PSGCOLLEGEOFTECHNOLOGY

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

19OH01-SocialandEconomicNetworkAnalysis

Topic: Applied Graphical Network Analysis using Python

TeamMembers:

18z343	Punal Raj P
19z465	Abdul Kaiyum S
18z353	Shivesh Karthic P
18z319	Hariprasath R
18z338	Vishnu Vardhan Reddy

ProblemStatement:

We have analyzed the Nashville-meetup network to determine the following results
•Who are the people who most influence the network?
•Who are the people who influence the transfer of information?
•Which are the best performers in information transfer?
To assign roles and make categories between individuals we will calculate mathematical indicators from the theory of complex graphs:

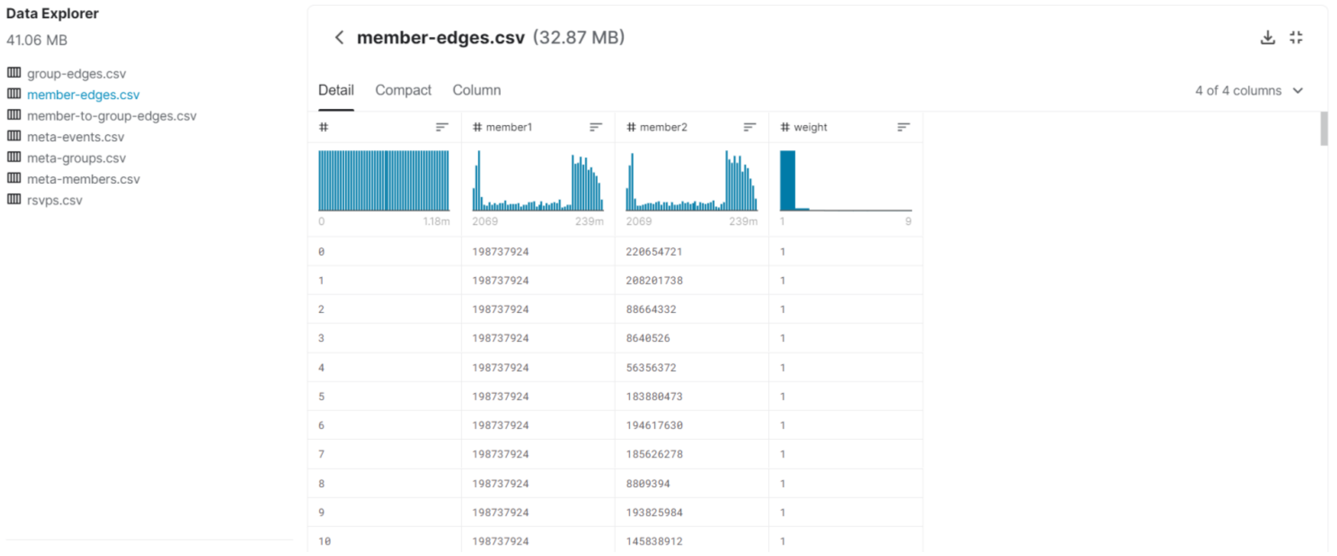
The centrality of proximity: This indicator makes it possible to detect the individuals who have a significant power on the transfer of information. Individuals with a large centralized proximity have the ability to contact a very large number of individuals easily

The betweenness centrality: This indicator can detect individuals who influence the transfer of information. If these individuals do not exist in the network, then the information can not flow on both sides of the network.

The eigenvector centrality: The individuals having a high spectral centralized are the individuals who have the most relation in the network, they are central and have influence in a general way on the network.

Dataset:

- Description: meetup.com is a website for people organizing and attending regular or semi-regular events ("meet-ups"). The relationships amongst users—who goes to what meetups—are a social network, ideal for graph-based analysis.
- Dataset Statistics: member-edges.csv: Edge list for constructing a member-to-member graph. Weights represent shared group membership.



