Graph Analysis and Algorithms:

Network analysis case study:
Analyzing the retweet network:

Course instructor:
Dr. Somasundaram K

Done by:
Smrithi. N (CB.SC.I5DAS19029)
Varun kumar.M (CB.SC.I5DAS19032)

Problem statement of the project:

- Twitter is a social media platform which is similar to the other social media platforms and contains millions of users (approx.). Twitter has a feature which allows the users to retweet a post that a person posts if they are able to relate to it or they like it.

- Our project focuses mainly on deriving insights from the network that is formed by taking the various users of twitter as the nodes and a person retweeting another person's post as a directed edge.

About our Dataset:

- Our dataset contains 761 nodes and 1030 edges. In other words, our dataset took 761 twitter users into account and found 1031 retweet instances between the considered users.

```
fromnode=[]
tonode=[]
txt=open("rt-twitter-copen.mtx","r")
for i in txt:
    x,y=i.split()
    fromnode.append(x)
    tonode.append(y)
```

```
df=pd.DataFrame()
df['fromnode']=fromnode
df['tonode']=tonode
```

```
a=pd.read_csv('twitter.csv',index_col=None)
a
```

	fromnode	tonode
0	761	761
1	245	1
2	512	1
3	209	2
4	255	2
...	...	...
1025	750	722
1026	735	728
1027	756	728
1028	749	737
1029	761	740

1030 rows × 2 columns

```
len(g.nodes)
```

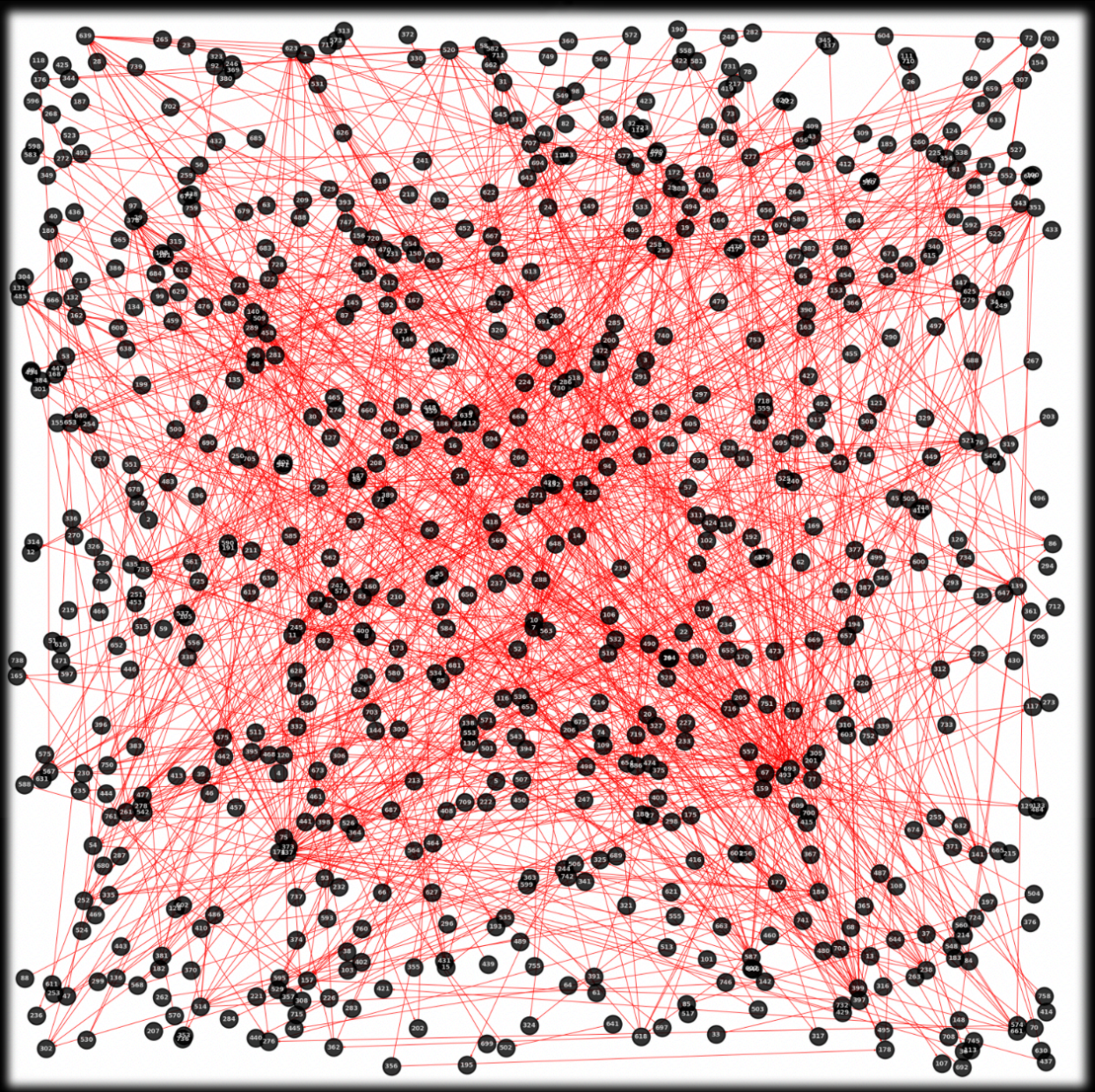
```
761
```

```
len(g.edges)
```

```
1030
```

Visualizing the dataset:

```
g=nx.from_pandas_edgelist(df, 'fromnode', 'tonode',create_using=nx.DiGraph())
plt.figure(figsize=(50, 50))
nx.draw_random(g,with_labels=True,node_size=2500, node_color="black", node_shape="o", alpha=0.8, linewidths=4,
               font_size=18, font_color="white", font_weight="bold", width=2, edge_color="red")
plt.show()
```



About Graph centralities:

- Graph centralities act as a very important measure to identify all the important nodes in a network.

