Social Information Networks

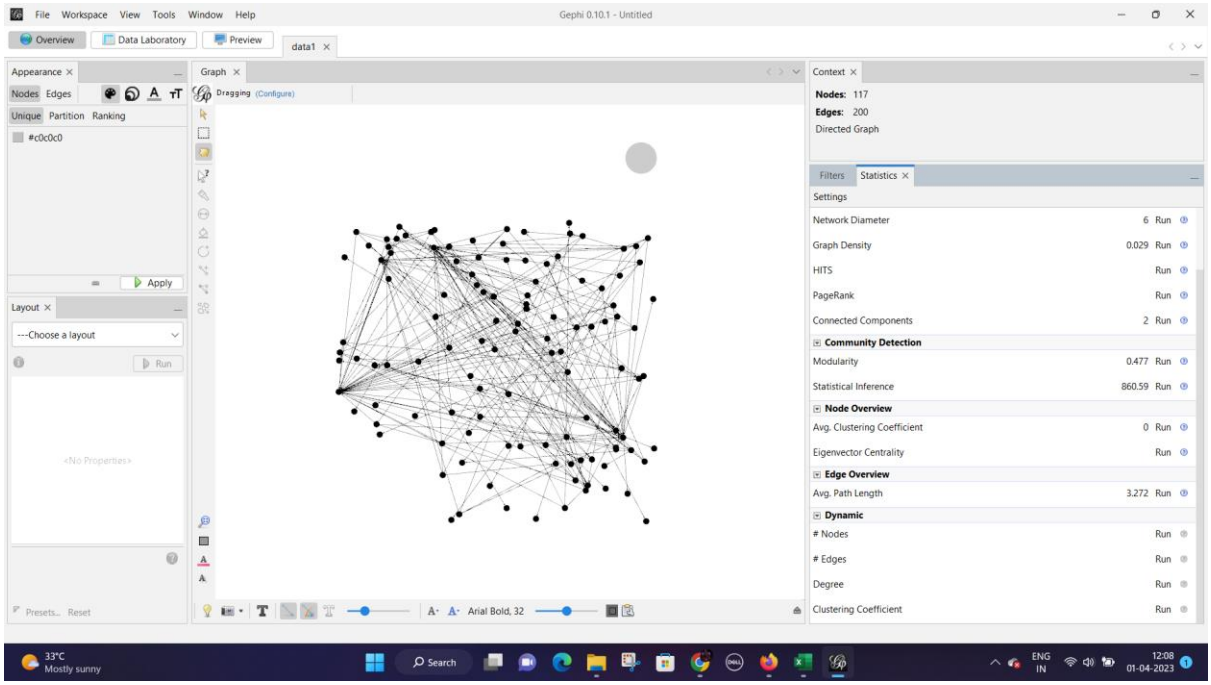
Digital Assignment-1

Name-Devansh Bajpai

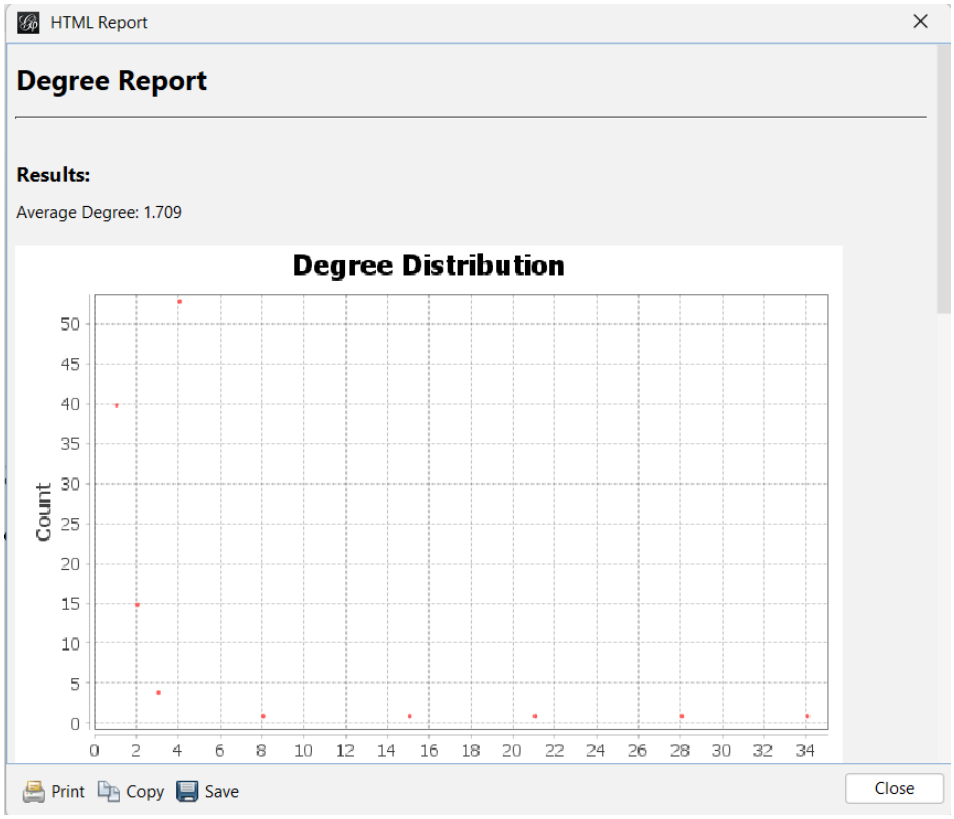
Registration No.-:20BCE0807

Slot-: B2

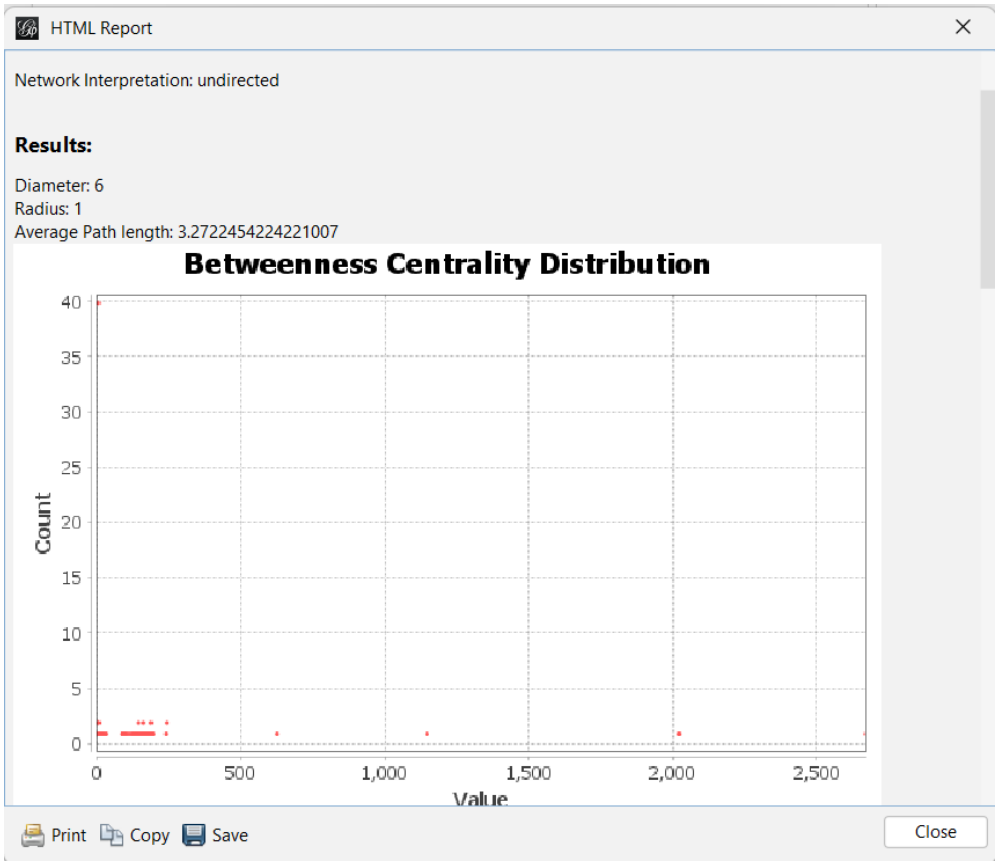
