19CSE356 - Social Network Analytics Assignment 1

by Veeramanohar A - CB.EN.U4EEE19145 - EEE -B

In [1]:

```
import networkx as nx
```

Importing Dataset

In [16]:

```
G = nx.karate_club_graph()
```

In [19]:

```
G.name
```

Out[19]:

```
"Zachary's Karate Club"
```

Exploring the Dataset

In [33]:

```
def get_insights(G):
    print(f'The given "{G.name}" graph has {len(G.nodes())} nodes and {len(G.nodes
())} edges.')
    print(f'Density: {round(nx.density(G),2)}')
    print(f'Average Shortest Path Length:
{round(nx.average_shortest_path_length(G),2)}')
    print(f'Diameter: {round(nx.diameter(G),2)}')
    print(f'{sorted(nx.degree(G),key=lambda x:x[1],reverse=True)[0][0]} is the nod
e with highest degree of {sorted(nx.degree(G),key=lambda x:x[1],reverse=True)[0][1
]}.')
```

In [34]:

```
get_insights(G)
```

```
The given "Zachary's Karate Club" graph has 34 nodes and 34 edges.
Density: 0.14
Average Shortest Path Length: 2.41
Diameter: 5
33 is the node with highest degree of 17.
```

In [17]:

```
print(f'No. of nodes: {len(G.nodes())} \nNo. of edges: {len(G.edges())}')
```

```
No. of nodes: 34
No. of edges: 78
```

In [3]:

```
nx.nodes(G)
```

Out[3]:

```
NodeView((0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33))
```

In [4]:

```
nx.edges(G)
```

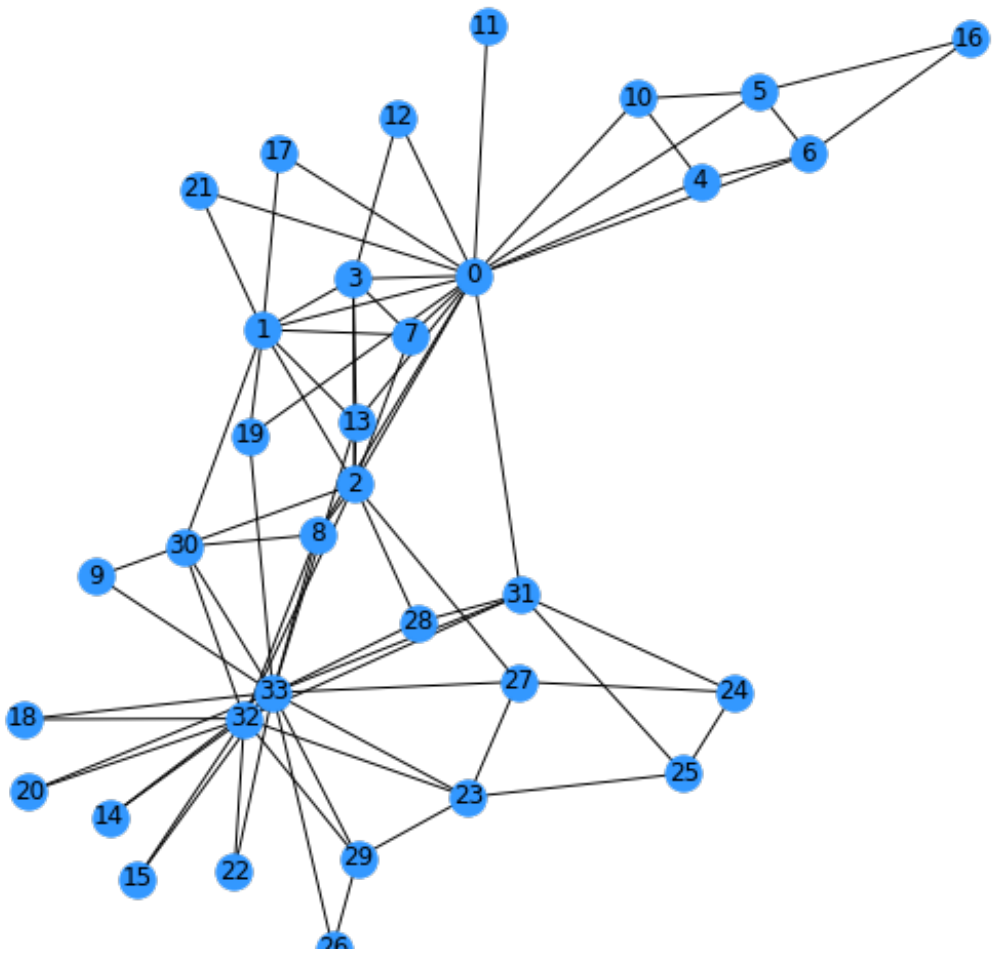
Out[4]:

```
EdgeView([(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 10)
, (0, 11), (0, 12), (0, 13), (0, 17), (0, 19), (0, 21), (0, 31), (1, 2), (1, 3),
(1, 7), (1, 13), (1, 17), (1, 19), (1, 21), (1, 30), (2, 3), (2, 7), (2, 8), (2,
9), (2, 13), (2, 27), (2, 28), (2, 32), (3, 7), (3, 12), (3, 13), (4, 6), (4, 10)
, (5, 6), (5, 10), (5, 16), (6, 16), (8, 30), (8, 32), (8, 33), (9, 33), (13,
33), (14, 32), (14, 33), (15, 32), (15, 33), (18, 32), (18, 33), (19, 33), (20, 32
), (20, 33), (22, 32), (22, 33), (23, 25), (23, 27), (23, 29), (23, 32), (23, 33),
(24, 25), (24, 27), (24, 31), (25, 31), (26, 29), (26, 33), (27, 33), (28, 31), (2
8, 33), (29, 32), (29, 33), (30, 32), (30, 33), (31, 32), (31, 33), (32, 33)])
```

