IR-Assignment-3 Report

Riya Garg-MT22058

Abhinav Garg -MT22002

Saksham Nautiyal-MT22061

DATASET: We have used thesoc-sign-bitcoin-otc.csv dataset.This dataset has information about who-trusts-whom network of people who trade using Bitcoin on a platform called Bitcoin OTC.

The dataset has the following properties:

● Weighted

● Signed

● Directed

● Temporal

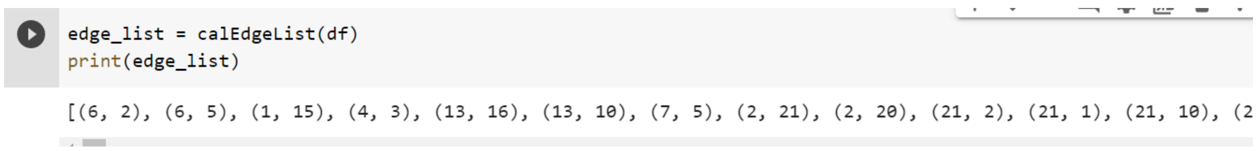
The number of nodes is 5,881 and the number of edges is 35,592.

LIbraries used in assignment:

- Pandas
- Numpy
- Networkx
- Matplotlib
- Math
- operator

Question1.

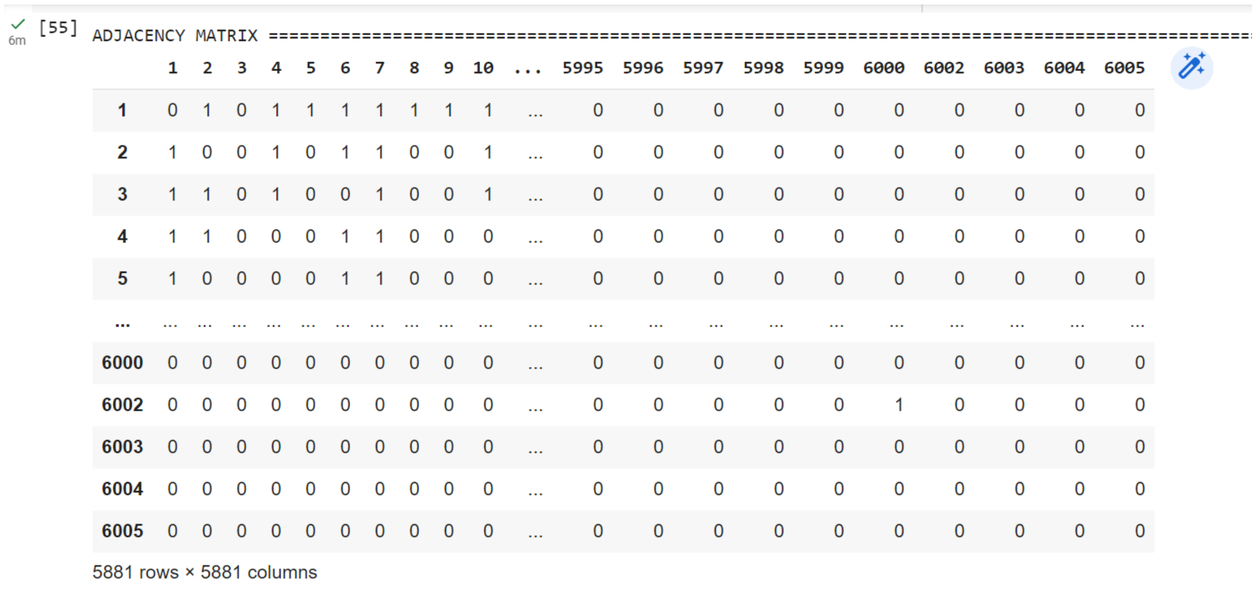
Edge representation of data:



```
edge_list = calEdgeList(df)
print(edge_list)
```

```
[(6, 2), (6, 5), (1, 15), (4, 3), (13, 16), (13, 10), (7, 5), (2, 21), (2, 20), (21, 2), (21, 1), (21, 10), (2
```

Adjacency matrix representation of data:

