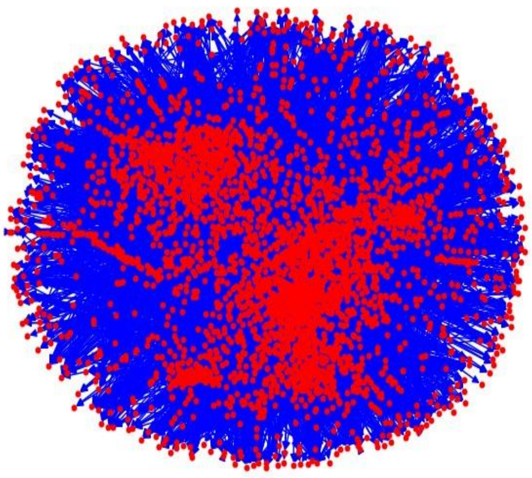


SNA_Origin

SNA Project Round-1

Prepared By:-

Kartik Gupta (21ucs107)
Aryan Gupta (21ucs248)
Mehul Khera (21ucs127)
Ashutosh Solanki(21ucs039)

Datasets Used:-

1. **Social circles: Facebook**

   https://snap.stanford.edu/data/ego-Facebook.html

2. **Gnutella peer-to-peer network**

   https://snap.stanford.edu/data/p2p-Gnutella08.html

Github Link:- https://github.com/algoviber/SNA_PROJECT

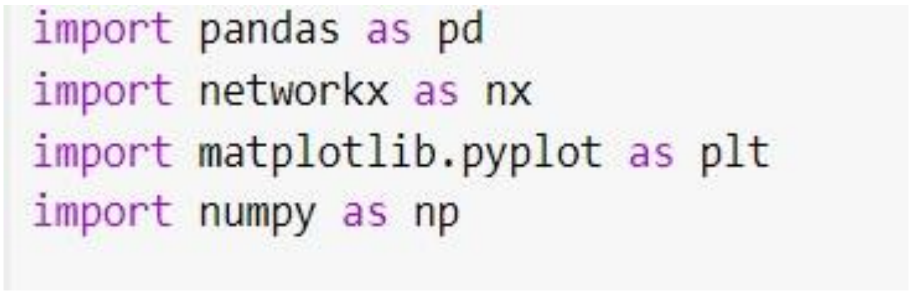
Dataset 1 - Social circles: Facebook

This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. The dataset includes node features (profiles), circles, and ego networks.

Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, while feature vectors from this dataset have been provided, the interpretation of those features has been obscured. For instance, where the original dataset may have contained a feature "political=Democratic Party", the new data would simply contain "political=anonymized feature 1". Thus, using the anonymized data it is possible to determine whether two users have the same political affiliations, but not what their individual political affiliations represent.

Dataset statistics	
Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Python Libraries/Packages Used