#	member1	member2	weight
0	198737924	220654721	1
1	198737924	208201738	1
2	198737924	88664332	1
3	198737924	8640526	1
4	198737924	56356372	1
5	198737924	183880473	1
6	198737924	194617630	1
7	198737924	185626278	1
8	198737924	8809394	1
9	198737924	193825984	1
10	198737924	145838912	1

- DatasetLink:https://www.kaggle.com/stkbailey/nashville-meetup?select=member-edges.csv

Toolsused:

- **Python:** We have used the Python Language for the coding part because of itsUser-friendlyDataStructures.
- **NetworkX:**NetworkX is the most popular Python package for manipulating andanalyzinggraphs.NetworkXissuitableforreal-worldgraph problemsandisgoodathandlingbigdataas well.
- **Jupyter notebook**:The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

ChallengesFaced:

- There was some error in the code, so we were unable to visualize the graph initially.
- Even though our code was debugged and ran, the expected output in terms of degree and centrality were all 0.
- Since we were new to NetworkX and Jupyter notebook, it was tiring to understand and visualize the graphs.

ContributionofTeamMembers:

RollNo:	Name	Contribution
18Z319	Hari Prasath	Project idea
18Z343	Punal Raj P	Graph visualization and coding
18Z338	Vishnu Vardhan Reddy	Collecting data set
18Z353	Shivesh Karthic P	Graph analysis and coding
19Z465	Abdul Kaiyum S	Documentation

AnnexureI:Code:

Libraries Used:

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import matplotlib.patches as mpatches
        import seaborn as sns
        import numpy as np
        plt.style.use('fivethirtyeight')

        ## Network
        import networkx as nx
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import pylab as plt
        from itertools import count
        from operator import itemgetter
        from networkx.drawing.nx_agraph import graphviz_layout
        import pylab
```

Members in Dataset:

```
In [2]: df = pd.read_csv('member-edges.csv')
        print(len(df))

        1176368
```

Limiting the Dataset:

```
In [3]: df = df[0 : 1000]
```

Graph Visualization:

```
In [4]: pd.set_option('precision',10)
        G = nx.from_pandas_edgelist(df, 'member1', 'member2', create_using = nx.Graph())

        nodes = G.nodes()
        degree = G.degree()
        colors = [degree[n] for n in nodes]
        size = [(degree[n]) for n in nodes]

        pos = nx.kamada_kawai_layout(G)
        #pos = nx.spring_layout(G, k = 0.2)
        cmap = plt.cm.viridis_r
        cmap = plt.cm.Greys

        vmin = min(colors)
        vmax = max(colors)

        fig = plt.figure(figsize = (15,9), dpi=100)

        nx.draw(G,pos,alpha = 0.8, nodelist = nodes, node_color = 'w', node_size = 10, with_labels= False,font_size = 6, width = 0.2, cma
        fig.set_facecolor('#0B243B')

        plt.legend()
        plt.show()
```

Finding Degree:

```
In [5]: for i in sorted(G.nodes()):
            G.nodes[i]['Degree'] = G.degree(i)
```

```
In [6]: nodes_data = pd.DataFrame([i[1] for i in G.nodes(data=True)], index=[i[0] for i in G.nodes(data=True)])
        nodes_data = nodes_data.sort_values(by = ['Degree'], ascending = False)
        nodes_data.index.names=['ID']
        nodes_data.reset_index(level=0, inplace=True)
```

Finding Betweenness Centrality:

```
In [7]: bet_cen = nx.betweenness_centrality(G)
        df_bet_cen = pd.DataFrame.from_dict(bet_cen, orient='index')
        df_bet_cen.columns = ['betweenness_centrality']
        df_bet_cen.index.names = ['ID']
        df_bet_cen.reset_index(level=0, inplace=True)
        analyse= pd.merge(nodes_data,df_bet_cen, on = ['ID'])
```

Finding Clustering Coefficient:

```
In [8]: clust_coefficients = nx.clustering(G)
        df_clust = pd.DataFrame.from_dict(clust_coefficients, orient='index')
        df_clust.columns = ['clust_coefficient']
        df_clust.index.names = ['ID']
        df_clust.reset_index(level=0, inplace=True)
        analyse= pd.merge(analyse, df_clust, on = ['ID'])
```

Finding Closeness Centrality:

```
In [9]: clo_cen = nx.closeness_centrality(G)
        df_clo = pd.DataFrame.from_dict(clo_cen, orient='index')
        df_clo.columns = ['closeness_centrality']
        df_clo.index.names = ['ID']
        df_clo.reset_index(level=0, inplace=True)
        analyse= pd.merge(analyse, df_clo, on = ['ID'])
```

Finding Eigenvector Centrality:

```
In [10]: eig_cen = nx.eigenvector_centrality_numpy(G)
         df_eig = pd.DataFrame.from_dict(eig_cen, orient='index')
         df_eig.columns = ['eigenvector_centrality']
         df_eig.index.names = ['ID']
         df_eig.reset_index(level=0, inplace=True)
         analyse= pd.merge(analyse, df_eig, on = ['ID'])

         print(analyse)
```

AnnexureII:SnapshotsoftheOutput:

Printing the dataset for first 1000 nodes:

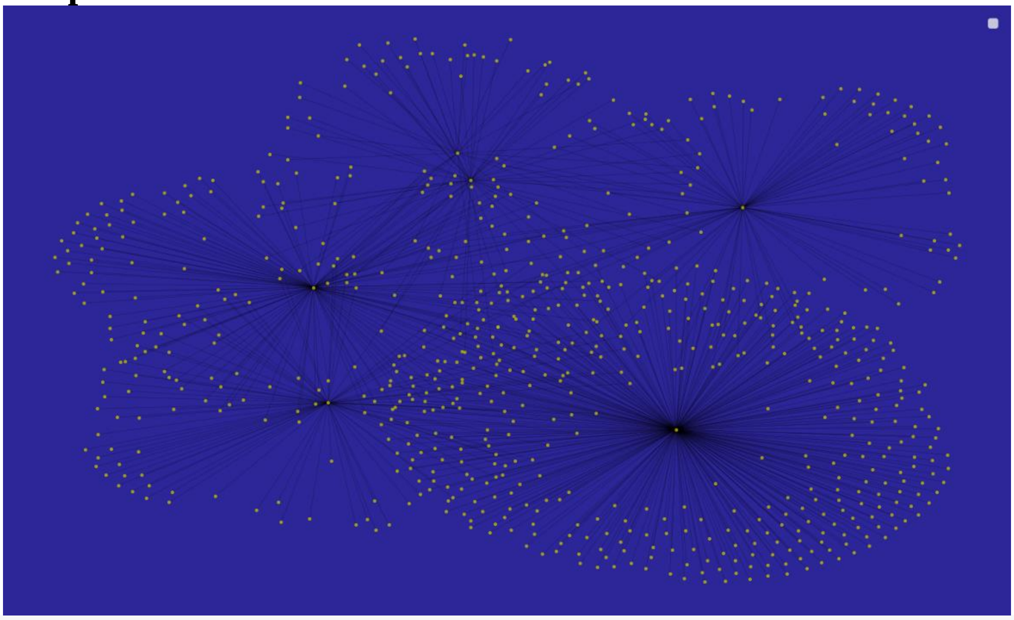
```
In [11]: df.iloc[1:1000]
```

Out[11]:

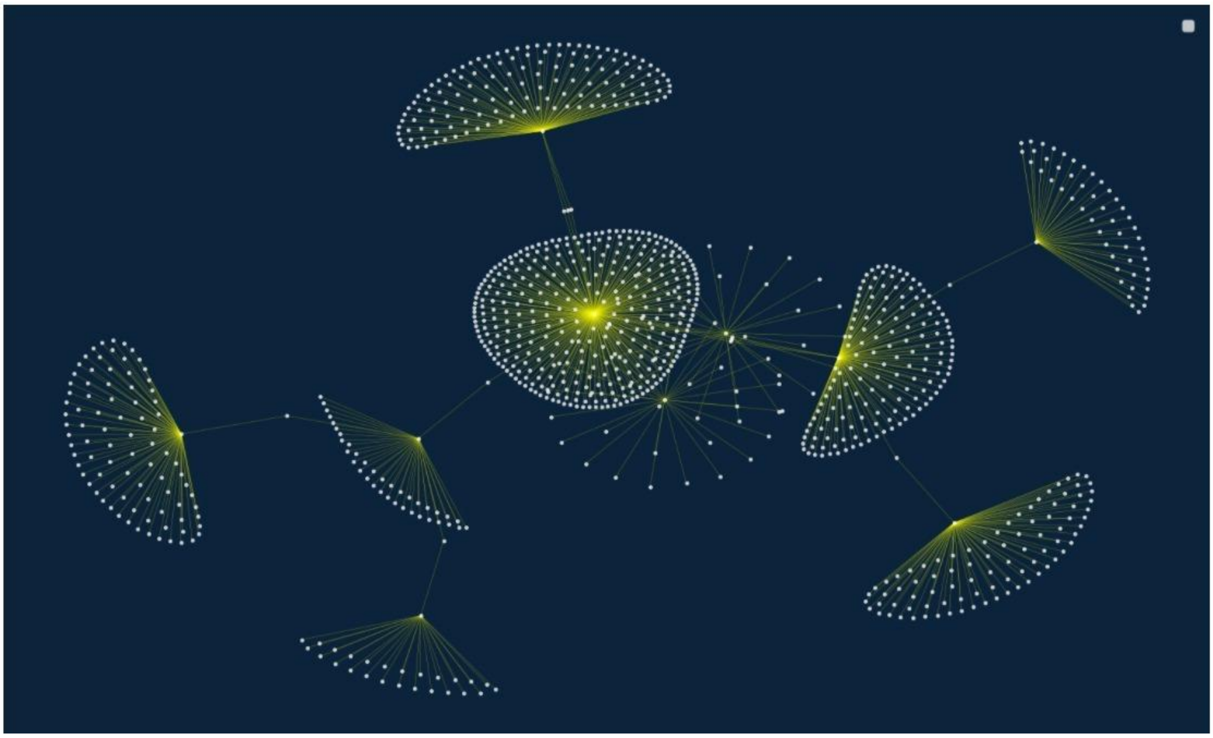
	Unnamed: 0	member1	member2	weight
1	1	198737924	208201738	1
2	2	198737924	88664332	1
3	3	198737924	8640526	1
4	4	198737924	56356372	1
5	5	198737924	183880473	1
...	...	...	...	...
995	995	226754592	237417427	1
996	996	226754592	216892372	1
997	997	226754592	220648421	1
998	998	226754592	220916721	1
999	999	226754592	231246833	1

999 rows × 4 columns

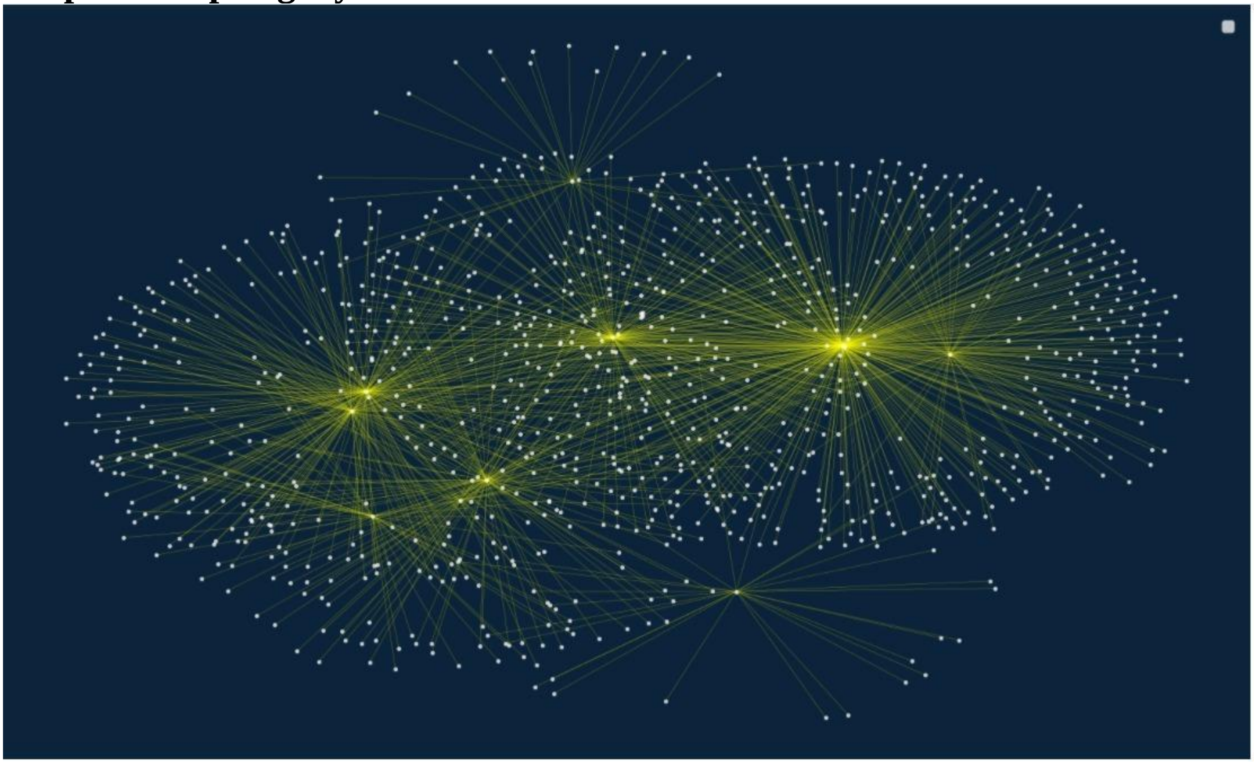
Graph visualization for 750 nodes:



Graph visualization for 1000 nodes(kamadakawailayout):



Graph with spring layout:



Analysis of the Dataset:

	ID	Degree	betweenness_centrality	clust_coefficient
0	234684445	359	0.7569401592	0
1	226754592	154	0.4537892234	0
2	216072216	137	0.2350066231	0
3	183566364	88	0.1566864923	0
4	73498632	88	0.1566864923	0
..	...	...	...	...
994	39322832	1	0.0000000000	0
995	12771542	1	0.0000000000	0
996	55746782	1	0.0000000000	0
997	174031072	1	0.0000000000	0
998	231246833	1	0.0000000000	0

	closeness_centrality	eigenvector_centrality
0	0.3401347451	0.7069631803
1	0.2674028213	0.0103417366
2	0.2247704345	0.0095497906
3	0.1641094980	0.0000081731
4	0.1829656420	0.0000381491
..	...	...
994	0.1397806665	0.0000004313
995	0.1397806665	0.0000004313
996	0.1397806665	0.0000004313
997	0.1397806665	0.0000004313
998	0.2083225274	0.0005457494

[999 rows x 6 columns]

Reference:

- Reference Links:
  - https://towardsdatascience.com/applied-network-analysis-using-python-25021633a702
  - https://www.kaggle.com/stkbailey/nashville-meetup
- Downloading Packages:https://www.youtube.com/watch?v=FKwicZF7xNE
- Tutorials:
  - https://www.youtube.com/watch?v=flwcAf1_1RU
  - https://www.youtube.com/watch?v=PouhDHfssYA
- **Plagiarism Report:**https://smallseotools.com/view-report/8078d54a9e88581dc8054392a61e872c