NAME: G.SAI SUSHANTH REDDY

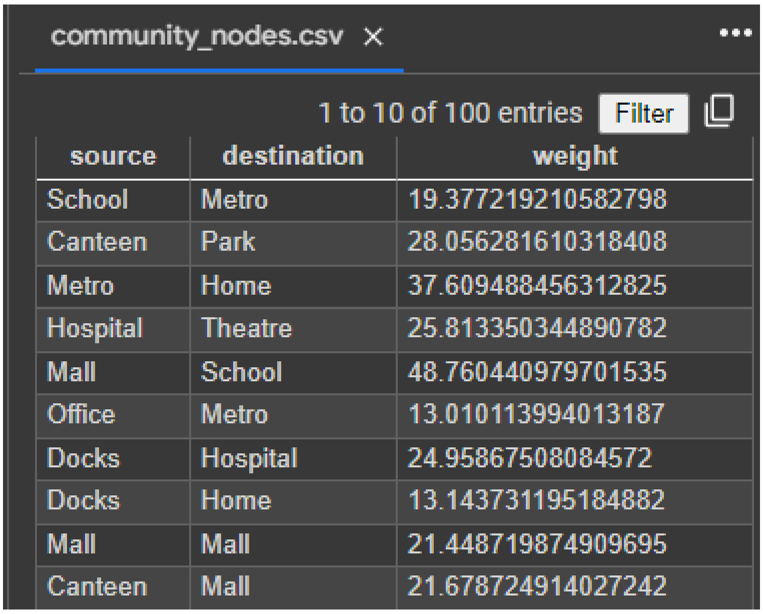
REG.NO: 22MCB0005

SUBJECT: SOCIAL NETWORK ANALYTICS

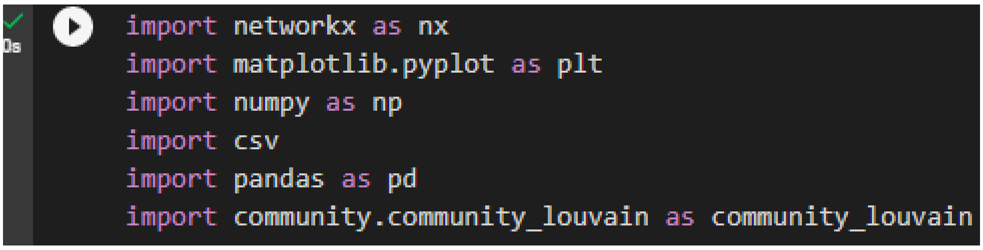
ASSESSMENT-2
(REPORT)

Implementing Community Detection Algorithms

Considering the dataset community_nodes.csv to analyse and detect the communities.

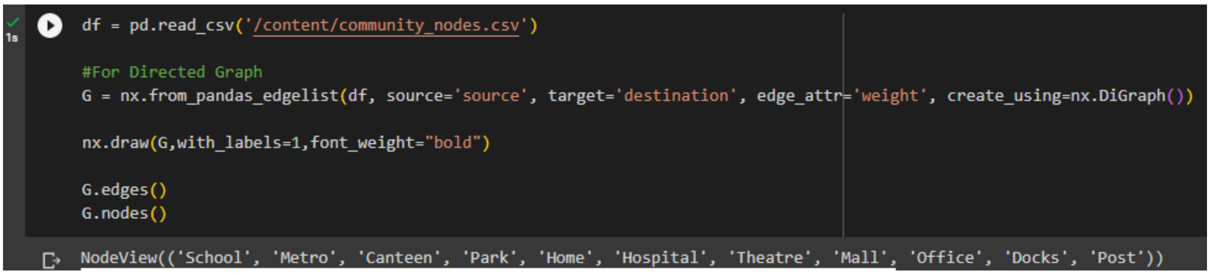


source	destination	weight
School	Metro	19.377219210582798
Canteen	Park	28.056281610318408
Metro	Home	37.609488456312825
Hospital	Theatre	25.813350344890782
Mall	School	48.760440979701535
Office	Metro	13.010113994013187
Docks	Hospital	24.95867508084572
Docks	Home	13.143731195184882
Mall	Mall	21.448719874909695
Canteen	Mall	21.678724914027242