GEPHI FOR SOCIAL NETWORK VISUALIZATION

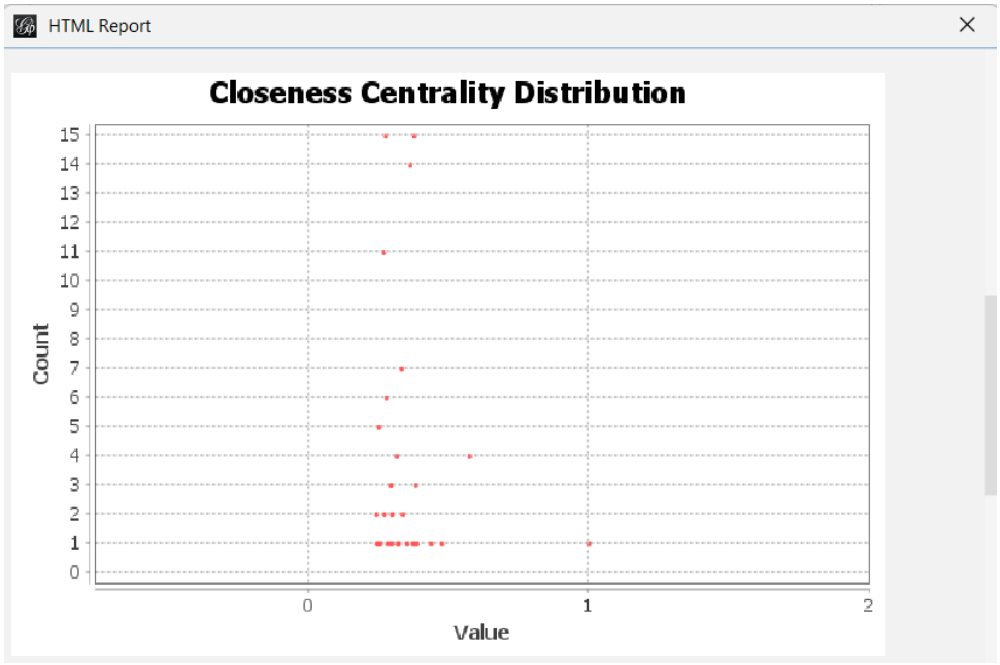


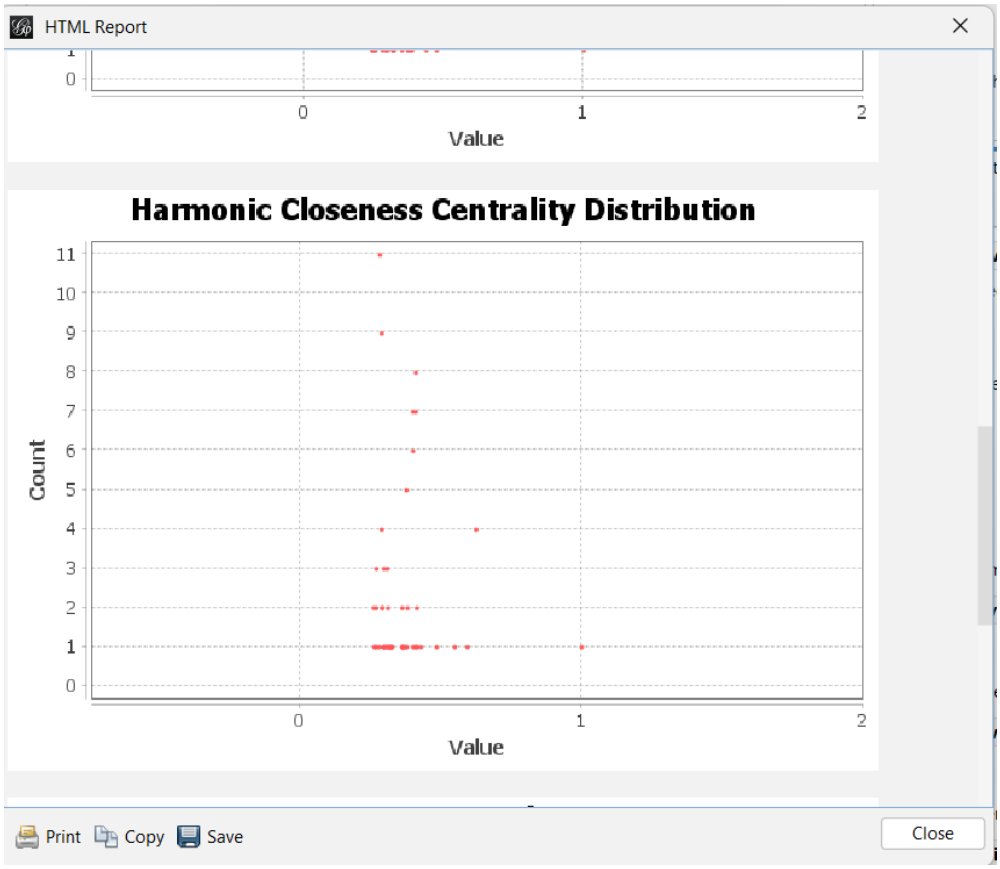
1.Average Degree

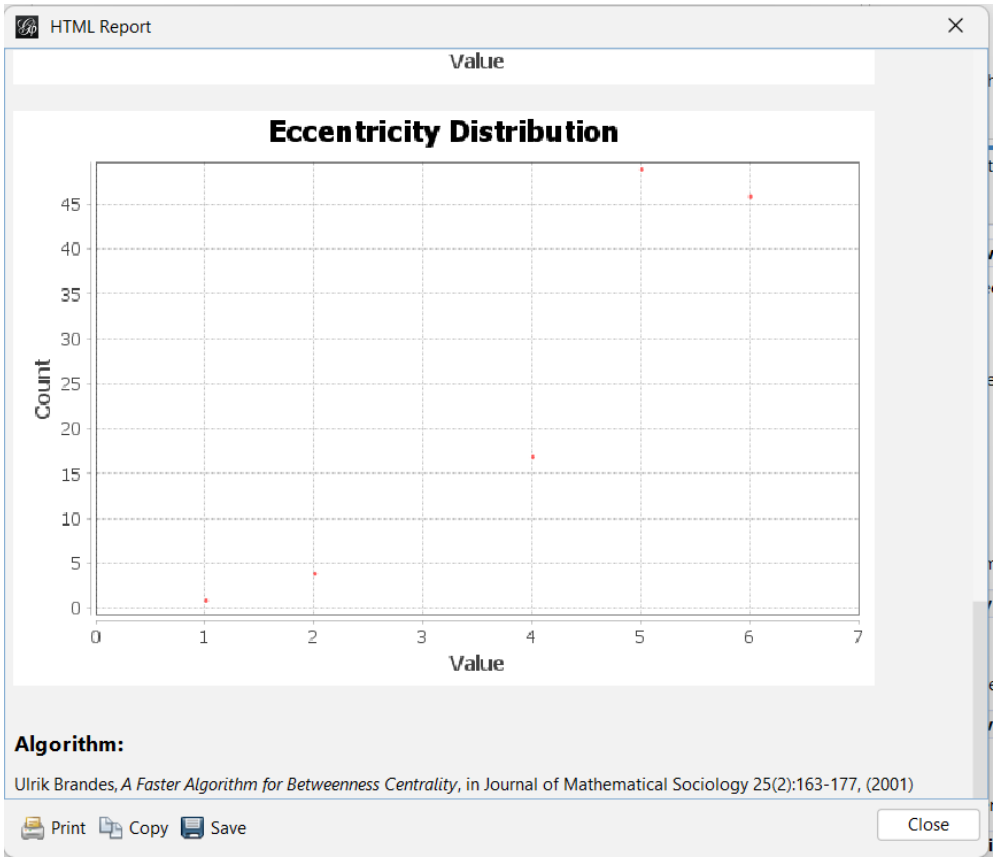


2. Network Diameter

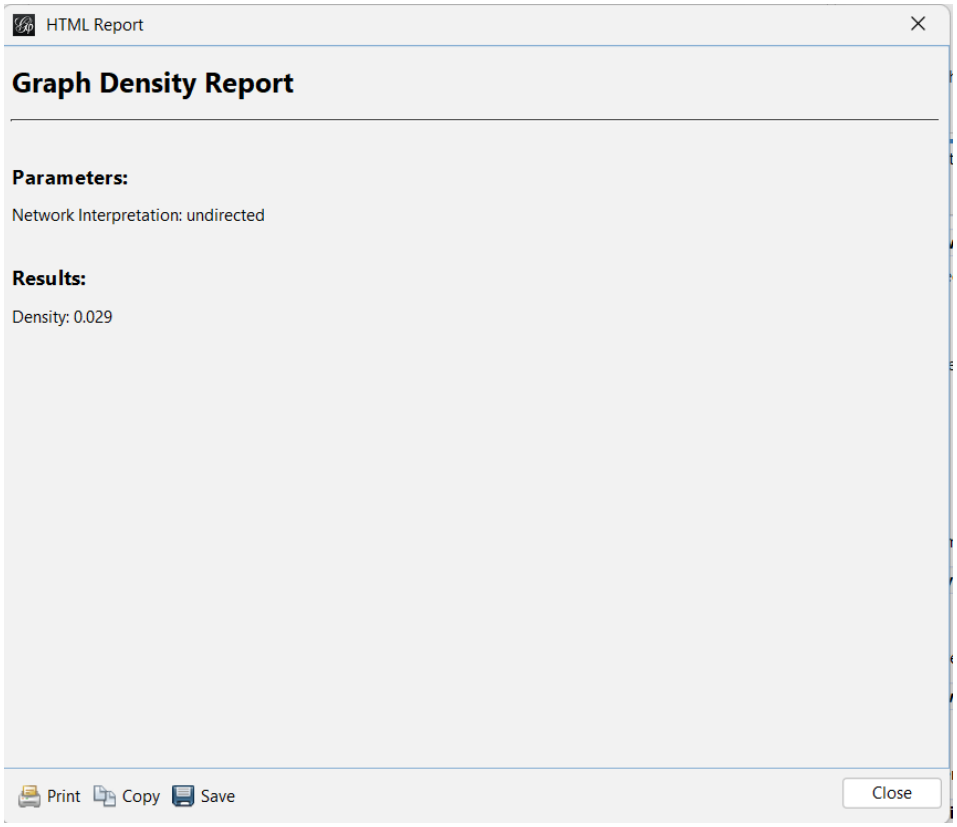