Visualization

In [5]:

```
import matplotlib.pyplot as plt
layout = nx.fruchterman_reingold_layout(G)
plt.figure(figsize=(10,10))
plt.axis("off")
nx.draw_networkx(G, layout, with_labels=True, node_color = '#3398ff' )
```



In [6]:

```
nx.density(G)
```

Out[6]:

```
0.13903743315508021
```

In [7]:

```
nx.average_shortest_path_length(G)
```

Out[7]:

```
2.408199643493761
```

In [8]:

```
nx.diameter(G)
```

Out[8]:

```
5
```

Community Detection

In [9]:

```
from networkx.algorithms.community import greedy_modularity_communities
greedy_modularity_communities(G)
```

Out[9]:

```
[frozenset({8,
            14,
            15,
            18,
            20,
            22,
            23,
            24,
            25,
            26,
            27,
            28,
            29,
            30,
            31,
            32,
            33}),
 frozenset({1, 2, 3, 7, 9, 12, 13, 17, 21}),
 frozenset({0, 4, 5, 6, 10, 11, 16, 19})]
```

Connected Component

In [10]:

```
nx.number_connected_components(G)
```

Out[10]:

```
1
```

In [11]:

```
nx.degree(G)
```

Out[11]:

```
DegreeView({0: 16, 1: 9, 2: 10, 3: 6, 4: 3, 5: 4, 6: 4, 7: 4, 8: 5, 9: 2, 10: 3, 1
1: 1, 12: 2, 13: 5, 14: 2, 15: 2, 16: 2, 17: 2, 18: 2, 19: 3, 20: 2, 21: 2, 22:
2, 23: 5, 24: 3, 25: 3, 26: 2, 27: 4, 28: 3, 29: 4, 30: 4, 31: 6, 32: 12, 33:
17})
```

In [12]:

```
sorted(nx.degree(G),key=lambda x:x[1],reverse=True)
```

Out[12]:

```
[(33, 17),
 (0, 16),
 (32, 12),
 (2, 10),
 (1, 9),
 (3, 6),
 (31, 6),
 (8, 5),
 (13, 5),
 (23, 5),
 (5, 4),
 (6, 4),
 (7, 4),
 (27, 4),
 (29, 4),
 (30, 4),
 (4, 3),
 (10, 3),
 (19, 3),
 (24, 3),
 (25, 3),
 (28, 3),
 (9, 2),
 (12, 2),
 (14, 2),
 (15, 2),
 (16, 2),
 (17, 2),
 (18, 2),
 (20, 2),
 (21, 2),
 (22, 2),
 (26, 2),
 (11, 1)]
```

Centrality

Betweeness Centrality

In [13]:

```
sorted(nx.betweenness_centrality(G, normalized=True).items(), key=lambda x:x[1], r
everse=True)[0:10]
```

Out[13]:

```
[(0, 0.43763528138528146),
 (33, 0.30407497594997596),
 (32, 0.145247113997114),
 (2, 0.14365680615680618),
 (31, 0.13827561327561325),
 (8, 0.05592682780182781),
 (1, 0.053936688311688304),
 (13, 0.04586339586339586),
 (19, 0.03247504810004811),
 (5, 0.02998737373737374)]
```

Closeness Centrality

In [14]:

```
sorted(nx.closeness_centrality(G).items(), key=lambda x:x[1], reverse=True)[0:10]
```

Out[14]:

```
[(0, 0.5689655172413793),
 (2, 0.559322033898305),
 (33, 0.55),
 (31, 0.5409836065573771),
 (8, 0.515625),
 (13, 0.515625),
 (32, 0.515625),
 (19, 0.5),
 (1, 0.4852941176470588),
 (3, 0.4647887323943662)]
```

Eigenvector Centrality

In [15]:

```
sorted(nx.eigenvector_centrality(G).items(), key=lambda x:x[1], reverse=True)[0:10
]
```

Out[15]:

```
[(33, 0.373371213013235),
 (0, 0.3554834941851943),
 (2, 0.31718938996844476),
 (32, 0.3086510477336959),
 (1, 0.2659538704545025),
 (8, 0.2274050914716605),
 (13, 0.22646969838808148),
 (3, 0.2111740783205706),
 (31, 0.19103626979791702),
 (30, 0.17476027834493085)]
```