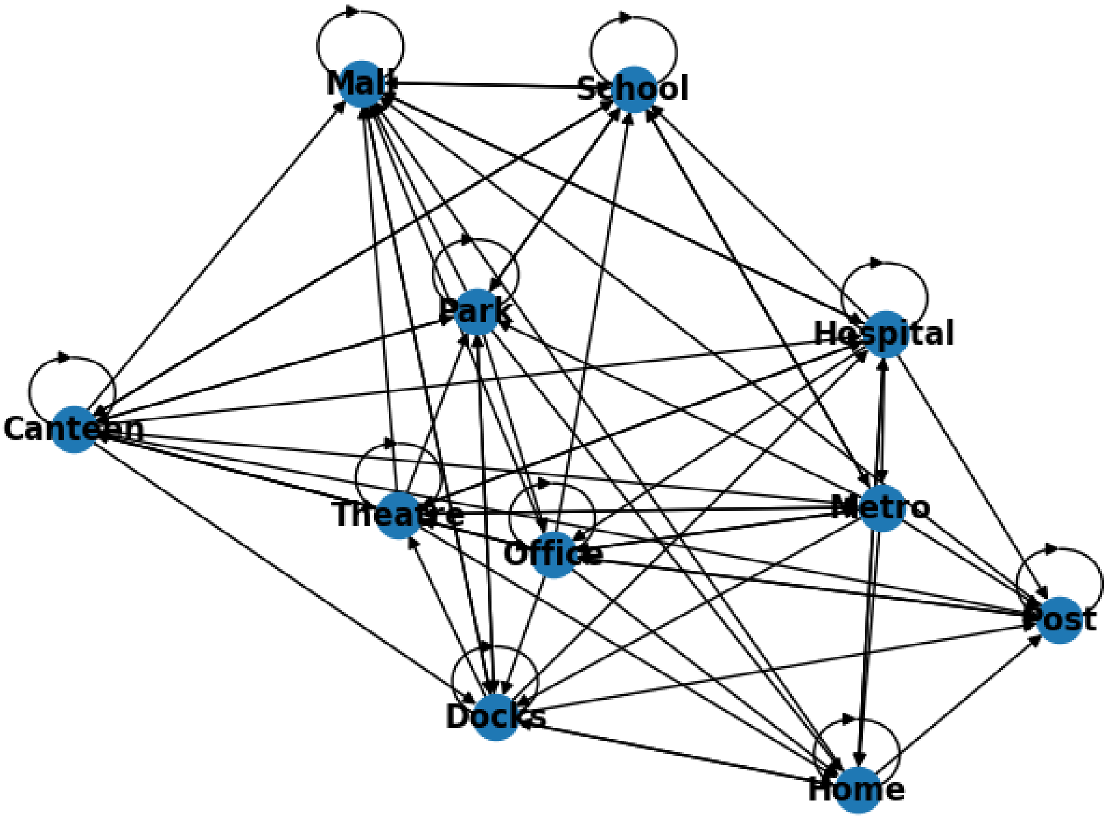
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import csv
import pandas as pd
import community.community_louvain as community_louvain
```

Plotting the Directed Graph:



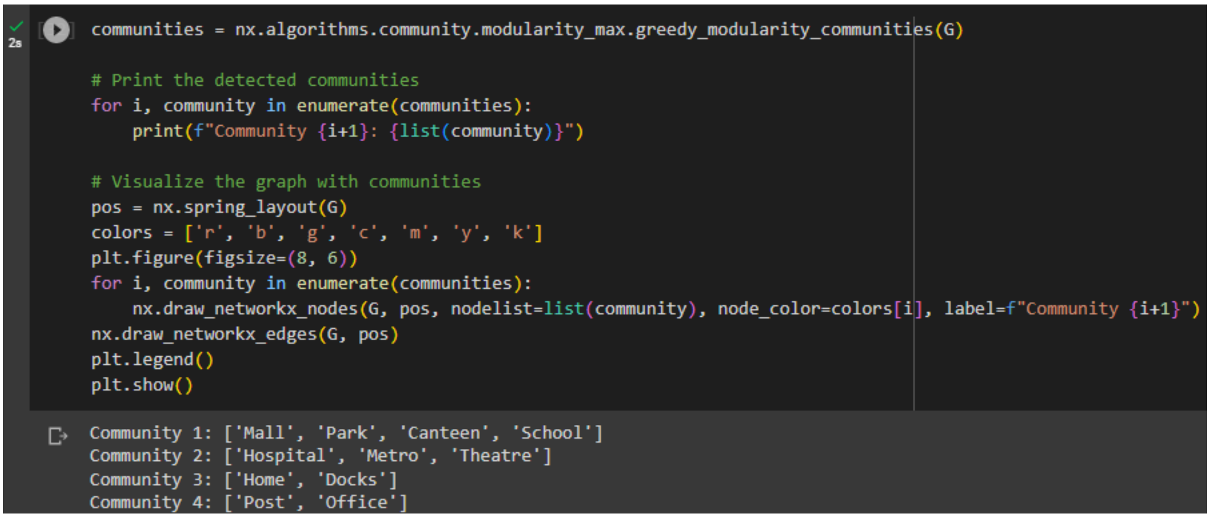
```
df = pd.read_csv('/content/community_nodes.csv')