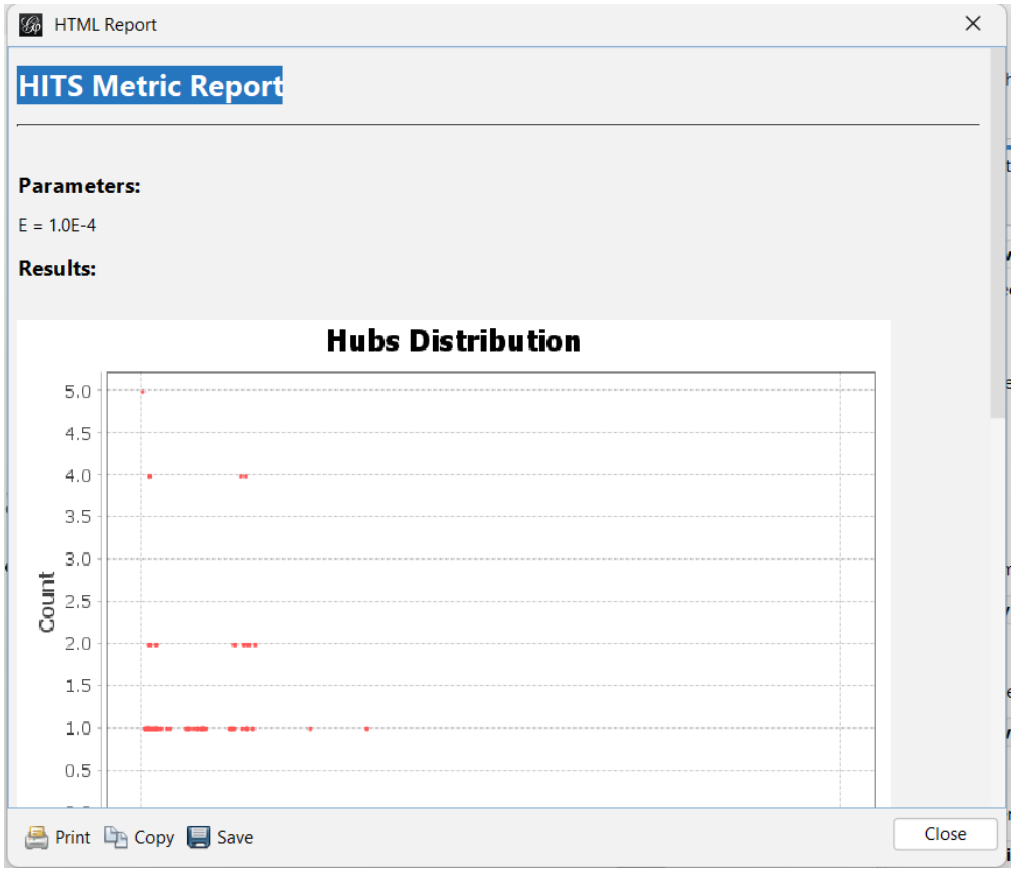

HTML Report
0
0
1
2
Value
Harmonic Closeness Centrality Distribution
Count
11
10
9
8
7
6
5
4
3
2
1
0
0
1
2
Value
Print
Copy
Save
Close


HTML Report
Value
Eccentricity Distribution
Count
45
40
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7
Value
Algorithm:
Ulrik Brandes, A Faster Algorithm for Betweenness Centrality, in Journal of Mathematical Sociology 25(2):163-177, (2001)
Print
Copy
Save
Close