- In this case, it is used to identify the influential person in the retweet network.

- The graph centralities shown in this project are:

    - degree centrality
    - Betweenness centrality
    - Closeness centrality
    - Eigen vector centrality
    - Clustering centrality

About degree centrality:

- Degree centrality of a node gives us the total number of edges that are incident on that particular node.
- Since the network that we have considered in this project is a

DIRECTED GRAPH, DEGREE CENTRALITY OF ONE NODE IN THIS NETWORK GIVES US THE TOTAL NUMBER OF INCOMING EDGES TO THAT PARTICULAR NODE.

```
v=pd.DataFrame()
deg_cent=nx.in_degree_centrality(g)
ver=list(deg_cent.keys())
deg=list(deg_cent.values())
v['vertices']=ver
v['degree_centrality']=deg
v.sort_values(by=['degree_centrality'],ascending=False)
```

	vertices	degree_centrality
105	137	0.038158
137	158	0.035526
29	10	0.031579
184	77	0.022368
264	229	0.018421
...	...	...
505	751	0.000000
504	705	0.000000
503	658	0.000000
502	578	0.000000
380	729	0.000000

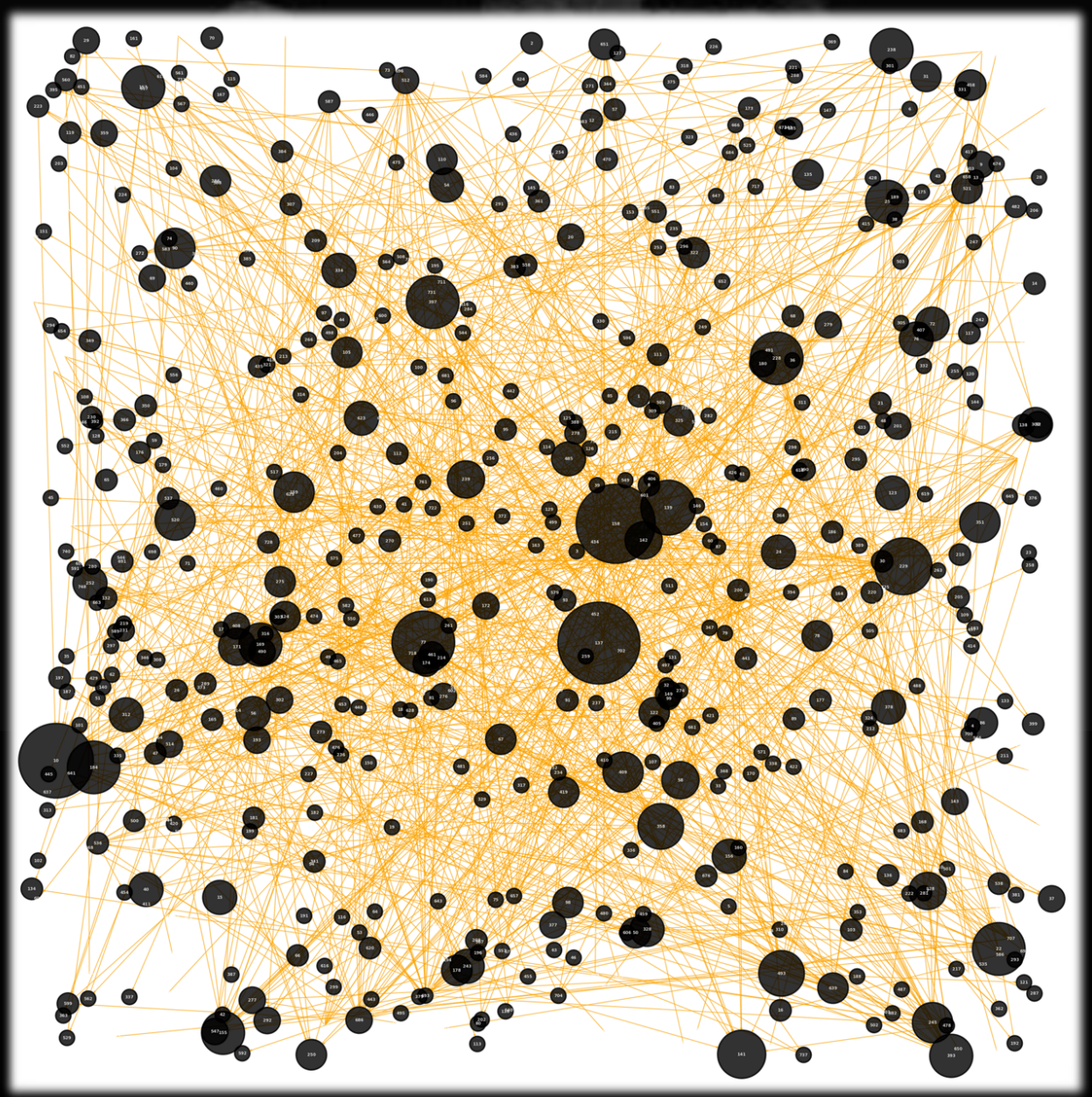
761 rows × 2 columns

NODE 137 HAS THE HIGHEST DEGREE CENTRALITY

Visualization of degree centrality:

```
size=[d * 1500000 for d in deg]
plt.figure(figsize=(50, 50))
nx.draw_random(g,with_labels=True,node_size=size, node_color="black", node_shape="o",
               alpha=0.8, linewidths=4,font_color='white',font_weight="bold", width=2, edge_color="orange")
plt.show()
```

The higher the degree the bigger the node



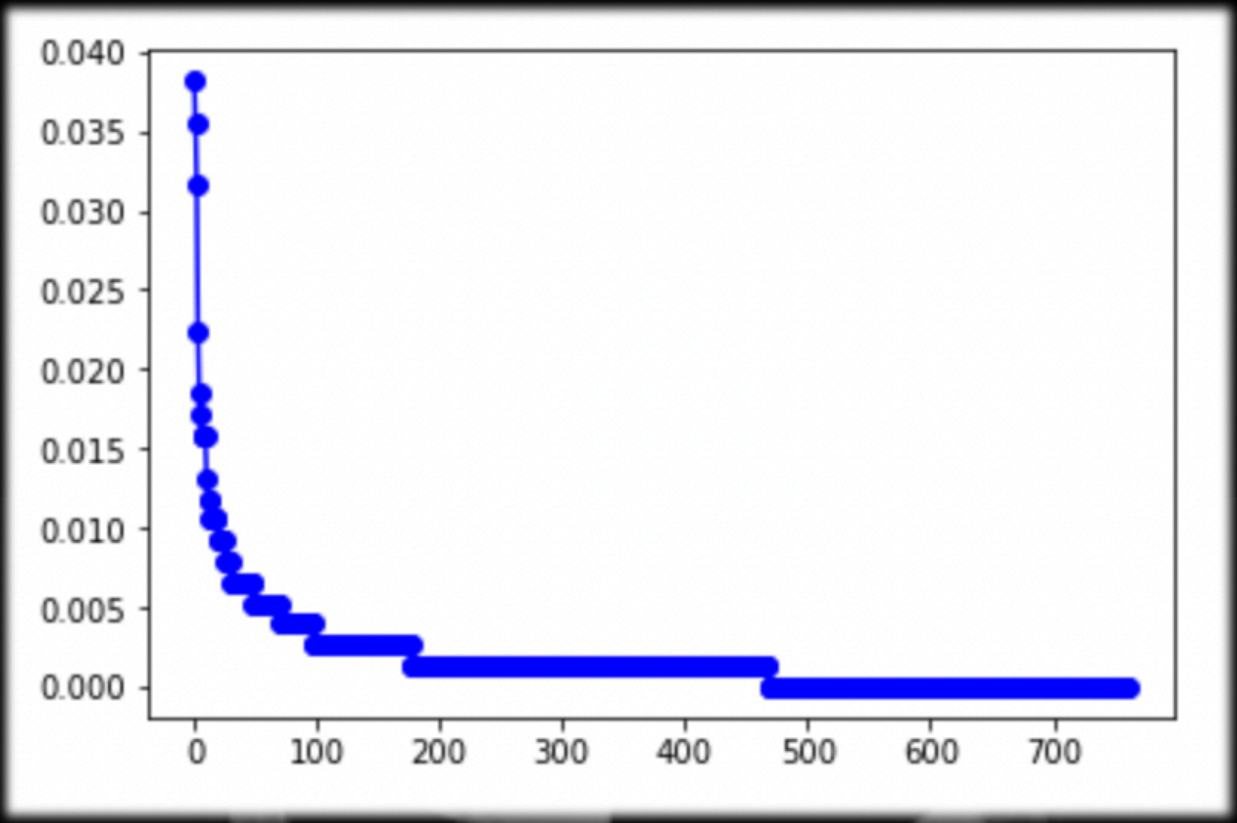
Inference on degree centrality:

- From the above visualization and degree centrality calculation it can be seen that node 137 has the highest degree centrality, therefore it can be inferred that the posts in this user's twitter account is liked by many other users. Similarly the same inference can be drawn for the nodes 158, 10, 77 and 229 as this set of nodes has a centrality that is almost equal to that of 105.

- **Degree vs rank plot:**

- The above plot assigns a rank to each node with respect to its degree centrality.

```
degs=sorted(list(deg_cent.values()),reverse=True)
graph=plt.plot(degs, "b-", marker="o")
```



About betweenness centrality:

- Betweenness centrality is used to measure the number of shortest paths that passes through a node.