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

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

G.edges()
G.nodes()
```

```
NodeView(('School', 'Metro', 'Canteen', 'Park', 'Home', 'Hospital', 'Theatre', 'Mall', 'Office', 'Docks', 'Post'))
```



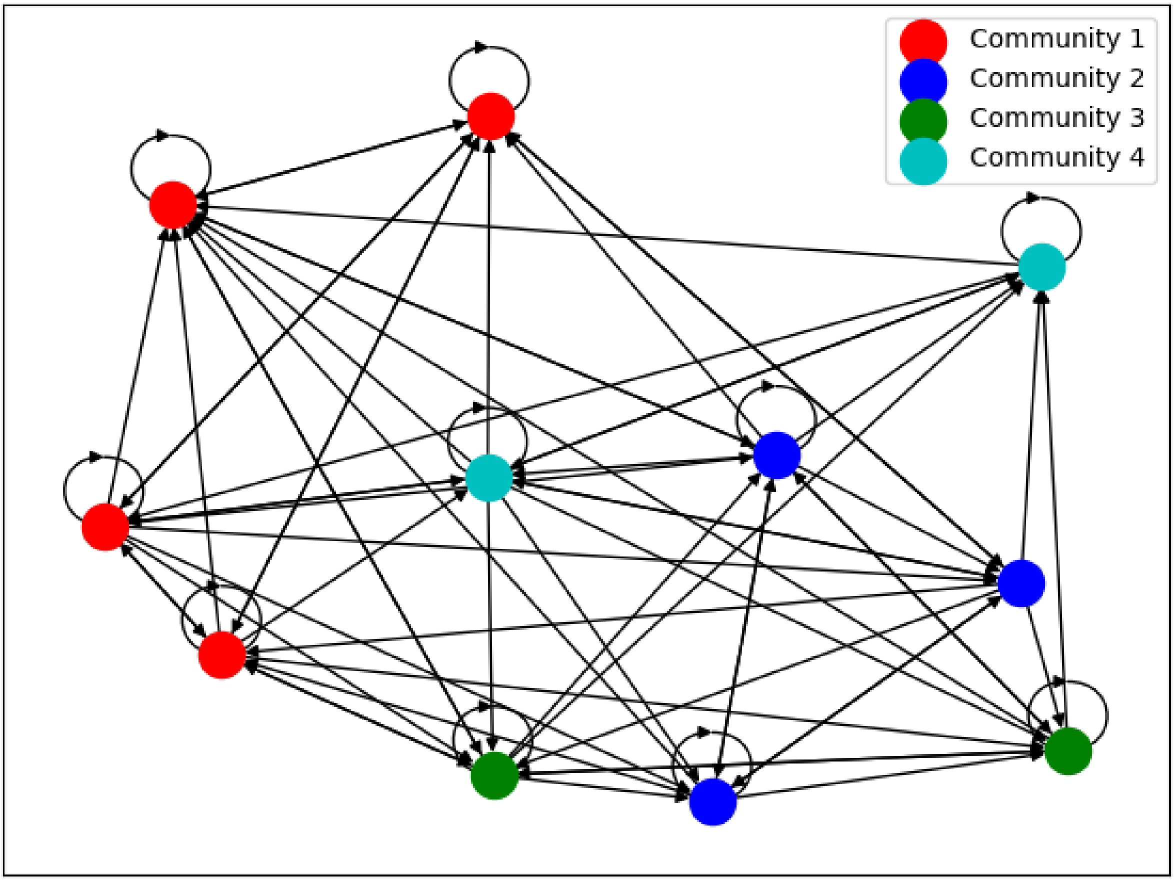
We have used this function called **Greedy Modularity Maximization** to find the community partition with the largest modularity.

Greedy modularity maximization begins with each node in its own community and repeatedly joins the pair of communities that lead to the largest modularity until no further increase in modularity is possible.



