

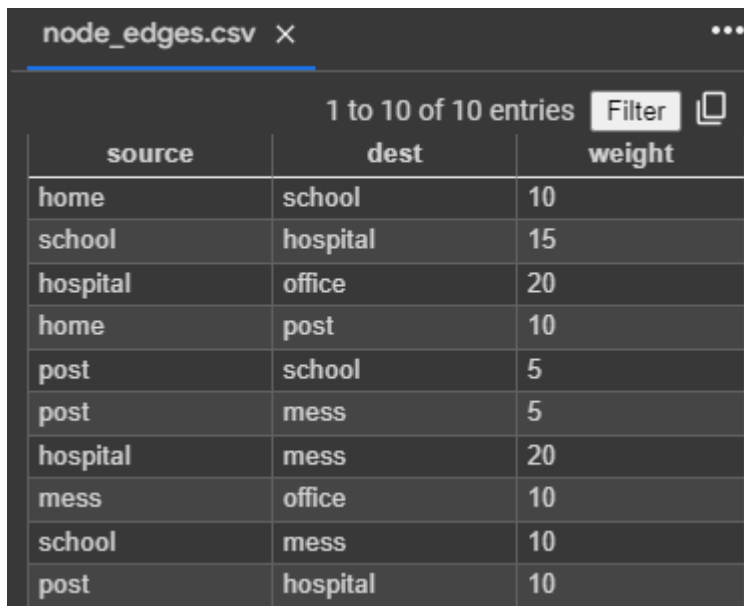
NAME: G.SAI SUSHANTH REDDY

REG.NO: 22MCB0005

SUBJECT: SOCIAL NETWORK ANALYTICS (MCSE618P)

ASSESSMENT-1

nodes_edges.csv



source	dest	weight
home	school	10
school	hospital	15
hospital	office	20
home	post	10
post	school	5
post	mess	5
hospital	mess	20
mess	office	10
school	mess	10
post	hospital	10

We use import network as nx (library in python) to create directed and undirected graphs.

```
# DIGITAL ASSESSMENT 1
# REG.NO 22MCB0005
import csv
import pandas as pd

#with open('/content/node_edges.csv', 'r') as f:
#    reader = csv.reader(f)
#    edges = list(reader)

df = pd.read_csv('/content/node_edges.csv')

#For Directed Graph
G = nx.from_pandas_edgelist(df, source='source', target='dest', edge_attr='weight', create_using=nx.DiGraph())

nx.draw(G,with_labels=1,font_weight="bold")

G.edges()
G.nodes()

NodeView(('home', 'school', 'hospital', 'office', 'post', 'mess'))
```

```
#For Undirected Graph
G1 = nx.from_pandas_edgelist(df, source='source', target='dest', edge_attr='weight', create_using=nx.Graph())

nx.draw(G1, with_labels=2, font_weight="bold")
```

Adjacency Matrix:

```
[44] # Adjacency matrix of the directed and undirected graph
A = nx.to_numpy_array(G)
print(A)

[[ 0. 10.  0.  0. 10.  0.]
 [ 0.  0. 15.  0.  0. 10.]
 [ 0.  0.  0. 20.  0. 20.]
 [ 0.  0.  0.  0.  0.  0.]
 [ 0.  5. 10.  0.  0.  5.]
 [ 0.  0.  0. 10.  0.  0.]]
```

Row wise and Column wise summation of the matrix:

```
# Row Wise and Column Wise summation for directed graph and undirected graph
row_sum = np.sum(A, axis=1)
print("Row-wise Sum:", row_sum)

col_sum = np.sum(A, axis=0)
print("Column-wise Sum:", col_sum)

Row-wise sum: [20. 25. 40.  0. 20. 10.]
Column-wise sum: [ 0. 15. 25. 30. 10. 35.]
```

Degree Centrality is a measure of the importance of a node in a graph based on the number of edges (or connections) it has to other nodes. The degree of a node is simply the number of edges that are incident on that node.

Degree Centrality = degree of node / (number of nodes in the graph - 1)

Nodes with Max and Min Degree and Degree Centrality:

```
[48] #Number of nodes with max and min degree

degrees = [G.degree(n) for n in G.nodes()]
max_degree = max(degrees)
min_degree = min(degrees)
print(max_degree)
print(min_degree)

nx.degree_centrality(G) # Degree Centrality

4
2
{'home': 0.4,
 'school': 0.8,
 'hospital': 0.8,
 'office': 0.4,
 'post': 0.8,
 'mess': 0.8}
```

In and Out Degree Centrality:

```
[49] nx.in_degree_centrality(G) # In Degree Centrality

{'home': 0.0,
 'school': 0.4,
 'hospital': 0.4,
 'office': 0.4,
 'post': 0.2,
 'mess': 0.6000000000000001}

[50] nx.out_degree_centrality(G) # Out Degree Centrality

{'home': 0.4,
 'school': 0.4,
 'hospital': 0.4,
 'office': 0.0,
 'post': 0.6000000000000001,
 'mess': 0.2}
```

Closeness centrality measures the average distance between a node and all other nodes in the network. Nodes with high closeness centrality are closer to all other nodes and can more easily communicate with them.

Betweenness centrality measures the extent to which a node lies on the shortest path between other nodes in the network. Nodes with high betweenness centrality act as bridges between different parts of the network and are important for maintaining the overall connectivity of the network.

Eigen vector centrality is a measure of the influence of a node in a network based on the concept of eigenvectors from linear algebra. It assigns a score to each node in the network based on its connections to other high-scoring nodes. Nodes that are connected to other nodes with high centrality scores themselves have higher centrality scores.

```
▶ nx.closeness_centrality(G) # Closeness Centrality
```

```
{'home': 0.0,  
 'school': 0.4,  
 'hospital': 0.44999999999999996,  
 'office': 0.5555555555555556,  
 'post': 0.2,  
 'mess': 0.6400000000000001}
```

```
[52] nx.betweenness_centrality(G) #Betweenness Centrality
```

```
{'home': 0.0,  
 'school': 0.07500000000000001,  
 'hospital': 0.07500000000000001,  
 'office': 0.0,  
 'post': 0.07500000000000001,  
 'mess': 0.07500000000000001}
```

```
[54] nx.eigenvector_centrality_numpy(G) # Eigen Vector Centrality
```

```
{'home': 4.268807529683727e-14,  
 'school': 9.478089956329108e-09,  
 'hospital': 4.478640829419511e-06,  
 'office': 0.999997760662034,  
 'post': 2.005835775253928e-11,  
 'mess': 0.0021162823202247327}
```

PageRank of the Graph:

PageRank is a measure of the importance or popularity of a node based on the links pointing to it. The basic idea is that a page is important if other important pages link to it.

```
[59] # Calculating the PageRank of the graph
pr = nx.pagerank(G)

# Print the PageRank of each node in the graph
for node, score in pr.items():
    print(f"Node {node}: {score}")

Node home: 0.07067737896683282
Node school: 0.12211784920394161
Node hospital: 0.17576220405346124
Node office: 0.32242854717151476
Node post: 0.10071561231222104
Node mess: 0.20829840829202825
```

In Degree and Out Degree (Min and Max):

In-degree and out-degree are measures of the number of incoming and outgoing edges, respectively, that are incident on a node in a directed graph. The in-degree of a node is the number of edges that point to the node, while the out-degree of a node is the number of edges that emanate from the node.

The minimum and maximum in-degree and out-degree of a graph refer to the minimum and maximum values of in-degree and out-degree across all nodes in the graph.

In-degree and out-degree measures can be used to analyze the connectivity and centrality of nodes in a graph and can help identify important nodes or patterns in the graph.

```
[27] # maximum and minimum in-degree and out-degree in the graph
```

```
in_degrees = [G.in_degree(n) for n in G.nodes()]
max_in_degree = max(in_degrees)
min_in_degree = min(in_degrees)

out_degrees = [G.out_degree(n) for n in G.nodes()]
max_out_degree = max(out_degrees)
min_out_degree = min(out_degrees)

print(max_in_degree)
print(min_in_degree)
print(max_out_degree)
print(min_out_degree)
```

```
3
0
3
0
```