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: 70
- Edges: 181
- Type: Undirected,unweighted

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

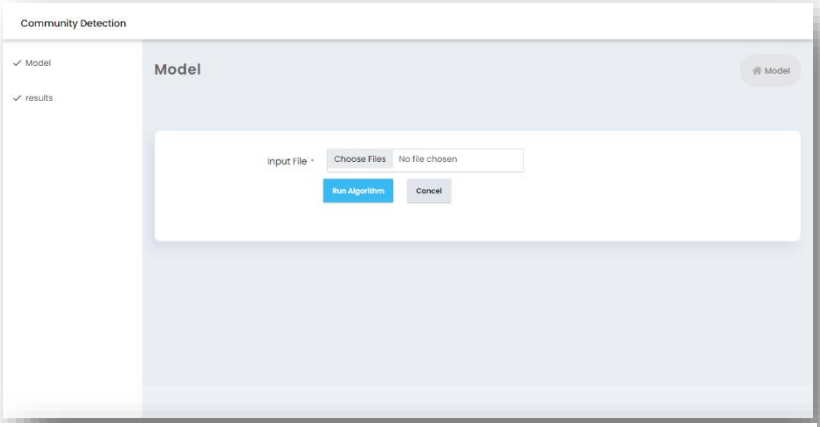
Community detection code:

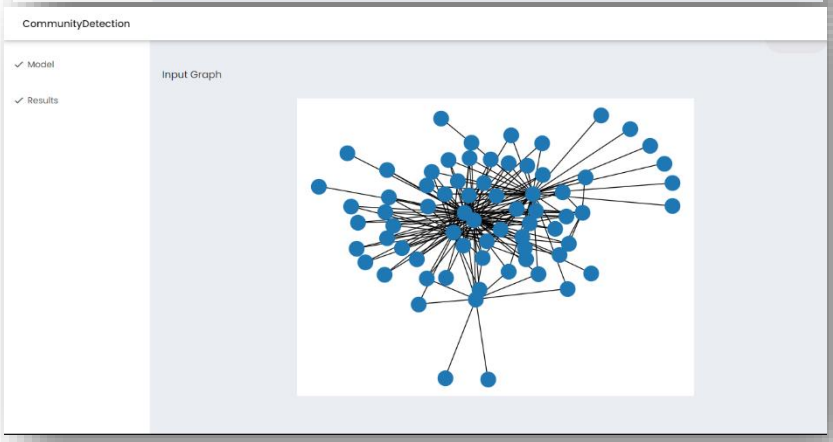
```
import os
  if request.method == 'POST':
   graph = request.files['input']
   graph.save(os.path.join("PATH TO DIRECTORY","graph.txt"))
   f="PATH TO DIRECTORY/graph.txt"
   from communities.algorithms import louvain_method,girvan_newman
   from communities.visualization import draw_communities
   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, ',')
   #matrix to array
   S= np.array(nx.to_numpy_matrix(G,dtype=int))
   print(S)
   #louvain
   communities, _ = louvain_method(S)
   draw_communities(S, communities,False,'PATH TO LOUVAIN IMAGE')
   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)
   #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]))
```

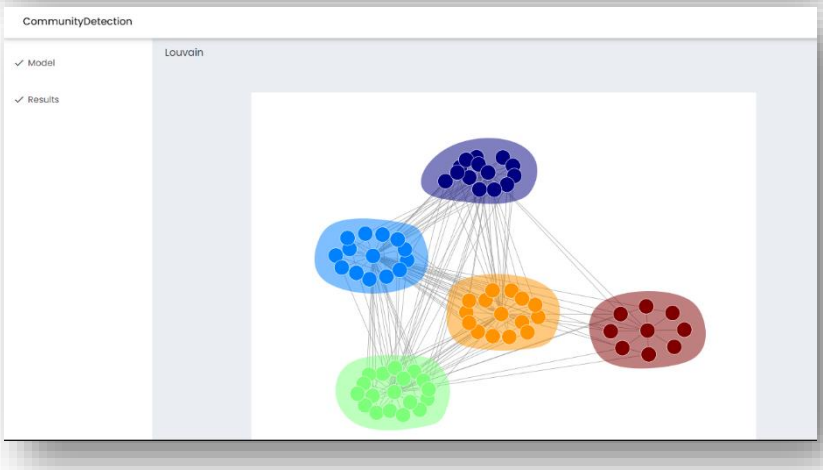
```
length=[]
for i in range(0,len(highest_degree)):
    length.append(i)
# draw the graph
pos = nx.spring_layout(L)
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(L, pos, partition.keys(), node_size=40,
cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(L, pos, alpha=0.5)
plt.savefig(‘PATH TO GIRVAN IMAGE’)
plt.clf()
plt.cla()
plt.close()
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
#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):
 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('PATH TO BARGRAPH IMAGE')
plt.clf()
plt.cla()
plt.close()
```