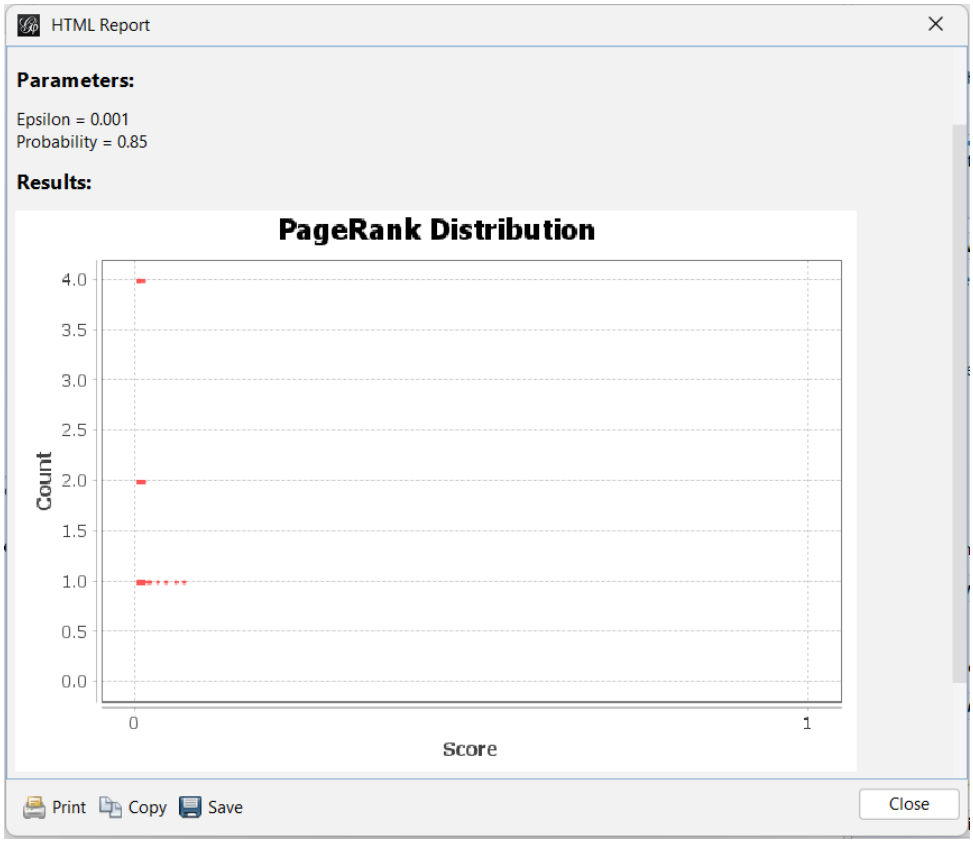
3. Graph Density



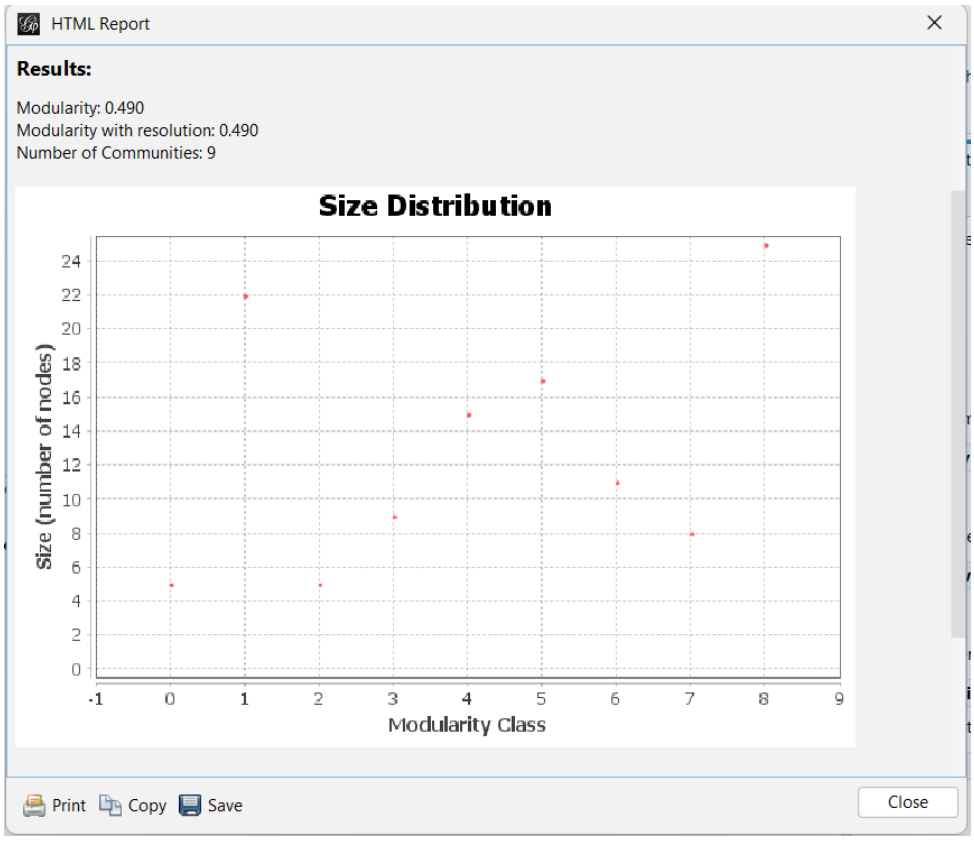
4. HITS Metric Report



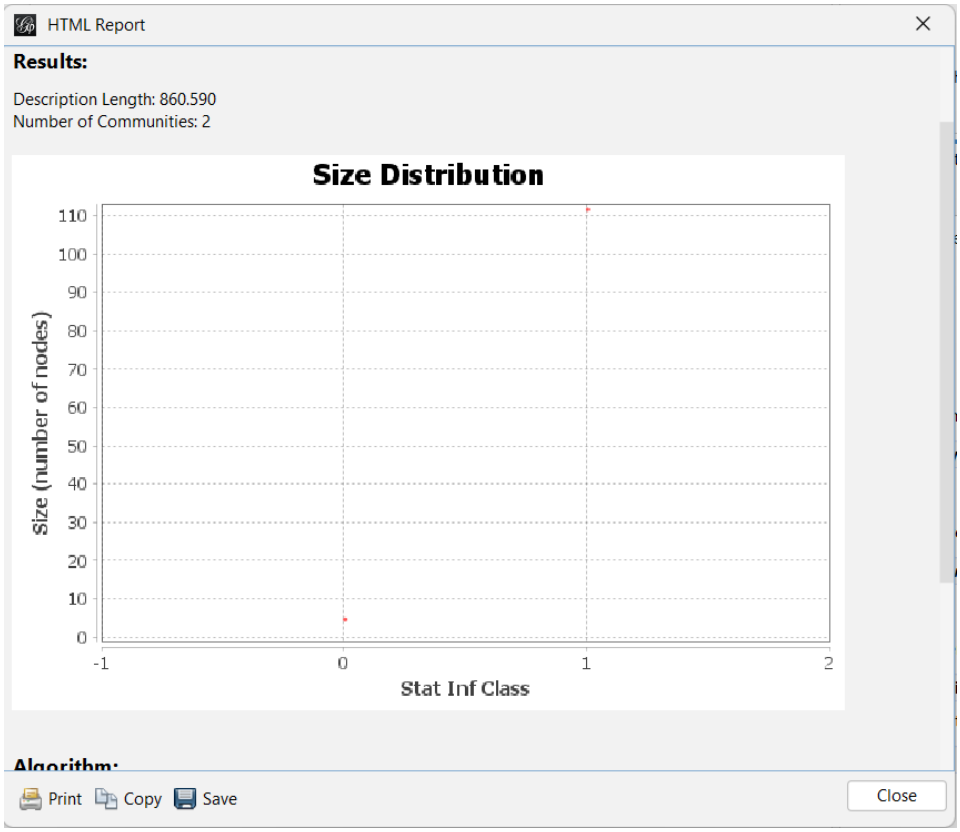
5. Page Rank Distribution



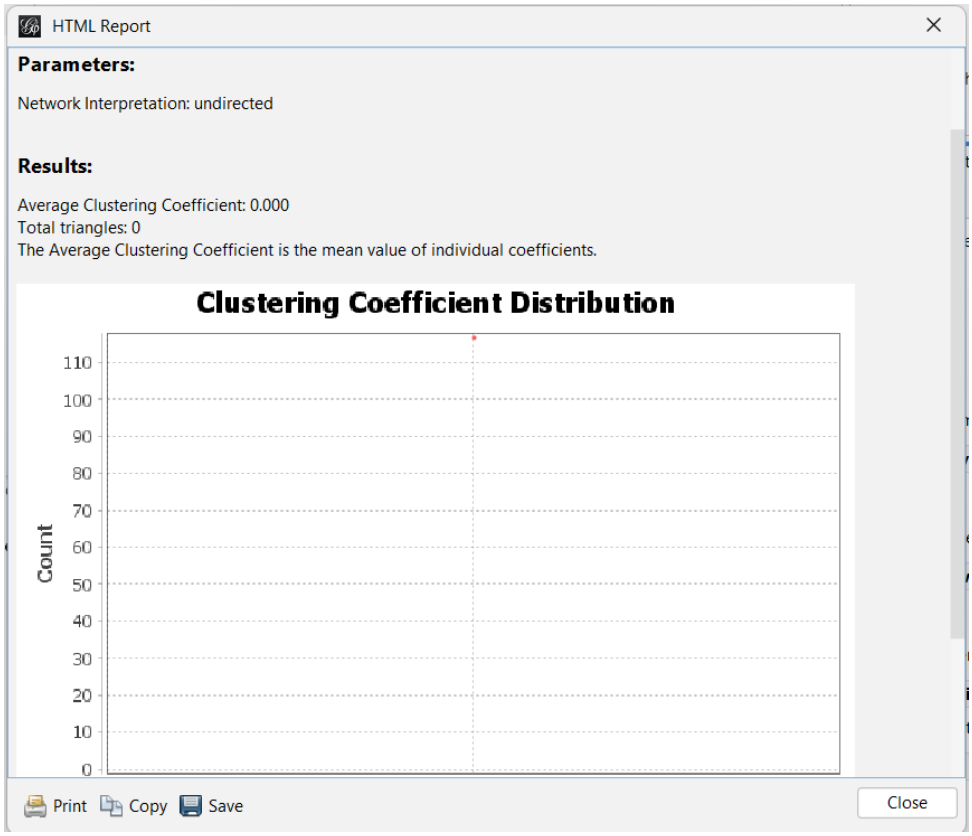
6. Modularity



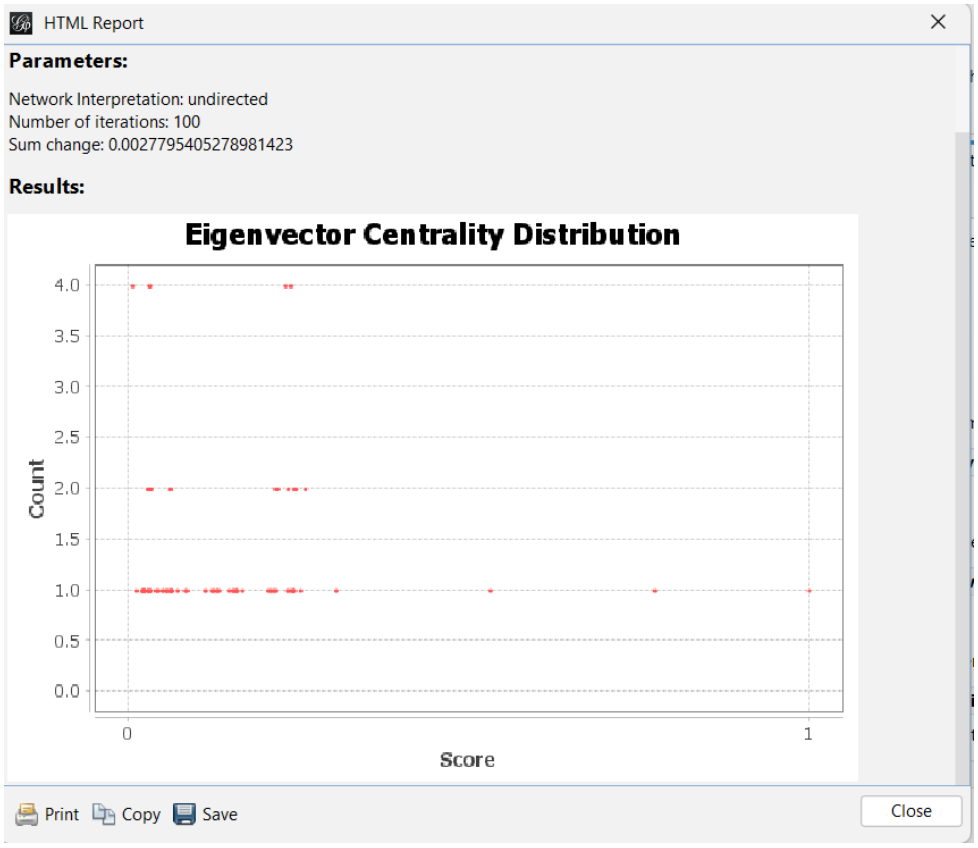
7.Statistical Inference Report



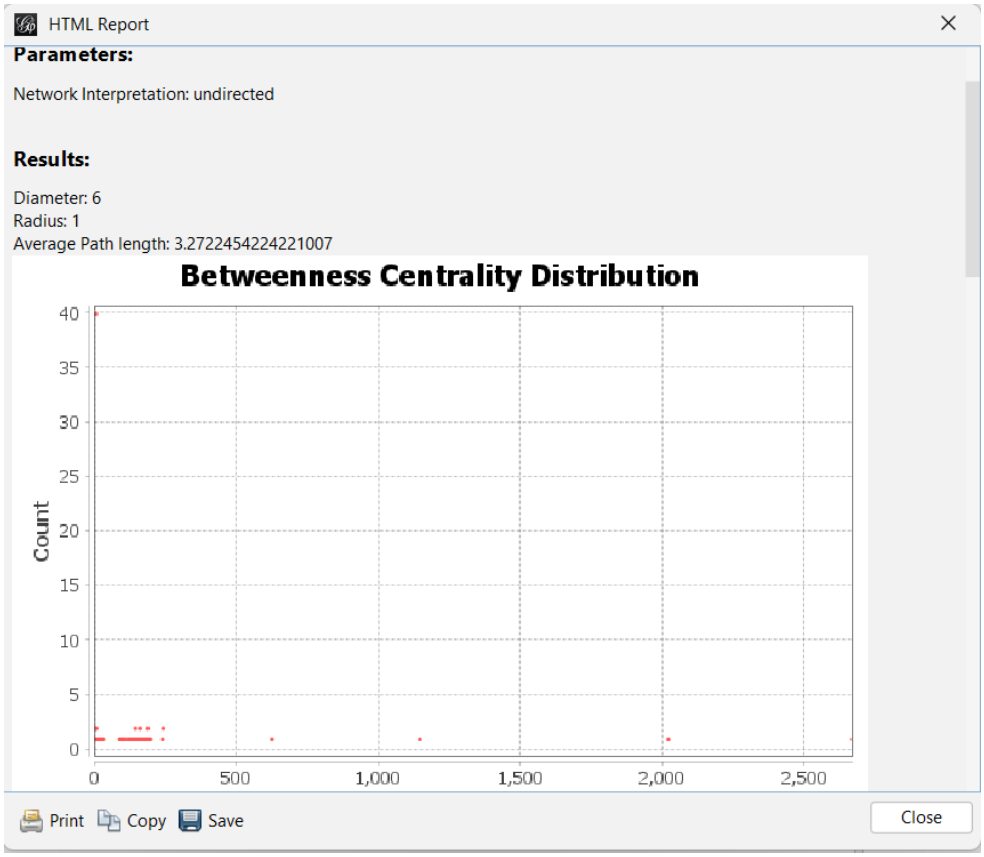
8.Average Clustering Coefficient



9.Eigen Vector Centrality



10.Average Path Length



Social Network Visualization using igraph library in Python

Code-:

```
import igraph as ig


# Create a graph with 6 nodes and 7 edges
g = ig.Graph()
g.add_vertices(21)
g.add_edges([(10, 8),(8, 1),(1, 3),(3, 7),(7 ,5),(5, 6),(6, 9),(9 ,10),(10, 4),(4, 9),(9, 0),(0, 6),(0, 4),(4, 2),(0, 2),(2, 1),(2 ,3),(4, 8),(0 ,5),(2, 7)])