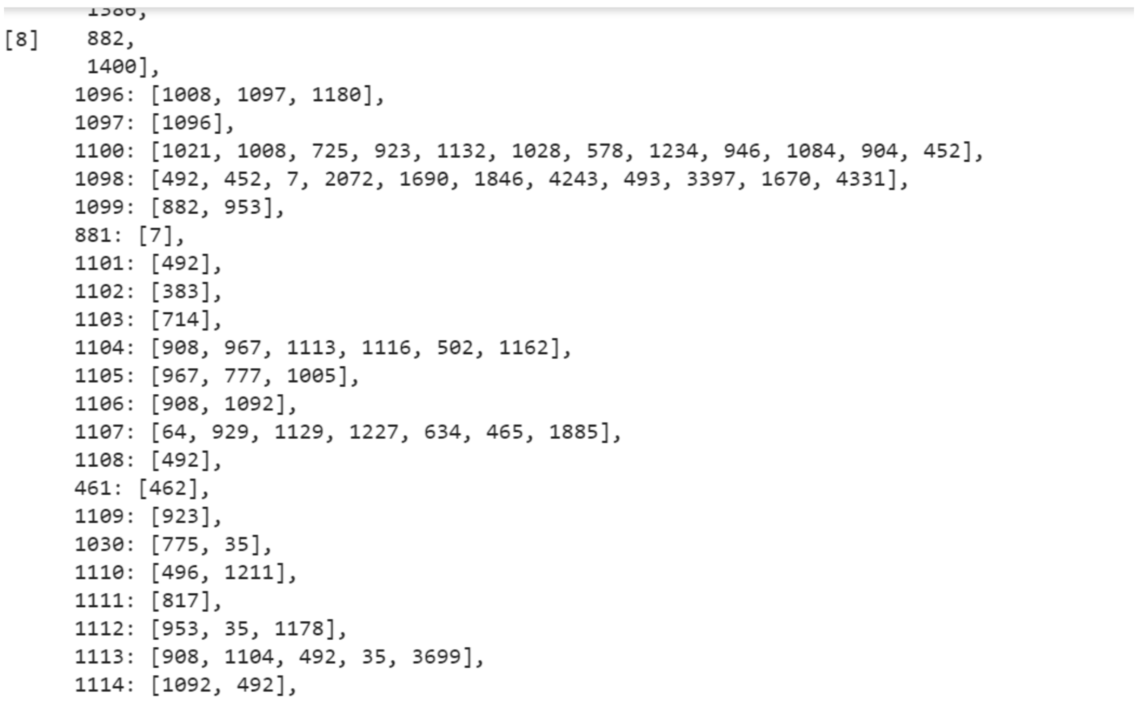


```
[55] ADJACENCY MATRIX ==========================================================================================
```

	1	2	3	4	5	6	7	8	9	10	...	5995	5996	5997	5998	5999	6000	6002	6003	6004	6005
1	0	1	0	1	1	1	1	1	1	1	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	1	1	0	0	1	...	0	0	0	0	0	0	0	0	0	0
3	1	1	0	1	0	0	1	0	0	1	...	0	0	0	0	0	0	0	0	0	0
4	1	1	0	0	0	1	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	1	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6000	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6002	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
6003	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6004	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6005	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

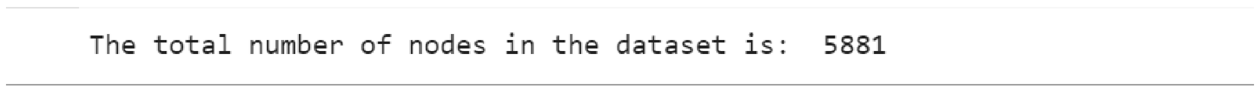
5881 rows × 5881 columns

Adjacency list representation:



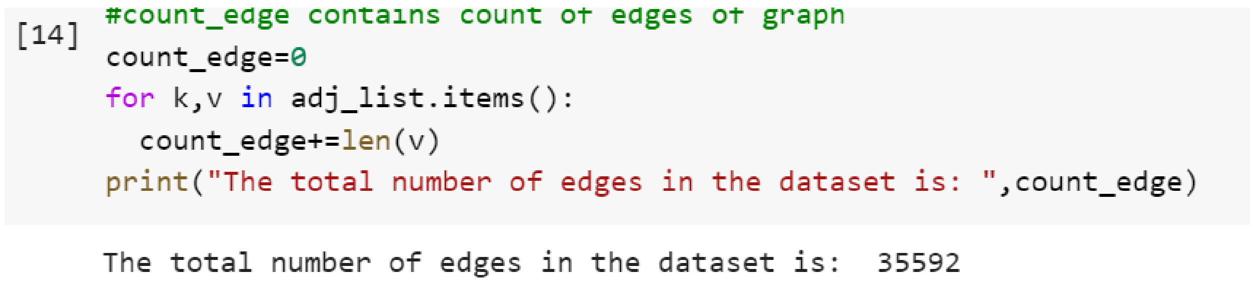
```
[8]   882,
      1400],
    1096: [1008, 1097, 1180],
    1097: [1096],
    1100: [1021, 1008, 725, 923, 1132, 1028, 578, 1234, 946, 1084, 904, 452],
    1098: [492, 452, 7, 2072, 1690, 1846, 4243, 493, 3397, 1670, 4331],
    1099: [882, 953],
    881: [7],
    1101: [492],
    1102: [383],
    1103: [714],
    1104: [908, 967, 1113, 1116, 502, 1162],
    1105: [967, 777, 1005],
    1106: [908, 1092],
    1107: [64, 929, 1129, 1227, 634, 465, 1885],
    1108: [492],
    461: [462],
    1109: [923],
    1030: [775, 35],
    1110: [496, 1211],
    1111: [817],
    1112: [953, 35, 1178],
    1113: [908, 1104, 492, 35, 3699],
    1114: [1092, 492],
```