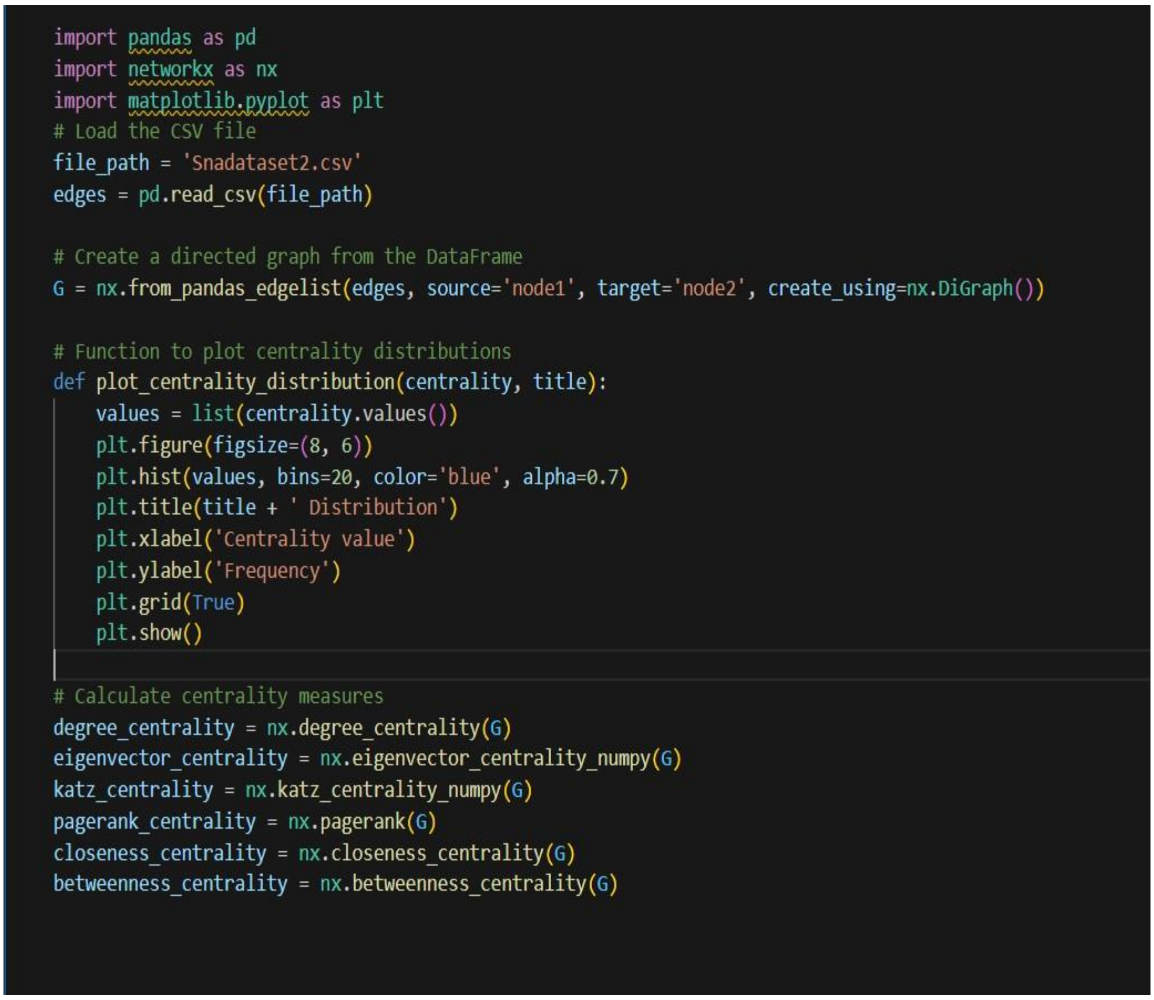
```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
```

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

NetworkX is a package for the Python programming language that's used to create, manipulate, and study the structure, dynamics, and functions of complex graph networks.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update.

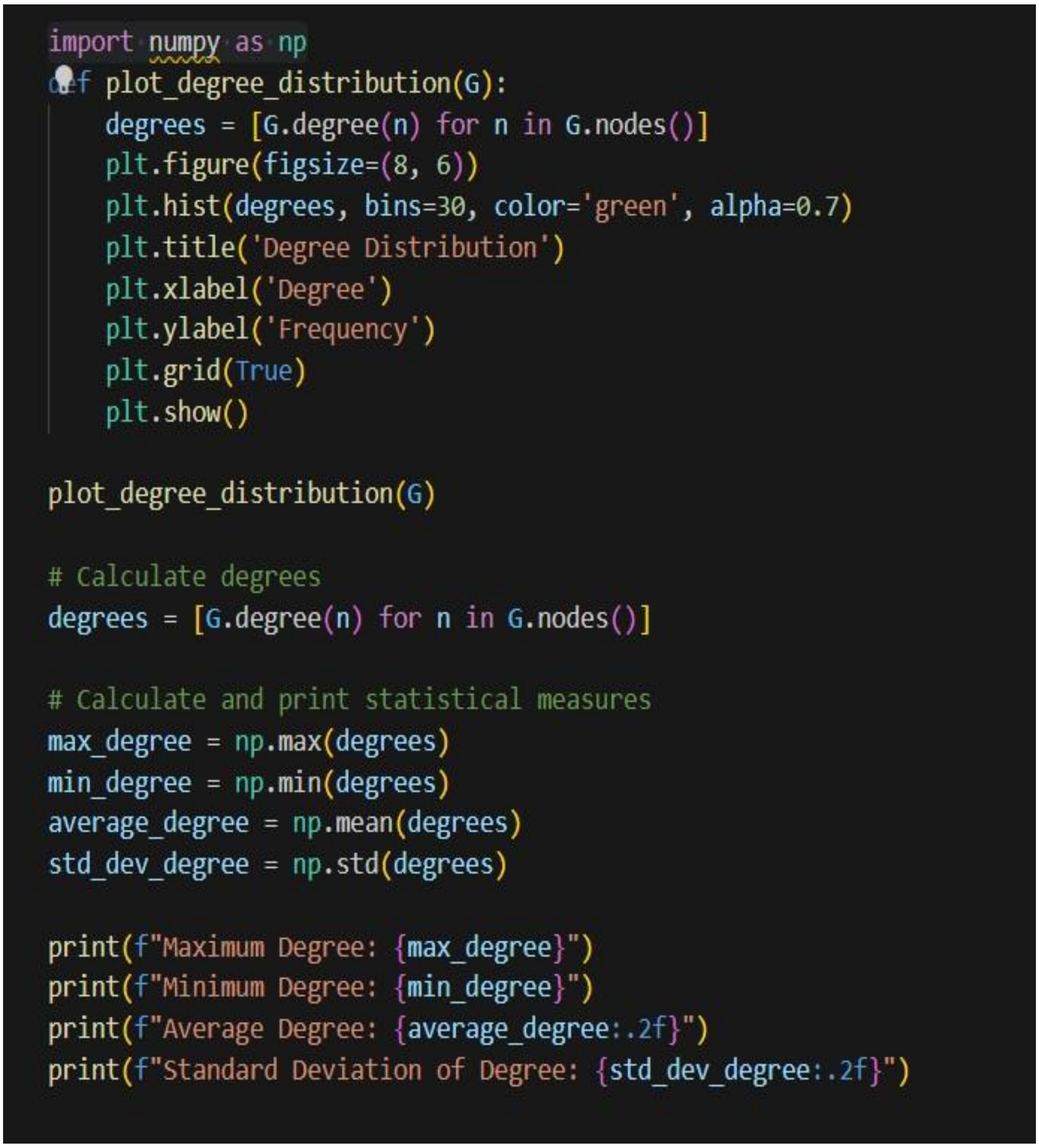
NumPy is a module for Python that allows you to work with multidimensional arrays and matrices. It's perfect for scientific or mathematical calculations because it's fast and efficient. In addition, NumPy includes support for signal processing and linear algebra operations.