```
communities = nx.algorithms.community.modularity_max.greedy_modularity_communities(G)

# Print the detected communities
for i, community in enumerate(communities):
    print(f"Community {i+1}: {list(community)}")

# Visualize the graph with communities
pos = nx.spring_layout(G)
colors = ['r', 'b', 'g', 'c', 'm', 'y', 'k']
plt.figure(figsize=(8, 6))
for i, community in enumerate(communities):
    nx.draw_networkx_nodes(G, pos, nodelist=list(community), node_color=colors[i], label=f"Community {i+1}")
nx.draw_networkx_edges(G, pos)
plt.legend()
plt.show()
```

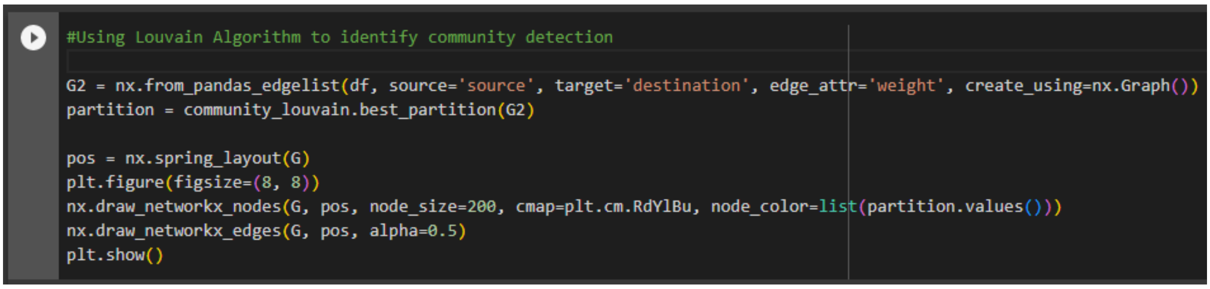
```
Community 1: ['Mall', 'Park', 'Canteen', 'School']
Community 2: ['Hospital', 'Metro', 'Theatre']
Community 3: ['Home', 'Docks']
Community 4: ['Post', 'Office']
```



From the following dataset we have identified that there are 4 set of communities that are present and are depicted graphically.

We have used **Louvain Algorithm** to identify community within the dataset.

The Louvain community detection algorithm is chosen due to its beautiful simplicity and the resulting ease of implementation. It allows circumventing the NP-complete problem of maximum cuts in graphs.



```
#Using Louvain Algorithm to identify community detection

G2 = nx.from_pandas_edgelist(df, source='source', target='destination', edge_attr='weight', create_using=nx.Graph())
partition = community_louvain.best_partition(G2)

pos = nx.spring_layout(G)
plt.figure(figsize=(8, 8))
nx.draw_networkx_nodes(G, pos, node_size=200, cmap=plt.cm.RdYlBu, node_color=list(partition.values()))
nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
```