1. Number of Nodes: In order to obtain the unique nodes, we iterated over each node using the adjacency list and saved it in a set.Thus no of unique nodes is equal to total no of nodes in the network.



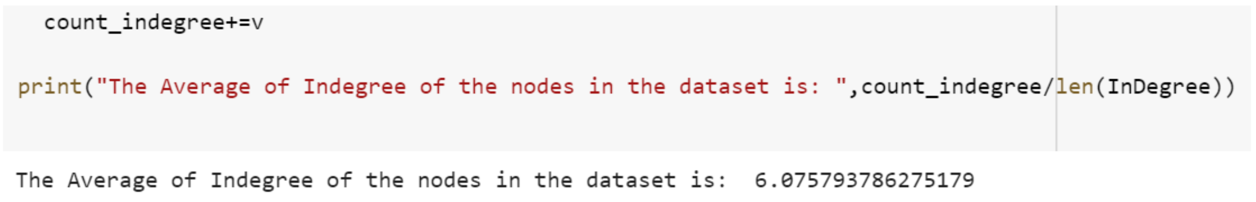
```
The total number of nodes in the dataset is:  5881
```

2. **Number of edges:**We used the adjacency matrix, iterated over it, and counted the number of 1s in the matrix to get the number of edges.



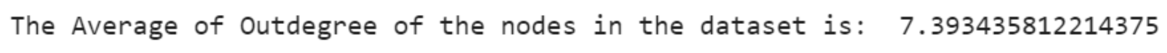
```
[14] #count_edge contains count of edges of graph
     count_edge=0
     for k,v in adj_list.items():
       count_edge+=len(v)
     print("The total number of edges in the dataset is: ",count_edge)

     The total number of edges in the dataset is:  35592
```

3. **Avg In-degree:** To calculate this we calculated indegree of each node and sum them then we divided that sum by number of nodes.



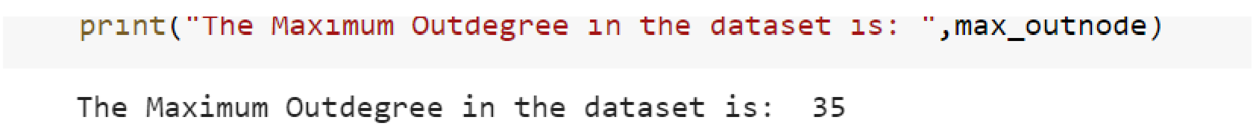
```
  count_indegree+=v

print("The Average of Indegree of the nodes in the dataset is: ",count_indegree/len(InDegree))

The Average of Indegree of the nodes in the dataset is:  6.075793786275179
```

4. Avg. Out-Degree: First, we determined the Out-Degree for each node. Then, we added up all the Out-Degrees and divided the result by the total number of nodes.



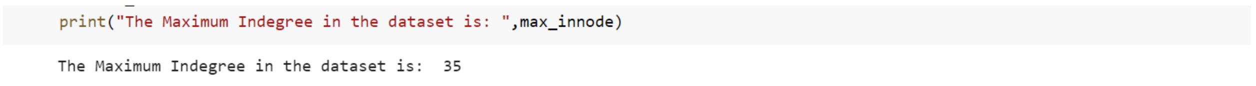
```
The Average of Outdegree of the nodes in the dataset is:  7.393435812214375
```

5.Node with max outdegree- We created a dictionary containing each node's out-degree, and from there we iterated to find the node with the maximum out-degree.



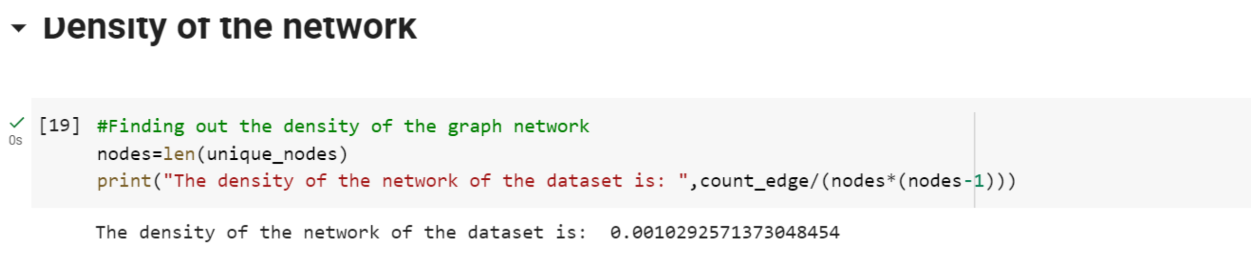
```
print("The Maximum Outdegree in the dataset is: ",max_outnode)

The Maximum Outdegree in the dataset is:  35
```

6. Node with max in degree-We created a dictionary containing each node's in-degree, and from there we iterated to find the node with the maximum in-degree.



```
print("The Maximum Indegree in the dataset is: ",max_innode)

The Maximum Indegree in the dataset is:  35
```

7. Density of network: The network's density is calculated by dividing the number of nodes by the total number of possible nodes.

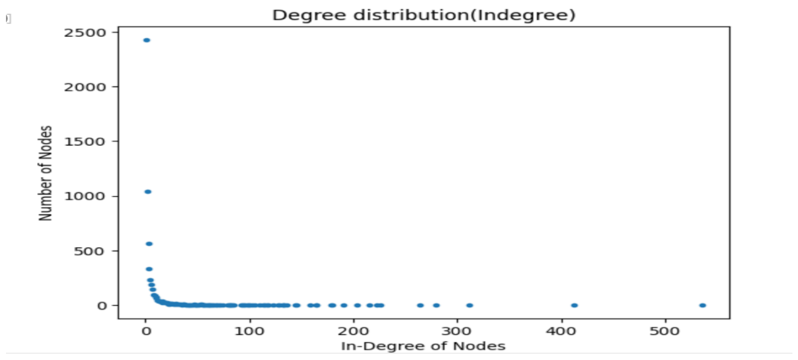


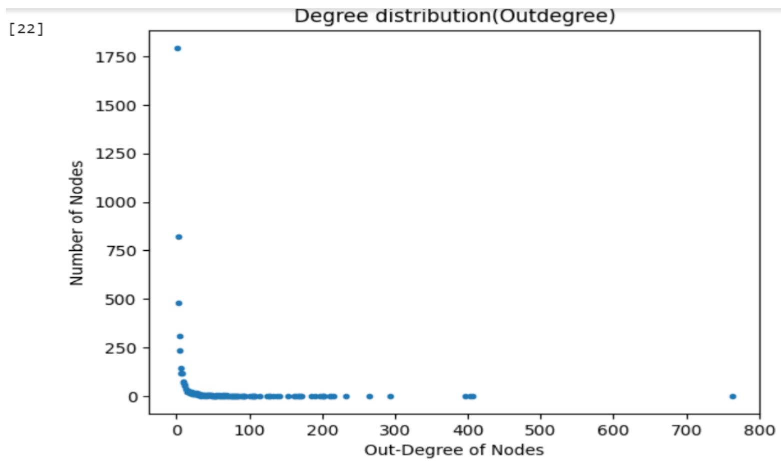
Density of the network

```
[19] #Finding out the density of the graph network
     nodes=len(unique_nodes)
     print("The density of the network of the dataset is: ",count_edge/(nodes*(nodes-1)))

     The density of the network of the dataset is:  0.0010292571373048454
```

<u>1.1</u>

<u>Plot degree distribution of the network:</u>





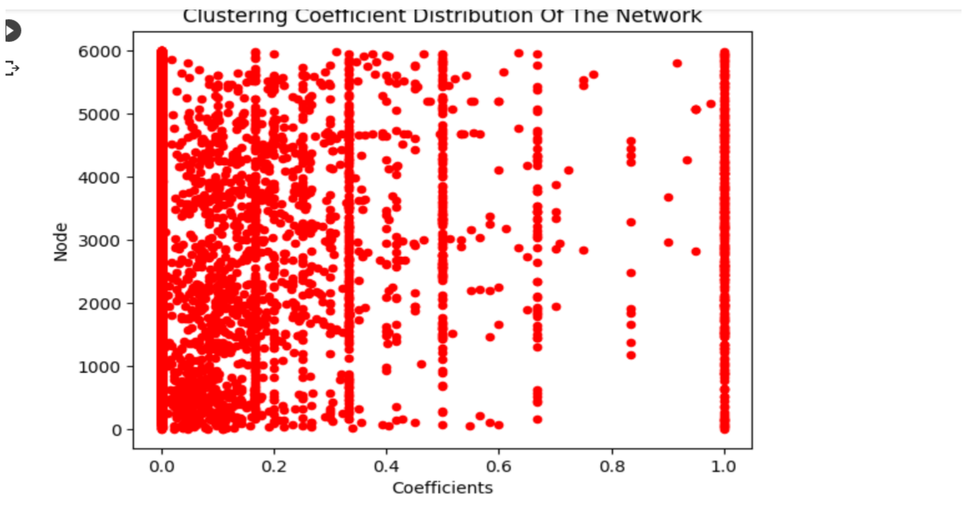
1c. Calculate the local clustering coefficient of each node and plot the clustering-coefficient distribution (lcc vs frequency of lcc) of the network.

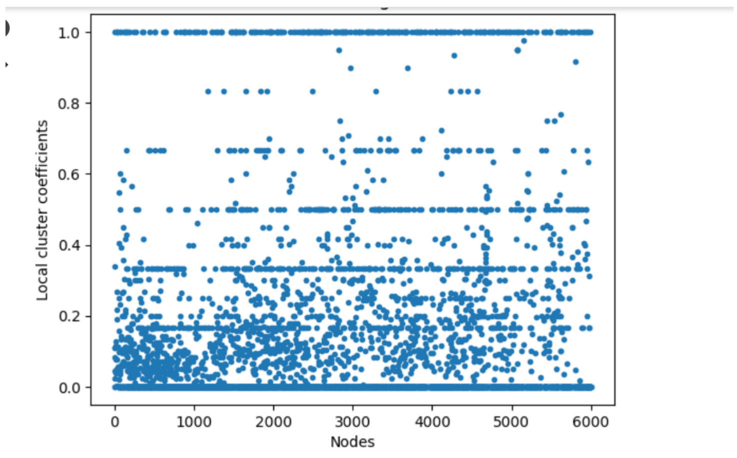
A cluster coefficient is a measure of the degree to which nodes in a graph tend to cluster together.

To calculate it, we have gone through each of the nodes, and after that, we found all the neighbors of that node.Then we found the number of actual pairs adjacent to each other. For this , for all these neighbors we have seen how many links are there between each of its neighbors.Then we divided the both values to get the local clustering coefficient.

- Lcc of each node is stored by us in a json file lcc.json

```
| 5934 |          0          |
| 5935 |          0          |
| 5936 |          0          |
| 5937 |          0          |
| 5938 |          0          |
| 5940 |          0          |
| 5941 |          0          |
| 5944 |          0          |
| 5945 |         0.0         |
| 5946 |          0          |
| 5947 |         0.0         |
| 5948 |         0.0         |
| 5952 |          0          |
| 5953 |          0          |
| 5954 |          0          |
| 5957 |          0          |
| 5959 |          0          |
| 5960 |          0          |
| 5961 |          0          |
| 5962 |          0          |
| 5963 |          0          |
| 5964 |          0          |
| 5966 |          0          |
```

- 

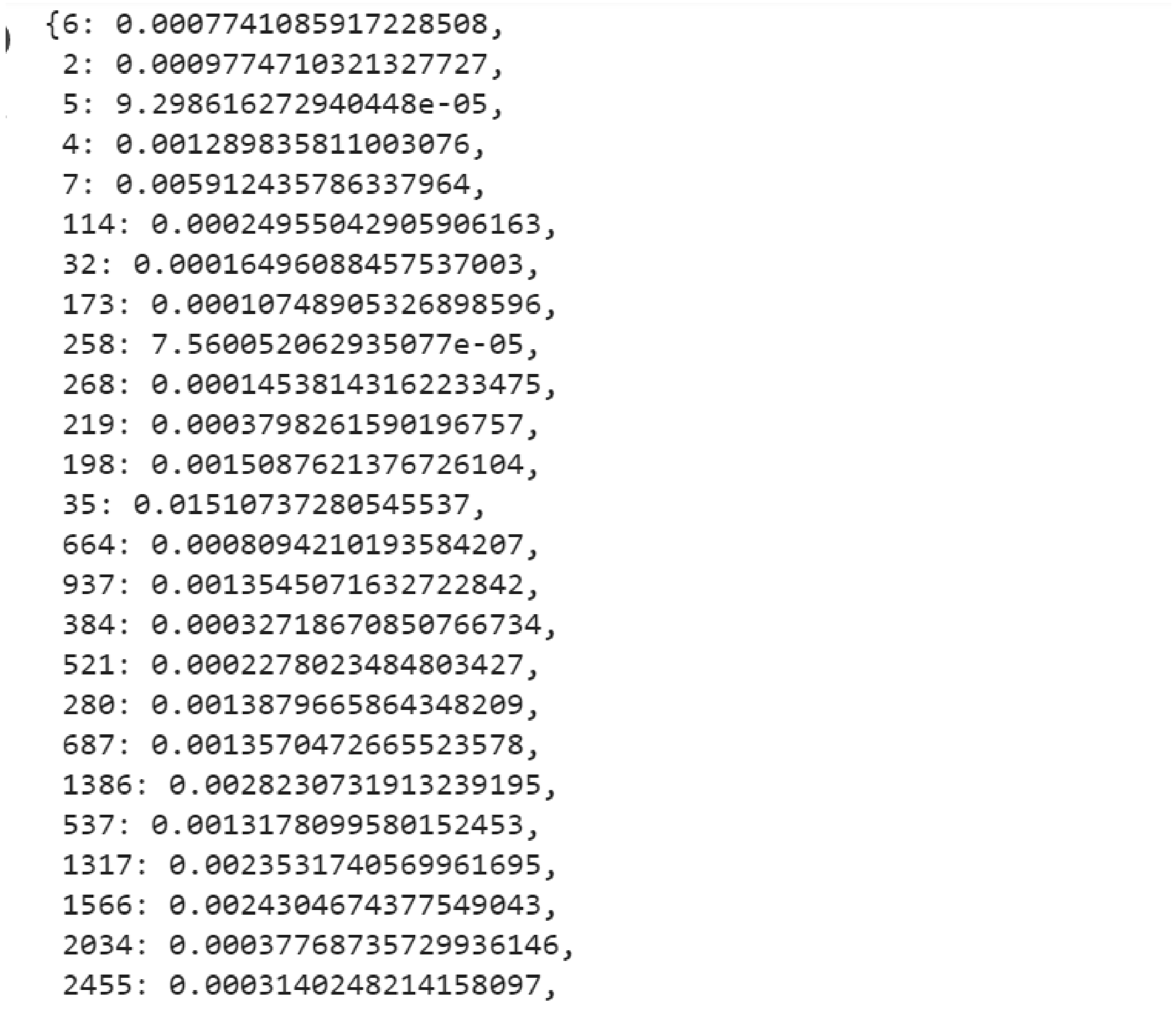


```
NODES WITH CLUSTERING COEFFICIENT 0:: 3756
NODES WITH CLUSTERING COEFFICIENT 1:: 262
CLUSTERING COEFFICIENT OF THE NETWORK:: 0.1192227938795901
```

QUESTION 2:

1. PageRank score for each node: Page rank ranks the pages in the order of relevance. More incoming edges implies higher page rank**.**

We have used the networkx library to obtain the page rank of the network.And we also calculated it from scratch.



```
{6: 0.0007741085917228508,
 2: 0.0009774710321327727,
 5: 9.298616272940448e-05,
 4: 0.001289835811003076,
 7: 0.005912435786337964,
 114: 0.0002495504290590616З,
 32: 0.0001649608845753700З,
 173: 0.0001074890532689859б,
 258: 7.560052062935077e-05,
 268: 0.0001453814316223347б,
 219: 0.0003798261590196757,
 198: 0.0015087621376726104,
 35: 0.01510737280545537,
 664: 0.0008094210193584207,
 937: 0.0013545071632722842,
 384: 0.0003271867085076673,
 521: 0.0002278023484803427,
 280: 0.0013879665864348209,
 687: 0.0013570472665523578,
 1386: 0.0028230731913239195,
 537: 0.0013178099580152453,
 1317: 0.0023531740569961695,
 1566: 0.0024304674377549043,
 2034: 0.00037768735729936146,
 2455: 0.00031402482141580 97,
```

2.Authority and Hub score for each node: We have implemented authority and hub scores for our network from the scratch.

1. Let the number of iterations be k.

2. Hub score = 1 and authority score =1 are assigned to each node.

3. Repeat k times.

4. Update the scores.

5. Normalize the scores by dividing hub scores by the square root of the sum of the squares of all hub scores and, similarly, authority scores

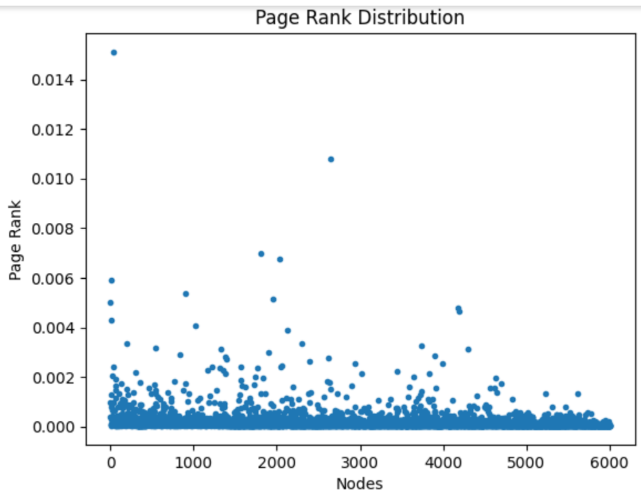
Authority scores:

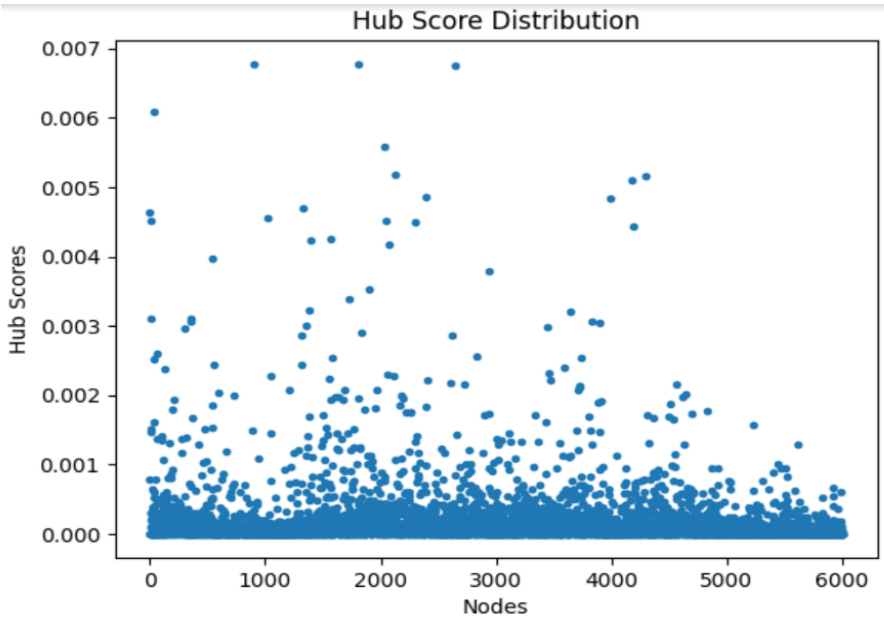
```
{0: 0.0, 1: 0.004508644601072364, 2: 0.0005920739279229431, 3: 0.0005492534773728823, 4: 0.0011249134077155882, 5: 0.00017073586924151596, 6: 0.001574
```

Hub scores:

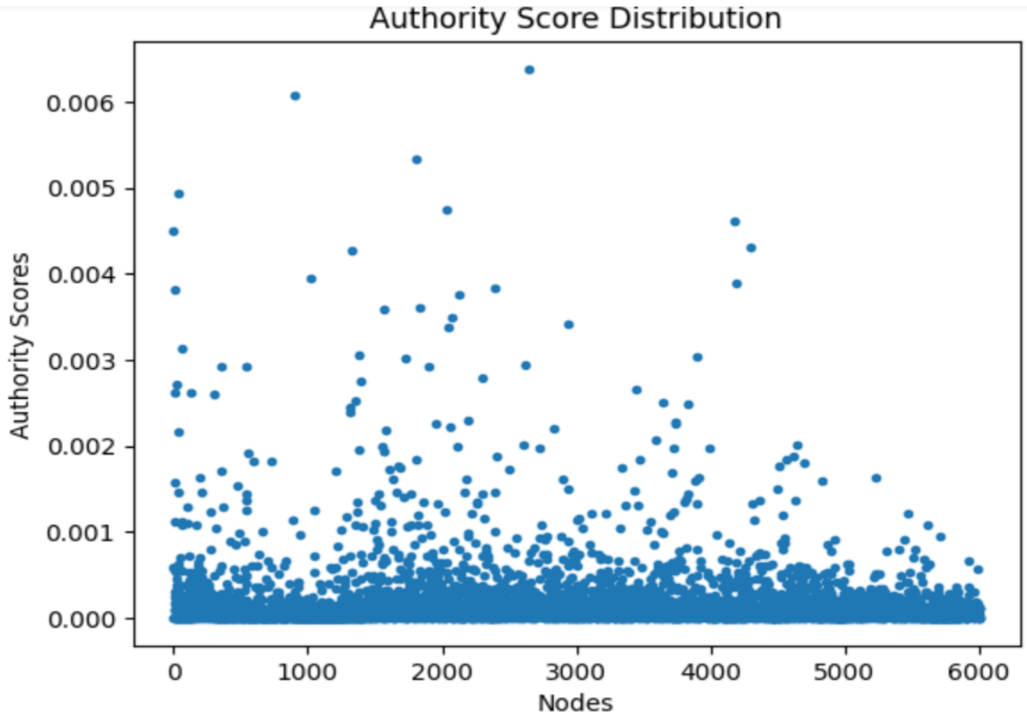
```
{0: 0.0, 1: 0.004639680682805912, 2: 0.0007769319266726913, 3: 0.0, 4: 0.001509279769466799, 5: 0.00020899535466224138, 6: 0.0014647799971546785, 7: 0.
```

3. Compare the results obtained from both the algorithms in parts 1 and 2 based on the node scores

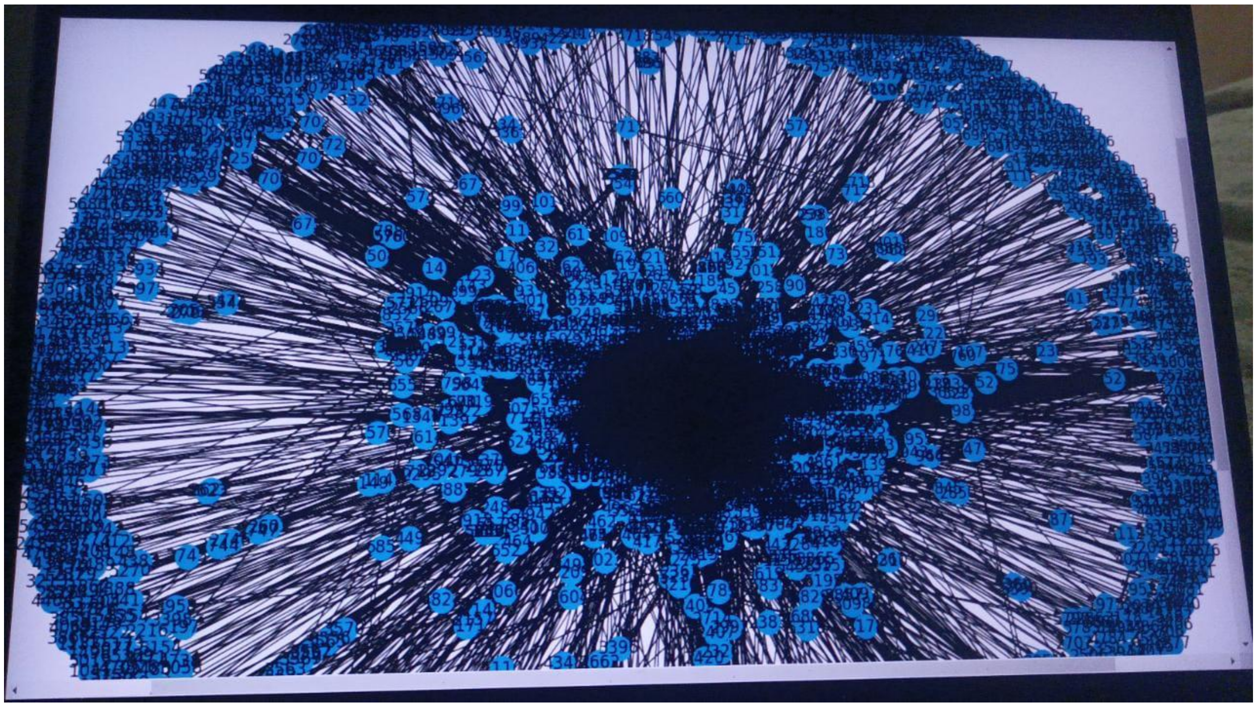




]



Visualization of the network:



Comparison: As we can see in the graphs of various distributions, including page rank, hub-score, and authority score, the outcomes are essentially the same for hub and authority because they both use small subsets of the graph to determine each node's score, while page rank uses the probability distribution of every node. Therefore, we can say that the outcomes of all three of these strategies in this dataset are comparable and often give similar results.