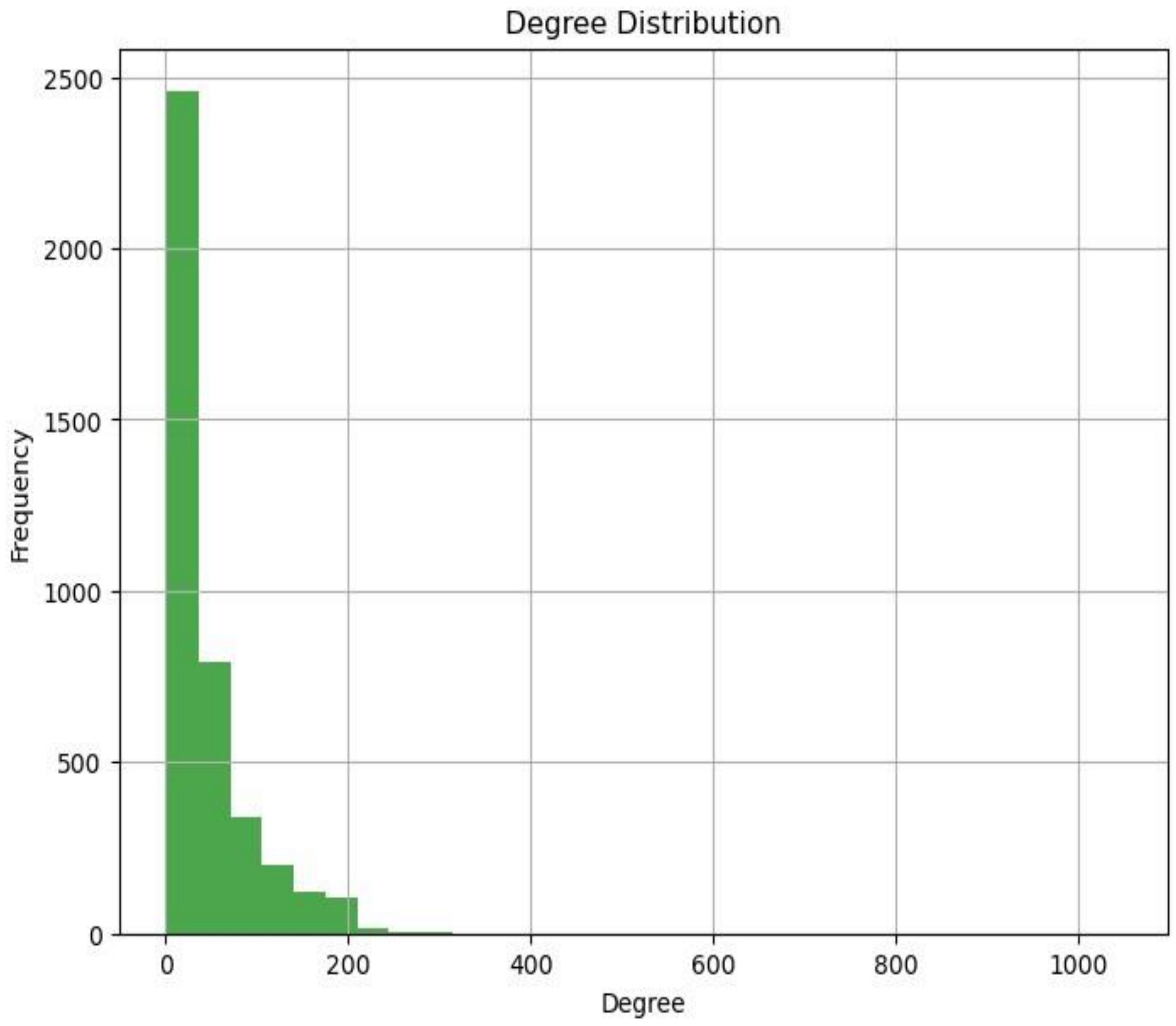
```python
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
# Load the CSV file
file_path = 'Snadataset2.csv'
edges = pd.read_csv(file_path)

# Create a directed graph from the DataFrame
G = nx.from_pandas_edgelist(edges, source='node1', target='node2', create_using=nx.DiGraph())

# Function to plot centrality distributions
def plot_centrality_distribution(centrality, title):
    values = list(centrality.values())
    plt.figure(figsize=(8, 6))
    plt.hist(values, bins=20, color='blue', alpha=0.7)
    plt.title(title + ' Distribution')
    plt.xlabel('Centrality value')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()

# Calculate centrality measures
degree_centrality = nx.degree_centrality(G)
eigenvector_centrality = nx.eigenvector_centrality_numpy(G)
katz_centrality = nx.katz_centrality_numpy(G)
pagerank_centrality = nx.pagerank(G)
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)
```