ANNEXURE II: Snapshots of the output





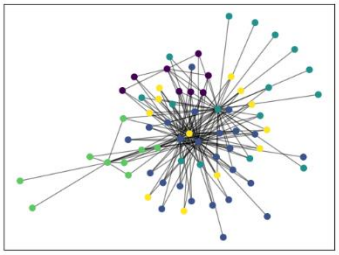


CommunityDetection

✓ Model

✓ Results

Girvan-newman



CommunityDetection

✓ Model

✓ Results

Details of Graph

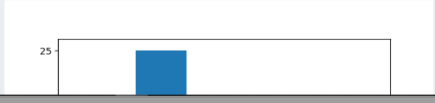
Total No. of communities	5
Community 1	[0, 14, 26, 37, 48, 52, 55, 56]
Community 2	[1, 3, 9, 19, 27, 28, 29, 30, 39, 42, 43, 47, 49, 50, 51, 53, 57, 62, 65, 67, 68, 71, 73, 59, 64]
Community 3	[2, 18, 21, 34, 35, 58, 66, 72, 6, 13, 32, 33, 38, 40, 45, 60]
Community 4	[5, 7, 17, 18, 20, 12, 15, 8, 10]
Community 5	[4, 22, 24, 25, 31, 41, 44, 54, 61, 63, 69, 70]
community 1 size:	8
community 2 size:	25
community 3 size:	16
community 4 size:	9

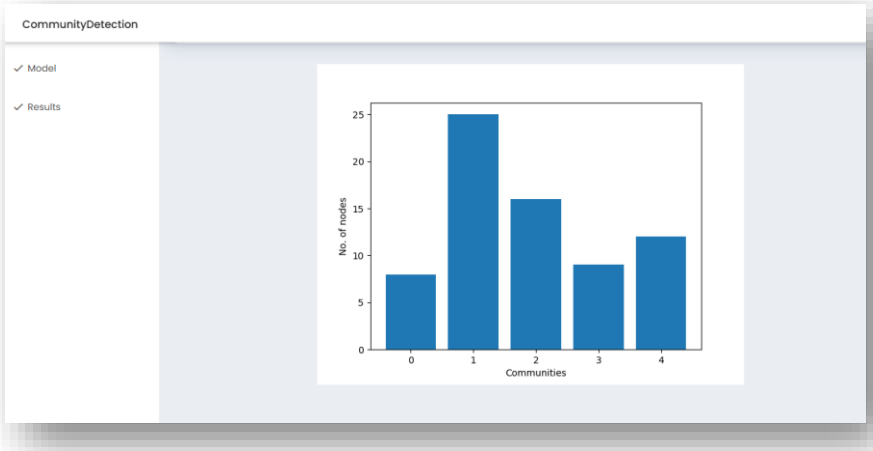
CommunityDetection

✓ Model

✓ Results

community 3 size:	16
community 4 size:	9
community 5 size:	12
Community: 1 , Node 0	Degree 8
Community: 2 , Node 1	Degree 53
Community: 3 , Node 2	Degree 42
Community: 4 , Node 5	Degree 11
Community: 5 , Node 4	Degree 44





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/