- the formula for betweenness centrality is given by:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

- $\sigma_{st}(v)$ - shortest distance between 2 vertices that pass through the node v

- σ_{st} - Number of shortest paths between the two nodes

```
bet_cent=list(nx.betweenness_centrality(g).values())
v['betweenness_centrality']=bet_cent
v.sort_values(by=['betweenness_centrality'],ascending=False)
```

	vertices	degree_centrality	betweenness_centrality
137	158	0.035526	0.002009
105	137	0.038158	0.001450
34	228	0.015789	0.001243
291	393	0.010526	0.001108
70	397	0.015789	0.000961
...	...	...	...
319	135	0.005263	0.000000
320	527	0.000000	0.000000
321	723	0.000000	0.000000
322	136	0.002632	0.000000
760	740	0.001316	0.000000

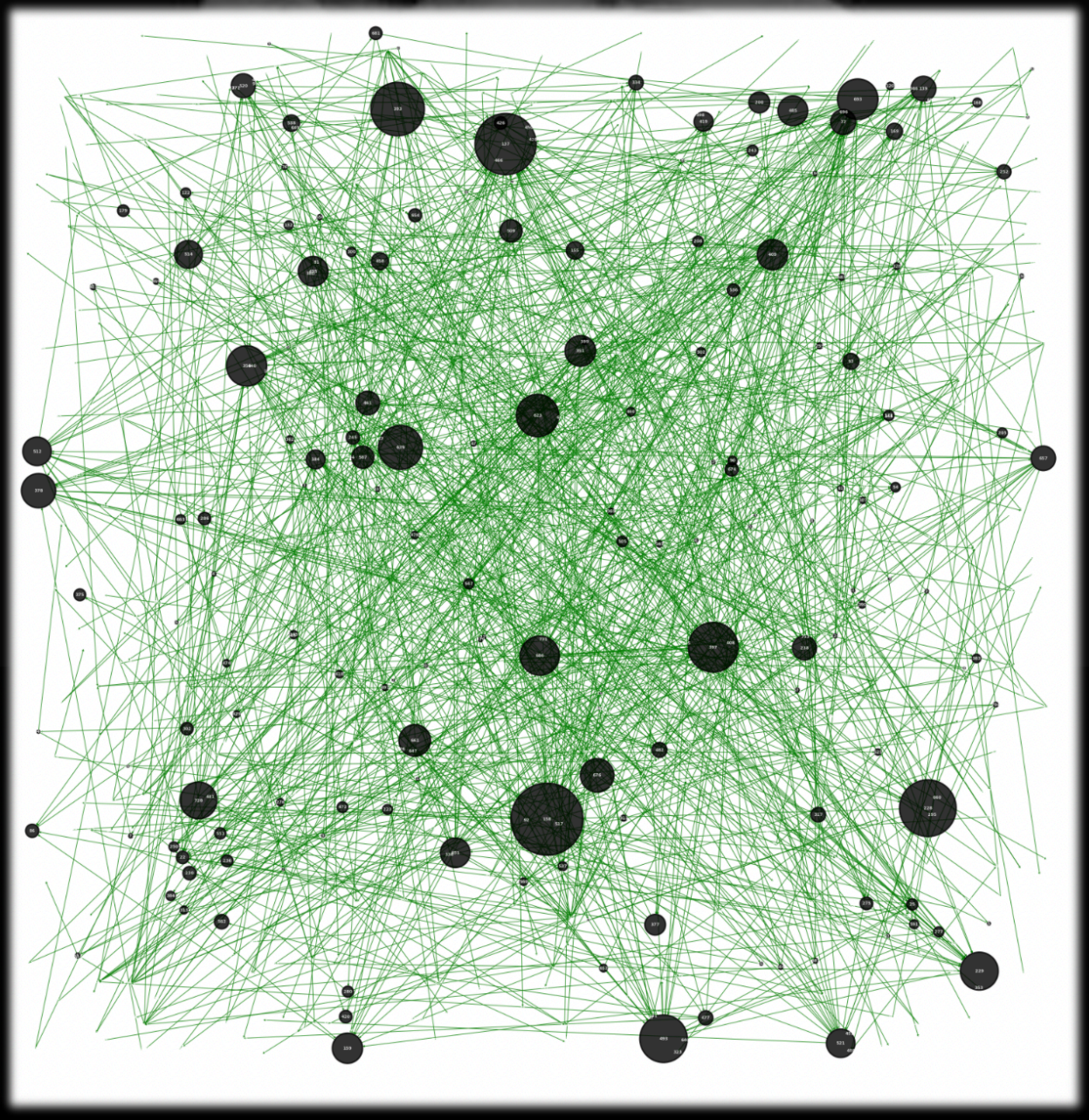
761 rows × 3 columns

Node 158 has the highest betweenness centrality.

Visualizing Betweenness Centrality:

```
size1=[b * 22000000 for b in bet_cent]
plt.figure(figsize=(50, 50))
nx.draw_random(g,with_labels=True,node_size=size1, node_color="black", node_shape="o", alpha=0.8, linewidths=4,
               font_color='white',font_weight="bold", width=2, edge_color="green")
plt.show()
```

The greater the betweenness centrality the bigger the node



Inference of betweenness centrality:

- Betweenness centrality will give us information like which node occurs more when we travel from one node to another node. From the above graph it can be seen that most of the shortest distance paths pass through node 158, which tells us that this node plays a really important role in communication. The nodes 137, 228, 393 and 397 also play an important role in communication as the betweenness centrality of these nodes are also on par with 158.

About closeness centrality:

- Closeness centrality is a measure that is used to measure how fast the information can be passed from that node to all other nodes.

- In other words, closeness centrality measures how close one node is to all the other nodes in a network and is measured by taking the average of the shortest path length from one node to all other nodes.

- The closeness centrality is given by:

$$CC(i) = \frac{N-1}{\sum_j d(i,j)}$$

```
close_cent=list(nx.closeness_centrality(g).values())
v['closeness_centrality']=close_cent
v.sort_values(by=['closeness_centrality'],ascending=False)
```

	vertices	degree_centrality	betweenness_centrality	closeness_centrality
184	77	0.022368	0.000244	0.069740
137	158	0.035526	0.002009	0.069524
105	137	0.038158	0.001450	0.067788
209	124	0.005263	0.000006	0.062633
282	139	0.017105	0.000239	0.060824
...	...	...	...	...
395	566	0.000000	0.000000	0.000000
393	370	0.000000	0.000000	0.000000
392	367	0.000000	0.000000	0.000000
391	340	0.000000	0.000000	0.000000
380	729	0.000000	0.000000	0.000000

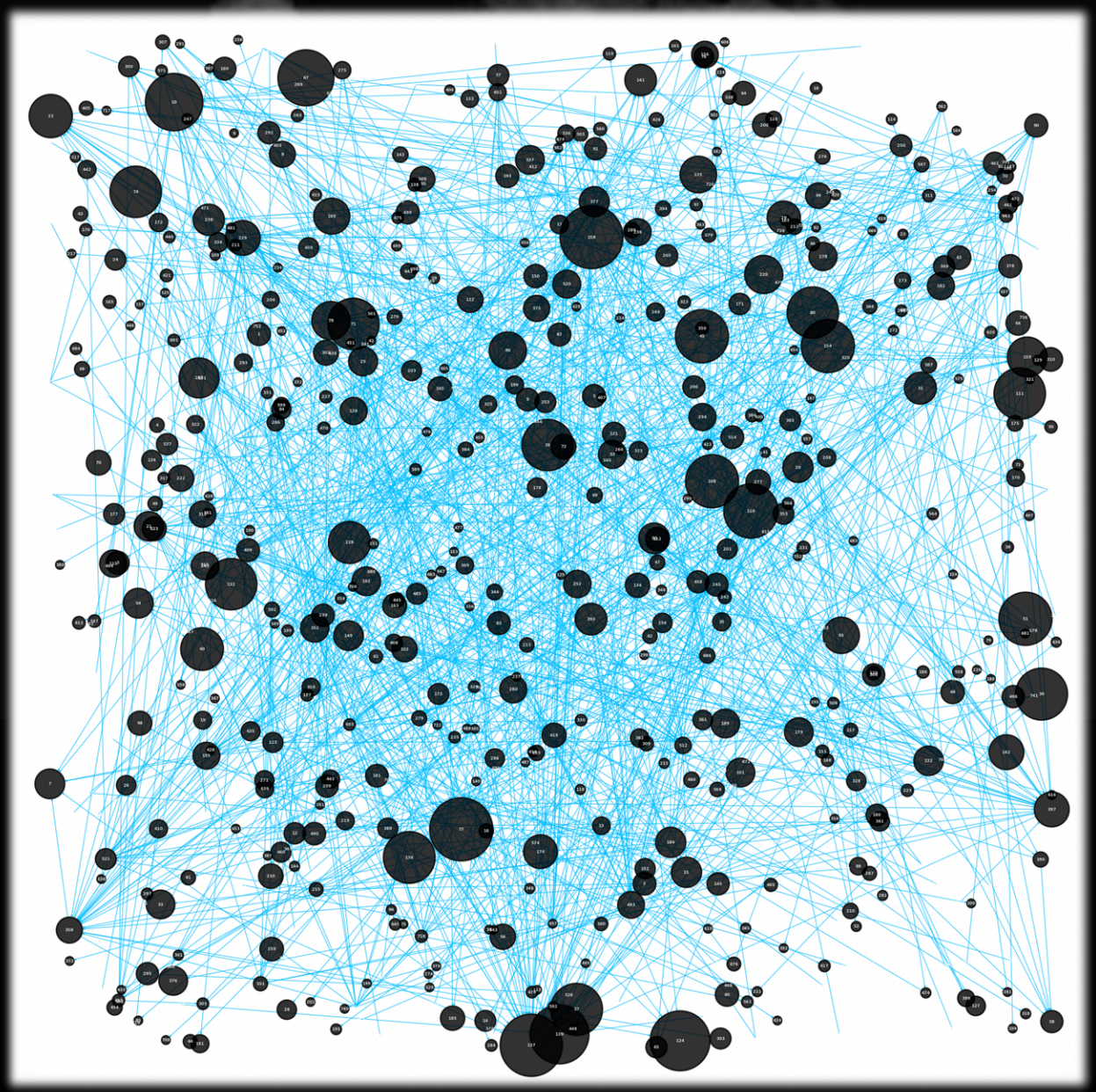
761 rows × 4 columns

NODE 77 HAS THE HIGHEST CLOSENESS CENTRALITY.

Visualization of closeness centrality:

```
size2=[c * 500000 for c in close_cent]
plt.figure(figsize=(50, 50))
nx.draw_random(g,with_labels=True,node_size=size2, node_color="black", node_shape="o", alpha=0.8, linewidths=4,
               font_color='white',font_weight="bold", width=2, edge_color="deepskyblue")
plt.show()
```

The greater the closeness centrality the bigger the node



Inference on closeness centrality:

- In twitter, the speed of the information spreading depends mainly on how many people retweet a post and how fast it is done. Therefore the higher the number of retweets the higher the chance that the information is spread. From the above graph it can be seen that the node with the label 77 has higher closeness centrality, therefore the probability that the information is spread to all the users is high if user 77 posts the information. The nodes 158, 137, 124 and 139 also have high closeness centrality.

About Eigenvector centrality:

- The Eigenvector centrality is used to identify the most influential node in a network.

- Each node is assigned an Eigen value based on the Eigen values of the nodes directly connected to it.

- The Eigenvector centrality of a node I calculated using:

$$x_I = \frac{1}{\lambda} \sum_{j}^{n} A_{ij} x_j$$

where $\lambda x = Ax$

```
eig_cent=list(nx.eigenvector_centrality(g).values())
v['eigenvector_centrality']=eig_cent
v.sort_values(by=['eigenvector_centrality'],ascending=False)
```

	vertices	degree_centrality	betweenness_centrality	closeness_centrality	eigenvector_centrality
185	67	0.005263	0.000000	0.054892	3.393272e-01
184	77	0.022368	0.000244	0.069740	3.391566e-01
278	110	0.005263	0.000000	0.051574	2.260752e-01
283	111	0.002632	0.000000	0.046464	1.695962e-01
29	10	0.031579	0.000000	0.057693	1.695618e-01
...	...	...	...	...	...
396	590	0.000000	0.000000	0.000000	8.031381e-16
395	566	0.000000	0.000000	0.000000	8.031381e-16
393	370	0.000000	0.000000	0.000000	8.031381e-16
392	367	0.000000	0.000000	0.000000	8.031381e-16
380	729	0.000000	0.000000	0.000000	8.031381e-16

761 rows × 5 columns

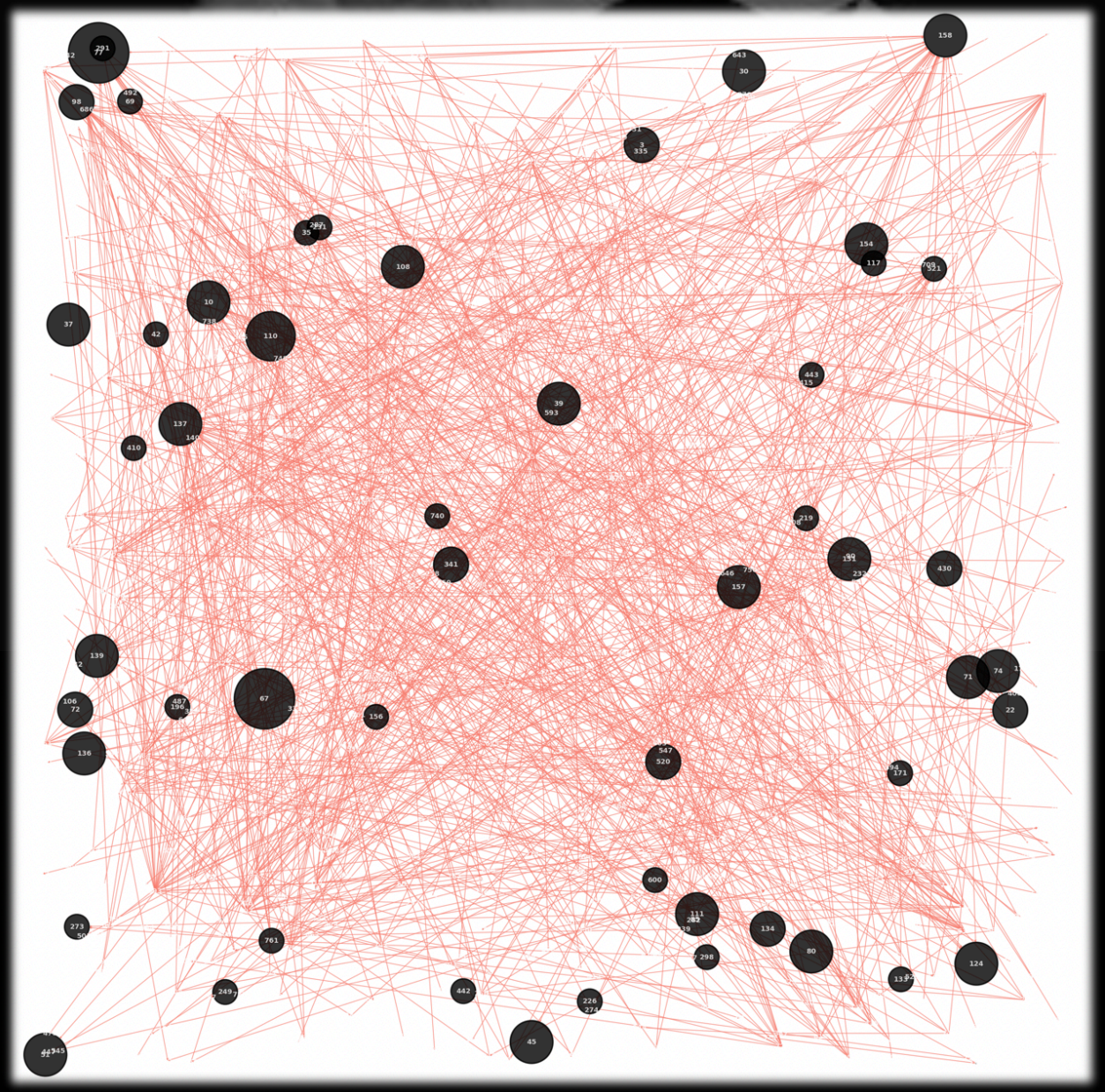
Node 67 has the highest Eigenvector centrality