# Calculate various network measures
degree = g.degree()  # nodal degree
betweenness = g.betweenness()  # betweenness centrality
local_clustering = g.transitivity_local_undirected()  # local clustering coefficients
global_clustering = g.transitivity_undirected()  # global clustering coefficient
density = g.density()  # density
membership = [0, 1, 1, 0, 0, 1, 1, 0, 1, 0,0, 1, 1, 0, 0, 1, 1, 0, 1, 0,1]
modularity = g.modularity(membership)# modularity
eigenvector_centrality = g.evcent()  # eigenvector centrality


# Print the results
print("Degree:", degree)
print("Betweenness Centrality:", betweenness)
print("Local Clustering Coefficients:", local_clustering)
print("Global Clustering Coefficient:", global_clustering)
print("Density:", density)
```

```
print("Modularity:", modularity)

print("Eigenvector Centrality:", eigenvector_centrality)
```

Output-:

```
Degree: [5, 3, 5, 3, 5, 3, 3, 3, 3, 4, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Betweenness Centrality: [9.083333333333334, 2.6666666666666665, 11.833333333333334, 1.1666666666666665, 9.083333333333332, 2.5,
1.0833333333333333, 2.6666666666666665, 2.5, 3.333333333333333, 1.0833333333333333, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0]
Local Clustering Coefficients: [0.4, 0.3333333333333333, 0.3, 0.6666666666666666, 0.4, 0.3333333333333333, 0.6666666666666666,
0.3333333333333333, 0.3333333333333333, 0.5, 0.6666666666666666, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]
Global Clustering Coefficient: 0.42105263157894735
Density: 0.09523809523809523
Modularity: -0.06125000000000003
Eigenvector Centrality: [0.9999999999999998, 0.49624546464118297, 0.8963822622038758, 0.48665718421982507, 1.0, 0.5430489479355
655, 0.611502314865473, 0.4962454646411833, 0.5430489479355655, 0.8303883844093118, 0.6115023148654734, 3.966613326807782e-17,
3.966613326807782e-17, 3.966613326807782e-17, 3.966613326807782e-17, 3.966613326807782e-17, 3.966613326807782e-17, 3.966613326807782e-17, 3.9666133268
07782e-17, 3.966613326807782e-17, 3.966613326807782e-17, 3.966613326807782e-17]
```

In [ ]: