

Python packages

- Networkx
  Graph manipulation and visualization
- Community
  This python package was used to perform community detection using the girvan-newman algorithm
- Communities
  This python package was used to perform community detection using the louvain-method algorithm
- Matplotlib
  Used for plotting the graphs

Challenges Faced

- Error when deprecated packages are used
- Graph plot overlapping
- Takes more time to generate results for bigger graphs
- Colab runtime disconnection (Longer execution time for graphs with large number of nodes and edges)

Contribution of Team Members

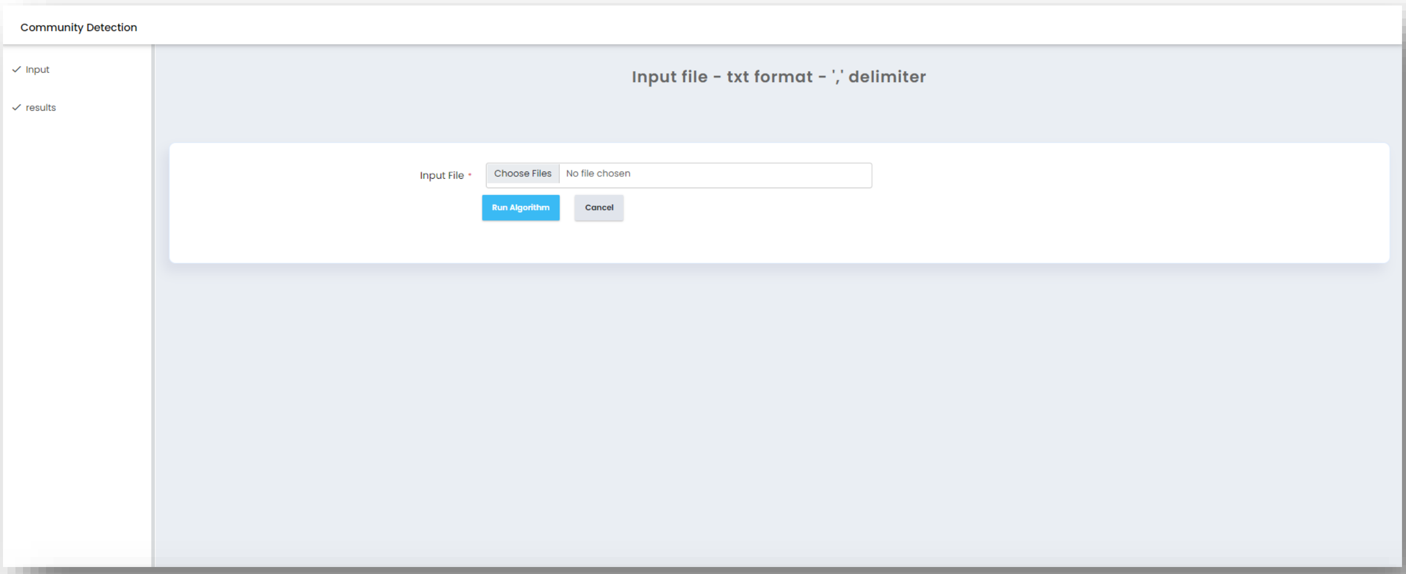
Roll No.	Name	Contribution
18Z209	Arunmozhi P	Algorithm implementation (girvan-newman)
18Z222	Jeffrey Nicholas Y	Algorithm implementation (girvan-newman)
18Z229	Mani Sankar T	Detailed analysis and visualization
18Z244	Salmaan Khan M	Algorithm implementation (louvain-method)
18Z248	Shibi Rahul S	Algorithm implementation (louvain-method)

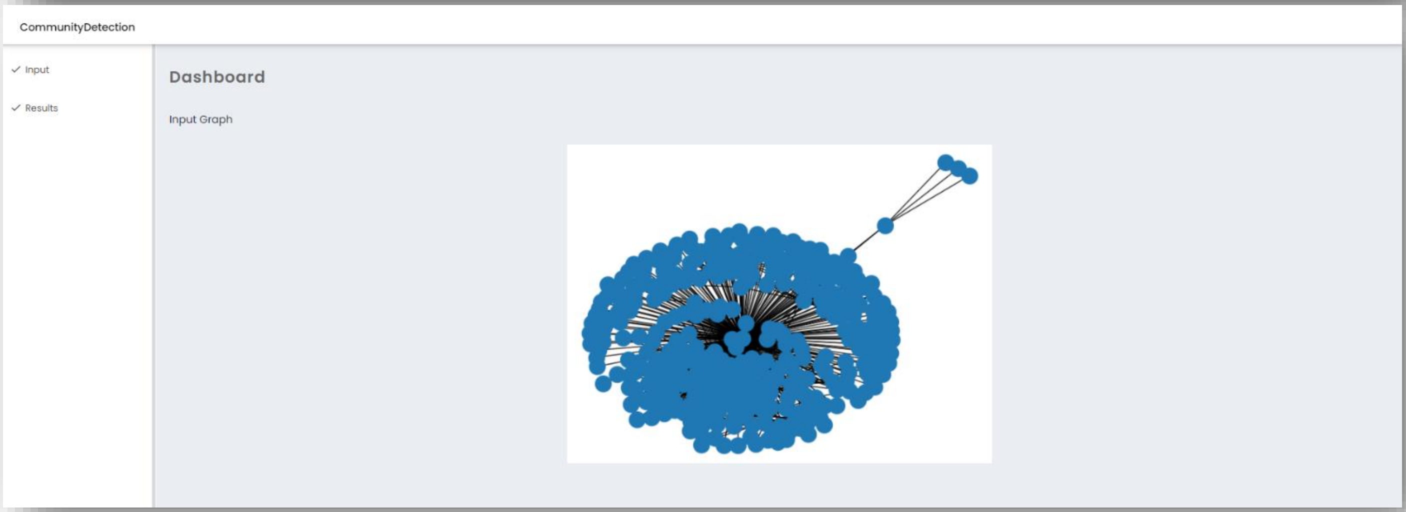
ANNEXURE I: CODE

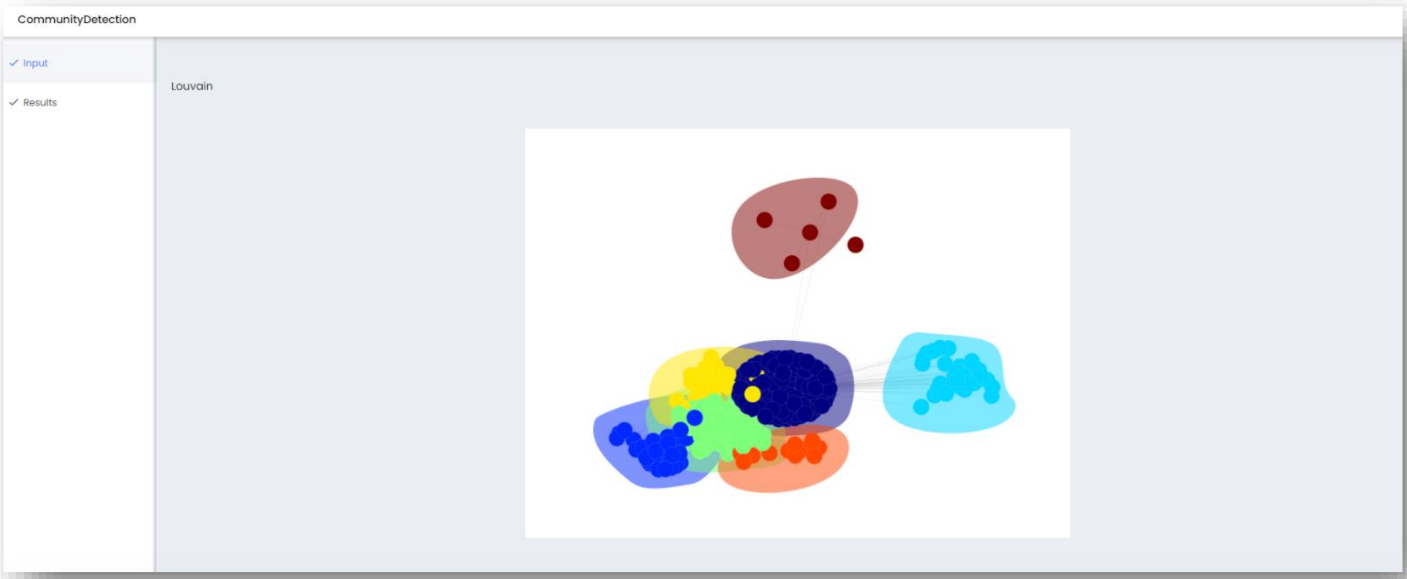
Community detection code:

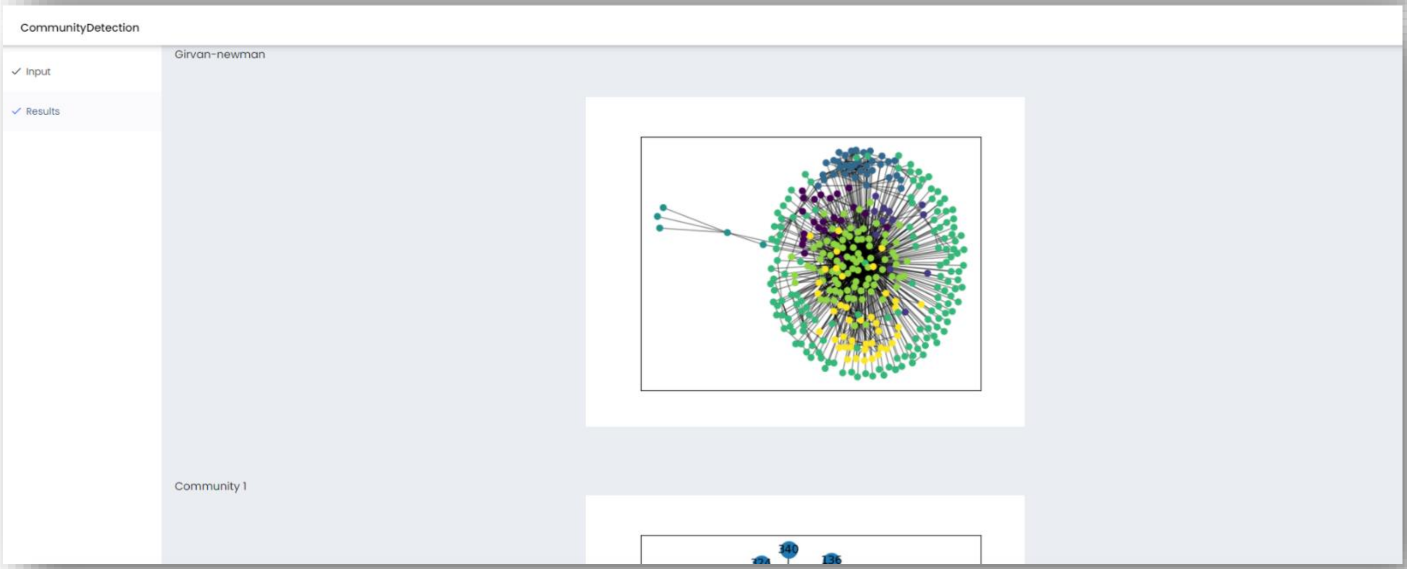
```
import os
   if request.method == 'POST':
    graph = request.files['input']
    graph.save(os.path.join("C:/Users/sanka/Downloads/cd_1/","graph.txt"))
    f="C:/Users/sanka/Downloads/cd_1/graph.txt"
    from communities.algorithms import louvain_method,girvan_newman
    from communities.visualization import draw_communities
    import networkx as nx
    import matplotlib.pyplot as plt
    import csv
    import sys
    import numpy as np


    def buildG(G, file_, delimiter_):
            reader = csv.reader(open(file_), delimiter=delimiter_)
            for line in reader:
                    G.add_edge(int(line[0]),int(line[1]))
    G = nx.Graph()
    print(G)
    buildG(G, f, ' ')
    nx.draw(G,pos=nx.spring_layout(G))
    plt.savefig('C:/My Web Sites/dashboard_CD_SENA/community/templates/admin/main/source/s
tatic/plotgraph.png')
    #matrix to array
    S= np.array(nx.to_numpy_matrix(G,dtype=int))
    print(S)
    plt.clf()
    plt.cla()
    plt.close()

    #louvain
    communities, _ = louvain_method(S)
    draw_communities(S, communities,False,'C:/My Web Sites/dashboard_CD_SENA/community/tem
plates/admin/main/source/static/louvain.png')
    print("communities")
    plt.clf()
    plt.cla()
    plt.close()

    #girvan
    import community as girvan_newman
    import matplotlib.cm as cm
    L=G
    partition = girvan_newman.best_partition(L)


    # draw the graph
    pos = nx.spring_layout(L)
    cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
    nx.draw_networkx_nodes(L, pos, partition.keys(), node_size=40,
    cmap=cmap, node_color=list(partition.values()))
    nx.draw_networkx_edges(L, pos, alpha=0.5)
    plt.savefig('C:/My Web Sites/dashboard_CD_SENA/community/templates/admin/main/source/static/girvan.png')
    plt.clf()
    plt.cla()
    plt.close()

    #visualization
    #Highest_degree
    com = set(partition.values())
    c_dict = {c: [l for l,i in partition.items() if i==c ] for c in com}
    highest_degree ={l: max(i, key=lambda x:G.degree(x)) for l,i in c_dict.items()}
    a = []
    for i in range(0,len(highest_degree)):
       a.append([])
       for j in range(0,2):
               a[i].append([])

    for i in range(0,len(highest_degree)):
        a[i][0]=(highest_degree[i])
        a[i][1]=(G.degree(highest_degree[i]))
    length=[]
    for i in range(0,len(highest_degree)):
        length.append(i)

    f=[]
    i=0
    y=0
    #total no of communities
    for i in partition:
      if partition[i] not in f:
        f.append(partition[i])
        y=y+1
    #print("Total No of Communities: ", y)
    #each community size
    import numpy as np
    l = np.zeros((y,), dtype=int)
    for i in partition:
      k = partition[i]
      l[k]=l[k]+1

    for i in range(0,y):
    #print("Community ", i+1," Size: ")
      print(l[i])

    #elements in each community
    k=len(partition)#no of partitions
    g=[]
    for i in range(0,y):
       g.append(i)#list of communities

    m=[]#community stored as list of lists
    for x in range(0,y):
     print("Community: ",x+1)
     q=[]
     for j in partition.keys():
        if(partition[j]==x):
           q.append(j)
     m.append(q)

    #generate bar graph
    height = l
    bars = range(len(l))
    y_pos = np.arange(len(bars))

    # Create bars
    plt.bar(y_pos, height)

    # Create names on the x-axis
    plt.xticks(y_pos, bars)

    plt.xlabel('Communities')
    plt.ylabel('No. of nodes')
    plt.savefig('C:/My Web Sites/dashboard_CD_SENA/community/templates/admin/main/source/s
tatic/bargraph.png')
    plt.clf()
    plt.cla()
    plt.close()
    #calculating radius, diameter,center, periphery for each community and image
    d=[]
    r=[]
    c=[]
    p=[]

    for i in range(0,len(highest_degree)):
        d.append([])
    for i in range(0,len(highest_degree)):
        r.append([])
    for i in range(0,len(highest_degree)):
        c.append([])
    for i in range(0,len(highest_degree)):
        p.append([])
    count=0

    for i in range(0,len(highest_degree)):
     L= G.copy()
     f='C:/Users/sanka/Downloads/cd_1/graph.txt'
     def removeG(L, file_, delimiter_):
            reader = csv.reader(open(file_), delimiter=delimiter_)
            for line in reader:
                if partition[int(line[0])]!=i:
                 if partition[int(line[1])]!=i and L.has_edge(int(line[0]),int(line[1])):

                    L.remove_edge(int(line[0]),int(line[1]))
                    L.remove_node(int(line[0]))
                    L.remove_node(int(line[1]))
                if partition[int(line[0])]!=i and L.has_node(int(line[0])):
                    L.remove_node(int(line[0]))
                if partition[int(line[1])]!=i  and L.has_node(int(line[1])):
                    L.remove_node(int(line[1]))
     removeG(L,f,' ')
     ecc = nx.eccentricity(L,v=None ,sp=None)
     k=nx.diameter(L,e=ecc)
     print(k)
     d[i].append(k)
     r[i].append(nx.radius(L,e=ecc))
     for k in nx.center(L,e=ecc):
        c[i].append(k)
     for k in nx.periphery(L,e=ecc):
        p[i].append(k)
     nx.draw_networkx(L, with_labels = True)
     f='C:/My Web Sites/dashboard_CD_SENA/community/templates/admin/main/source/static/com
munity'+str(count)+'.png'
     count=count+1
     plt.savefig(f)
     plt.clf()
     plt.cla()
     plt.close()
```

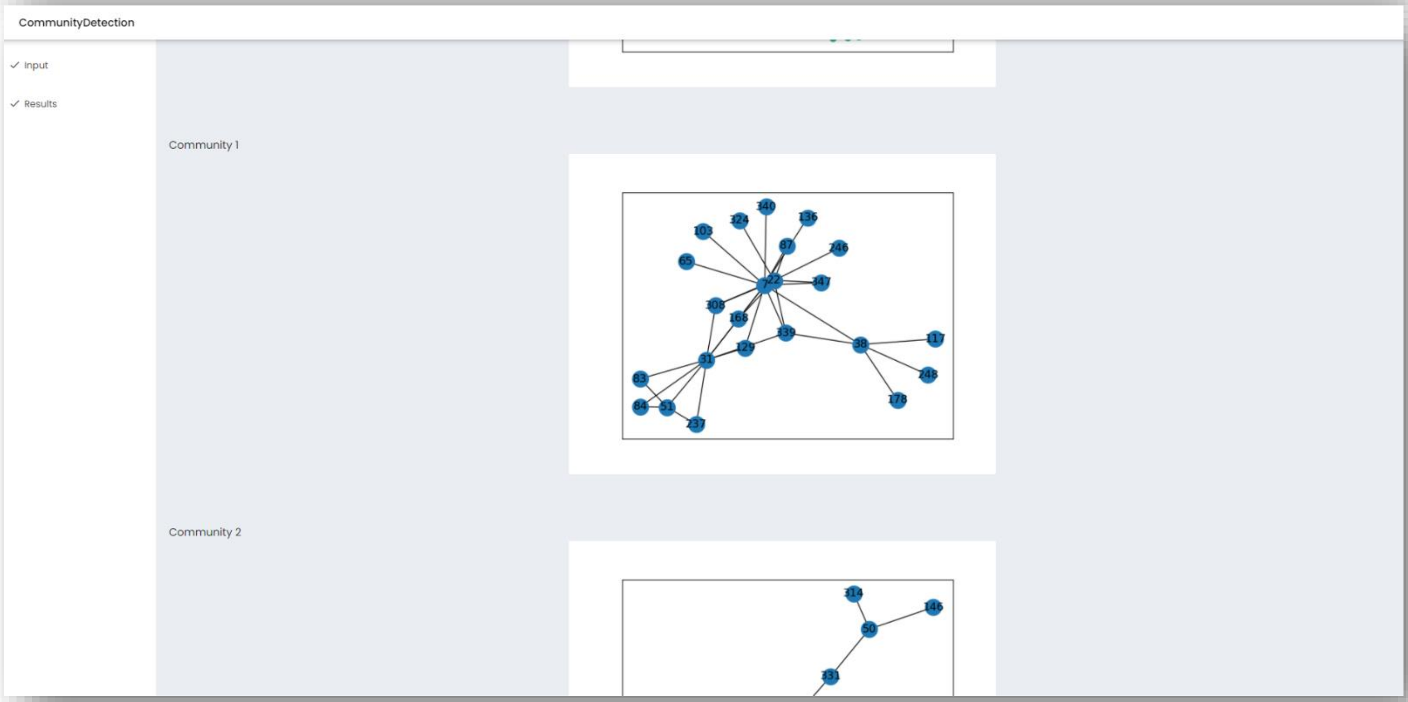
ANNEXURE II: Snapshots of the output

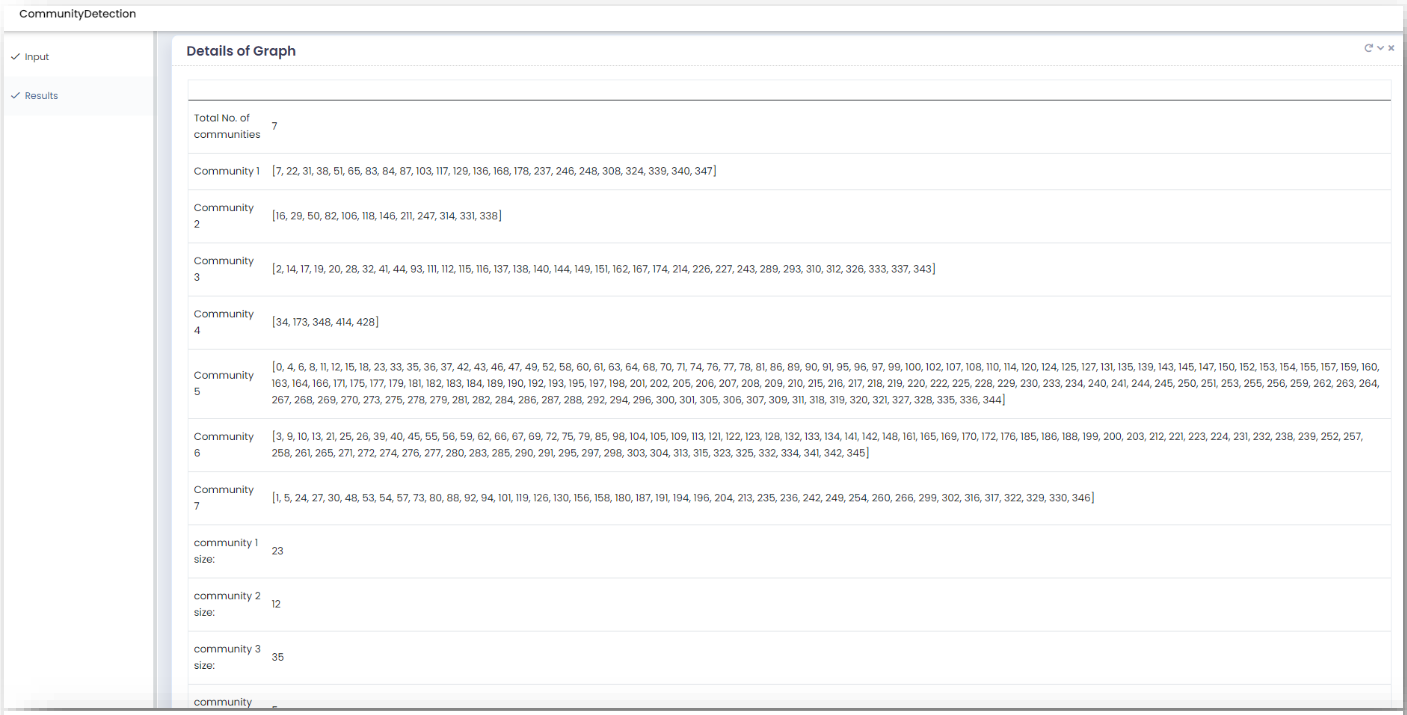





CommunityDetection
✓ Input
✓ Results
Louvain


CommunityDetection
✓ Input
✓ Results
Girvan-newman
Community 1
340
136


CommunityDetection
✓ Input
✓ Results
Community 1
340
324
136
103
87
746
65
72
347
306
166
139
129
38
117
31
248
63
178
84
51
237
Community 2
314
146
50
231



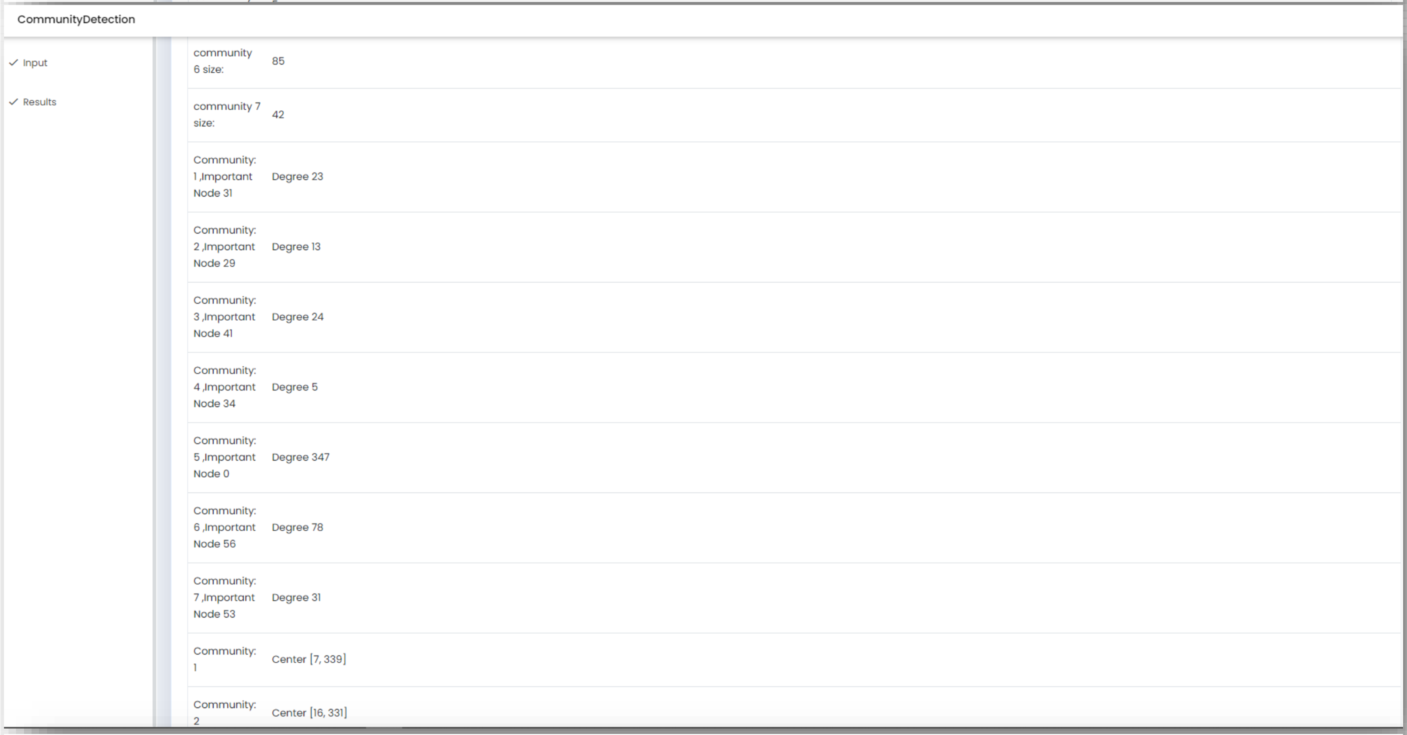
CommunityDetection

✓ Input

✓ Results

Details of Graph

Total No. of communities	7
Community 1	[7, 22, 31, 38, 51, 65, 83, 84, 87, 103, 117, 129, 136, 168, 178, 237, 246, 248, 308, 324, 339, 340, 347]
Community 2	[16, 29, 50, 82, 106, 118, 146, 211, 247, 314, 331, 338]
Community 3	[2, 14, 17, 19, 20, 28, 32, 41, 44, 93, 111, 112, 115, 116, 137, 138, 140, 144, 149, 151, 162, 167, 174, 214, 226, 227, 243, 289, 293, 310, 312, 326, 333, 337, 343]
Community 4	[34, 173, 348, 414, 428]
Community 5	[0, 4, 6, 8, 11, 12, 15, 18, 23, 33, 35, 36, 37, 42, 43, 46, 47, 49, 52, 58, 60, 61, 63, 64, 68, 70, 71, 74, 76, 77, 78, 81, 86, 89, 90, 91, 95, 96, 97, 99, 100, 102, 107, 108, 110, 114, 120, 124, 125, 127, 131, 135, 139, 143, 145, 147, 150, 152, 153, 154, 155, 157, 159, 160, 163, 164, 166, 171, 175, 177, 179, 181, 182, 183, 184, 189, 190, 192, 193, 195, 197, 198, 201, 202, 205, 206, 207, 208, 209, 210, 215, 216, 217, 218, 219, 220, 222, 225, 228, 229, 230, 233, 234, 240, 241, 244, 245, 250, 251, 253, 255, 256, 259, 262, 263, 264, 267, 268, 269, 270, 273, 275, 278, 279, 281, 282, 284, 286, 287, 288, 292, 294, 296, 300, 301, 305, 306, 307, 309, 311, 318, 319, 320, 321, 327, 328, 335, 336, 344]
Community 6	[3, 9, 10, 13, 21, 25, 26, 39, 40, 45, 55, 56, 59, 62, 66, 67, 69, 72, 75, 79, 85, 98, 104, 105, 109, 113, 121, 122, 123, 128, 132, 133, 134, 141, 142, 148, 161, 165, 169, 170, 172, 176, 185, 186, 188, 199, 200, 203, 212, 221, 223, 224, 231, 232, 238, 239, 252, 257, 258, 261, 265, 271, 272, 274, 276, 277, 280, 283, 285, 290, 291, 295, 297, 298, 303, 304, 313, 315, 323, 325, 332, 334, 341, 342, 345]
Community 7	[1, 5, 24, 27, 30, 48, 53, 54, 57, 73, 80, 88, 92, 94, 101, 119, 126, 130, 156, 158, 180, 187, 191, 194, 196, 204, 213, 235, 236, 242, 249, 254, 260, 266, 299, 302, 316, 317, 322, 329, 330, 346]
community 1 size:	23
community 2 size:	12
community 3 size:	35
community	

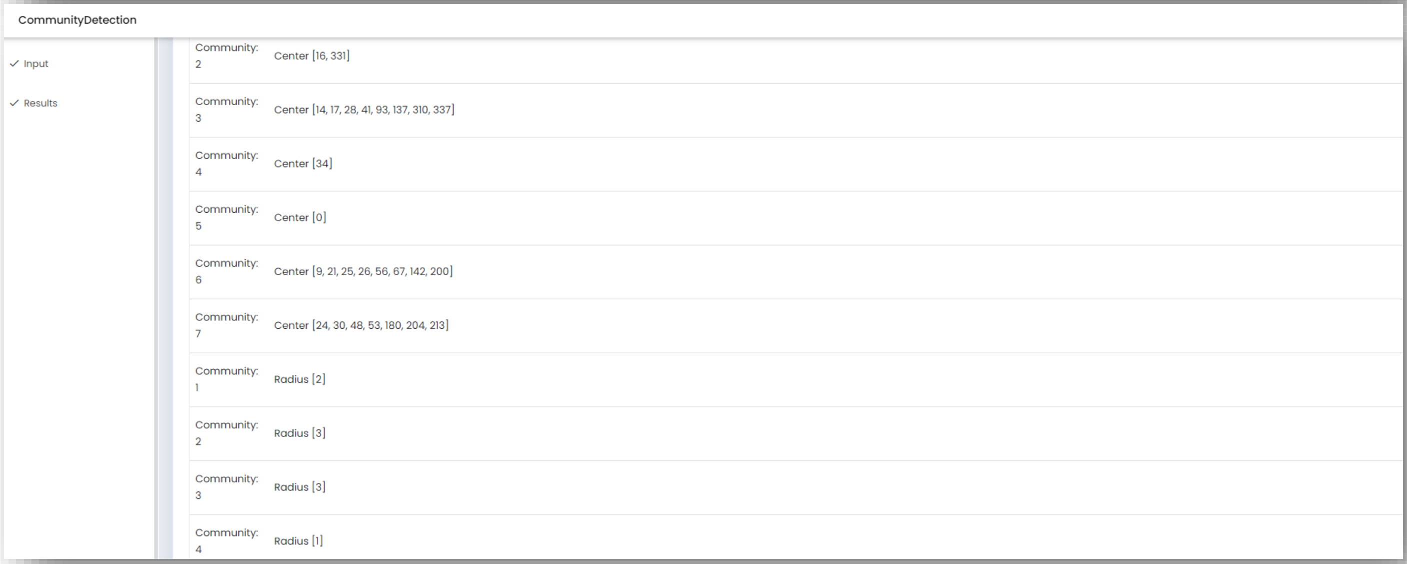


CommunityDetection

✓ Input

✓ Results

community 6 size:	85
community 7 size:	42
Community: 1 ,Important Node 31	Degree 23
Community: 2 ,Important Node 29	Degree 13
Community: 3 ,Important Node 41	Degree 24
Community: 4 ,Important Node 34	Degree 5
Community: 5 ,Important Node 0	Degree 347
Community: 6 ,Important Node 56	Degree 78
Community: 7 ,Important Node 53	Degree 31
Community: 1	Center [7, 339]
Community: 2	Center [16, 331]

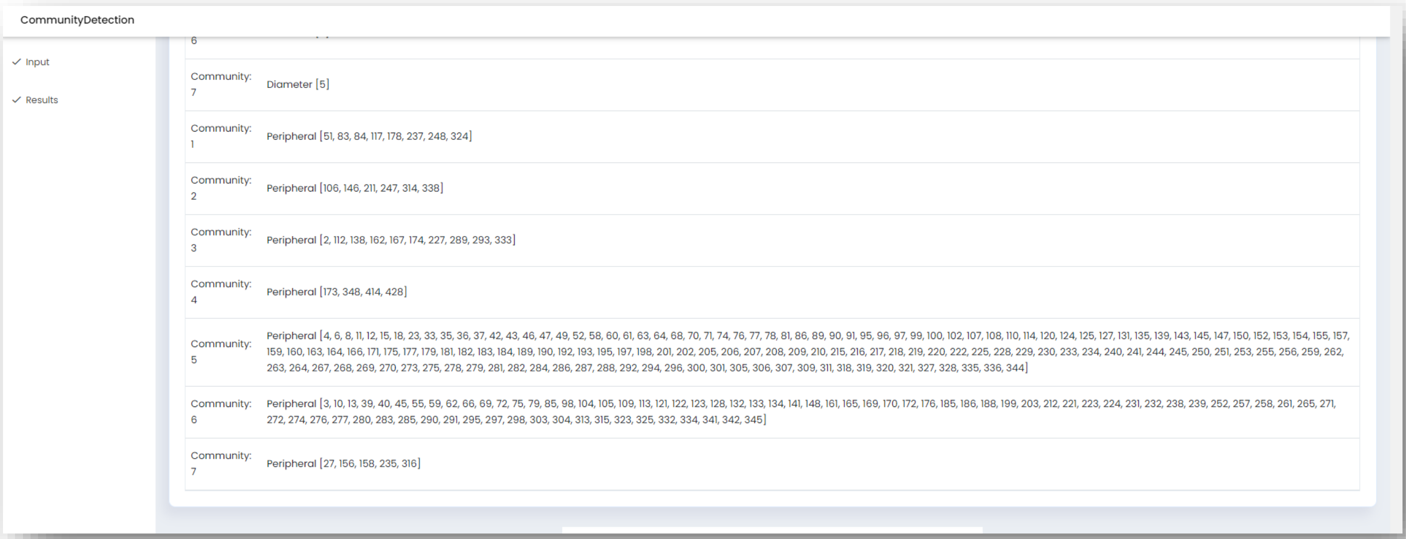


CommunityDetection

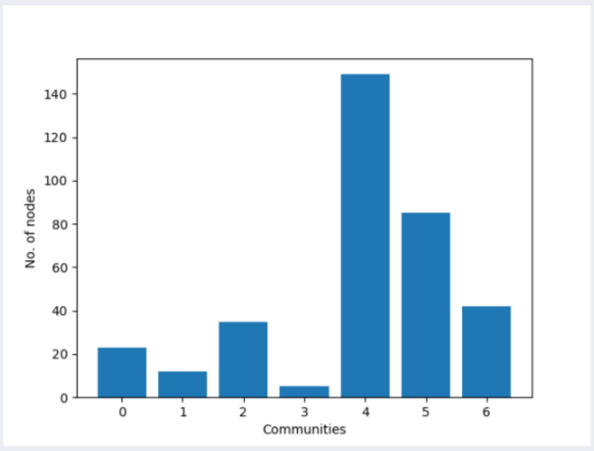
✓ Input

✓ Results

Community: 2	Center [16, 331]
Community: 3	Center [14, 17, 28, 41, 93, 137, 310, 337]
Community: 4	Center [34]
Community: 5	Center [0]
Community: 6	Center [9, 21, 25, 26, 56, 67, 142, 200]
Community: 7	Center [24, 30, 48, 53, 180, 204, 213]
Community: 1	Radius [2]
Community: 2	Radius [3]
Community: 3	Radius [3]
Community: 4	Radius [1]



6	
Community: 7	Diameter [5]
Community: 1	Peripheral [51, 83, 84, 117, 178, 237, 248, 324]
Community: 2	Peripheral [106, 146, 211, 247, 314, 338]
Community: 3	Peripheral [2, 112, 138, 162, 167, 174, 227, 289, 293, 333]
Community: 4	Peripheral [173, 348, 414, 428]
Community: 5	Peripheral [4, 6, 8, 11, 12, 15, 18, 23, 33, 35, 36, 37, 42, 43, 46, 47, 49, 52, 58, 60, 61, 63, 64, 68, 70, 71, 74, 76, 77, 78, 81, 86, 89, 90, 91, 95, 96, 97, 99, 100, 102, 107, 108, 110, 114, 120, 124, 125, 127, 131, 135, 139, 143, 145, 147, 150, 152, 153, 154, 155, 157, 159, 160, 163, 164, 166, 171, 175, 177, 179, 181, 182, 183, 184, 189, 190, 192, 193, 195, 197, 198, 201, 202, 205, 206, 207, 208, 209, 210, 215, 216, 217, 218, 219, 220, 222, 225, 228, 229, 230, 233, 234, 240, 241, 244, 245, 250, 251, 253, 255, 256, 259, 262, 263, 264, 267, 268, 269, 270, 273, 275, 278, 279, 281, 282, 284, 286, 287, 288, 292, 294, 296, 300, 301, 305, 306, 307, 309, 311, 318, 319, 320, 321, 327, 328, 335, 336, 344]
Community: 6	Peripheral [3, 10, 13, 39, 40, 45, 55, 59, 62, 66, 69, 72, 75, 79, 85, 98, 104, 105, 109, 113, 121, 122, 123, 128, 132, 133, 134, 141, 148, 161, 165, 169, 170, 172, 176, 185, 186, 188, 199, 203, 212, 221, 223, 224, 231, 232, 238, 239, 252, 257, 258, 261, 265, 271, 272, 274, 276, 277, 280, 283, 285, 290, 291, 295, 297, 298, 303, 304, 313, 315, 323, 325, 332, 334, 341, 342, 345]
Community: 7	Peripheral [27, 156, 158, 235, 316]



References:

- https://www.geeksforgeeks.org/detecting-communities-in-social-networks-using-girvan-newman-algorithm-in-python/
- https://www.analyticsvidhya.com/blog/2020/04/community-detection-graphs-networks/
- https://medium.com/analytics-vidhya/implement-louvain-community-detection-algorithm-using-python-and-gephi-with-visualization-871250fb2f25
- https://www.kaggle.com/lsjsj92/network-graph-with-louvain-algorithm
- https://pypi.org/project/python-louvain/
- https://pypi.org/project/matplotlib/
- https://pypi.org/project/Flask/
- https://pypi.org/project/networkx/
- https://pypi.org/project/community/
- https://pypi.org/project/communities/