Visualization of Eigenvector centrality:

The greater the Eigenvector centrality the bigger the node

```
size3=[e * 90000 for e in eig_cent]
plt.figure(figsize=(50, 50))
nx.draw_random(g,with_labels=True,node_size=size3, node_color="black", node_shape="o", alpha=0.8, linewidths=4,
               font_color='white',font_size=20,font_weight="bold", width=2, edge_color="salmon")
plt.show()
```

Inference on Eigenvector centrality:



- Eigenvector centrality is almost similar to degree centrality except for the fact that it takes the relevance of the nearby nodes into account when it is calculated. From the above graph it can be seen that node 67 has the highest Eigenvector centrality which tells us that user 67 is well connected with users who have connections with many influential people. The nodes 77, 110, 111 and 10 also have high eigenvector centralities indicating that these users also have influential connections.

About clustering centrality:

- Clustering centrality is used to measure how much the nodes cluster together in other words it is used to measure the connectivity of one node to all the other nodes.

- Local clustering coefficient off a node shows how close its neighbors are to being a clique.

- The local clustering coefficient is calculated by:

$$C_i = \frac{No\,.\,ofneighbors}{\frac{n(n-1)}{2}}$$

- The clustering coefficient of the whole network is calculated by:

$$C = \frac{1}{n}\sum c_i$$

- The maximum clustering coefficient in the retweet network is given by 1.0346932651815 37e-05.

```
n=len(g.nodes)+1
cluster=[]
adjacency_matrix=np.zeros([n,n],dtype=float)
edges=list(g.edges())
for i in edges:
    adjacency_matrix[int(i[0])][int(i[1])]=1
print('\n')
print("**********The adjacency matrix is given by**********\n")
print(adjacency_matrix)
print('\n')
print("********** CLUSTERING COEFFICIENT OF EACH NODE **********")
print('\n')
for i in range(len(adjacency_matrix)):
    neighbours=adjacency_matrix[i].sum()
    clust=((2*neighbours)/(n*(n-1)))
    cluster.append(clust)
    if(clust>0):
        print(f"The clustering coefficient of node {i} is {clust}")
print('\n')
print("********** THE CLUSTERING COEFFEICIENT OF THE LEFT OUT NODES IS ZERO **********")
print('\n')
print("********** CLUSTERING COEFFICIENT OF THE WHOLE GRAPH **********\n")
print(f"The clustering coefficient of the whole graph is {sum(cluster)/n}")
```

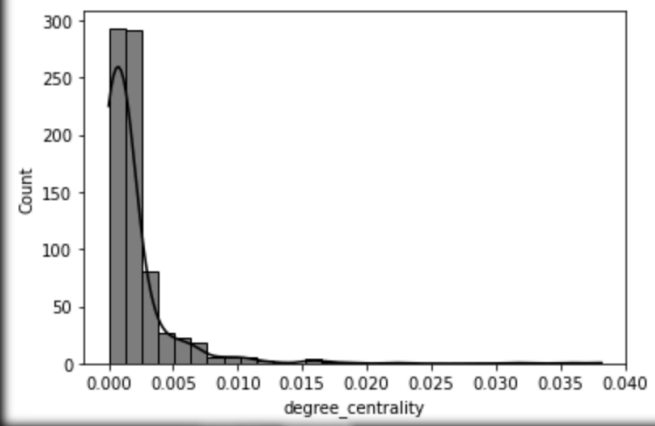
- The average clustering coefficient of the network is given by 4.662003775752353 5e-06

The Range of different centralities:

- Degree centrality:

```
sns.histplot(x=deg,kde=True,color='k',alpha=0.5,bins=30).set(xlabel='degree_centrality')
```

```
[Text(0.5, 0, 'degree_centrality')]
```

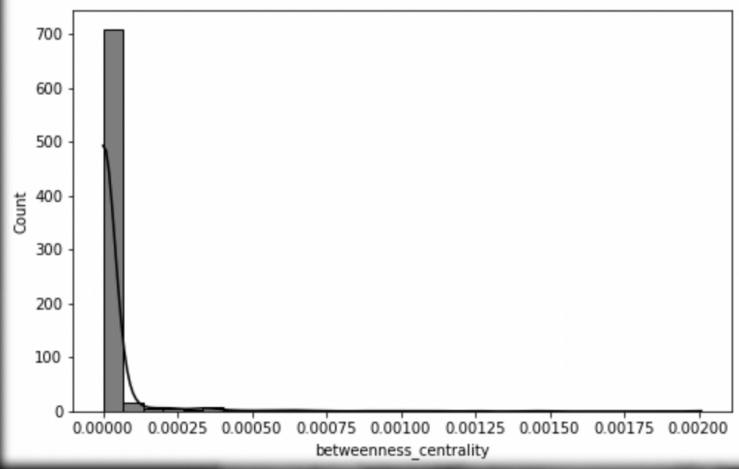


The range of the degree centralities of the 761 nodes is in between 0 to 0.038(approximately)

- Betweenness centrality:

The range of the betweenness centralities of the 761 nodes is in between 0 to 0.00028(approximately)

```
plt.figure(figsize=(8,5))
sns.histplot(x=bet_cent,kde=True,color='k',alpha=0.5,bins=30).set(xlabel='betweenness_centrality')
plt.show()
```

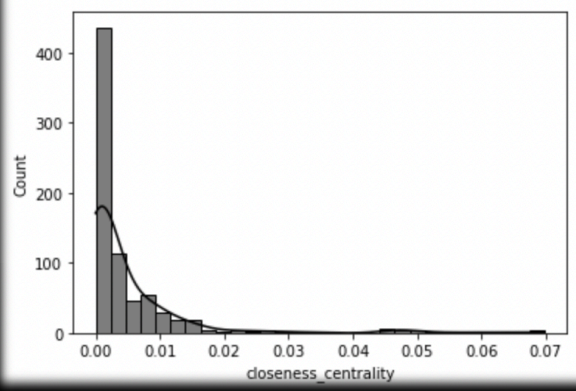


- **Closeness centrality:**

- **The range of the closeness centralities of the 761nodes is in between 0 to 0.02 and 0.045 to 0.07(approximately)**

```
sns.histplot(x=close_cent,kde=True,color='k',alpha=0.5,bins=30).set(xlabel='closeness_centrality')
```

```
[Text(0.5, 0, 'closeness_centrality')]
```

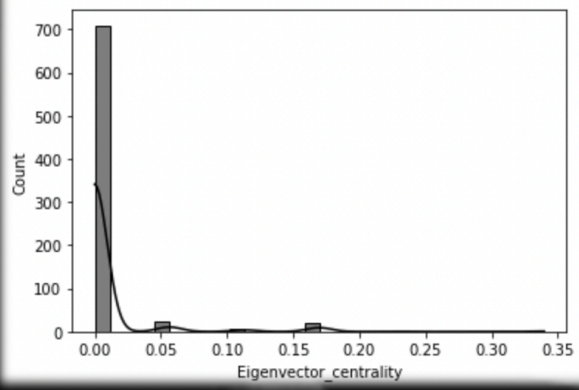


- Eigenvector centrality:

The Eigen vector centrality values of various nodes mostly lie on 0 or 0.05 or 0.17

```
sns.histplot(x=eig_cent,kde=True,color='k',alpha=0.5,bins=30).set(xlabel='Eigenvector_centrality')
```

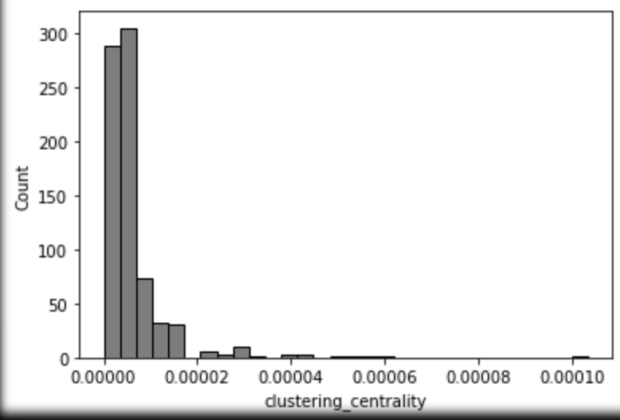
```
[Text(0.5, 0, 'Eigenvector_centrality')]
```



- Clustering centrality:

```
sns.histplot(x=cluster,color='k',alpha=0.5,bins=30).set(xlabel='clustering_centrality')
```

```
[Text(0.5, 0, 'clustering_centrality')]
```



The range of the clustering centralities of the 761nodes is in between 0 to 0.000061(approximately

Conclusion:

- Taking all the graph centralities into account it can be seen that the nodes 137, 158, 10, 77, 229, 67, 110 are very important as this set has high graph centrality measures. Therefore it can be concluded that graph centrality can be used to find the most influential and important nodes in a network.