1. **Data Handling**: The script begins by loading data from a CSV file named 'Snadataset2.csv' using pandas. This data will be used to construct a network graph.
2. **Graph Construction**: Using NetworkX, the script creates a directed graph (**DiGraph**) from the loaded data. The nodes and edges of the graph are derived from the DataFrame created by reading the CSV file.
3. **Centrality Calculation**: Various centrality measures are computed for the nodes in the graph. These measures include degree centrality, eigenvector centrality, Katz centrality, PageRank centrality, closeness centrality, and betweenness centrality. Each of these measures quantifies the importance or centrality of nodes in the network based on different criteria.
4. **Centrality Visualization**: The script defines a function to plot the distribution of centrality values. It then iterates over the centrality measures calculated earlier and generates a histogram for each measure, visualizing the distribution of centrality values across the nodes in the network.

Plotting Degree Distribution:



```
import numpy as np
def plot_degree_distribution(G):
    degrees = [G.degree(n) for n in G.nodes()]
    plt.figure(figsize=(8, 6))
    plt.hist(degrees, bins=30, color='green', alpha=0.7)
    plt.title('Degree Distribution')
    plt.xlabel('Degree')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()

plot_degree_distribution(G)

# Calculate degrees
degrees = [G.degree(n) for n in G.nodes()]

# Calculate and print statistical measures
max_degree = np.max(degrees)
min_degree = np.min(degrees)
average_degree = np.mean(degrees)
std_dev_degree = np.std(degrees)

print(f"Maximum Degree: {max_degree}")
print(f"Minimum Degree: {min_degree}")
print(f"Average Degree: {average_degree:.2f}")
print(f"Standard Deviation of Degree: {std_dev_degree:.2f}")
```



1. **Plot Degree Distribution**: **plot_degree_distribution** function plots the degree distribution of nodes in **G**.
2. **Calculate Degree Statistics**: It computes statistical measures (maximum, minimum, average, and standard deviation) of the degrees of nodes in **G**. 3. **Output Degree Statistics**: It prints the calculated statistical measures.

Inference:

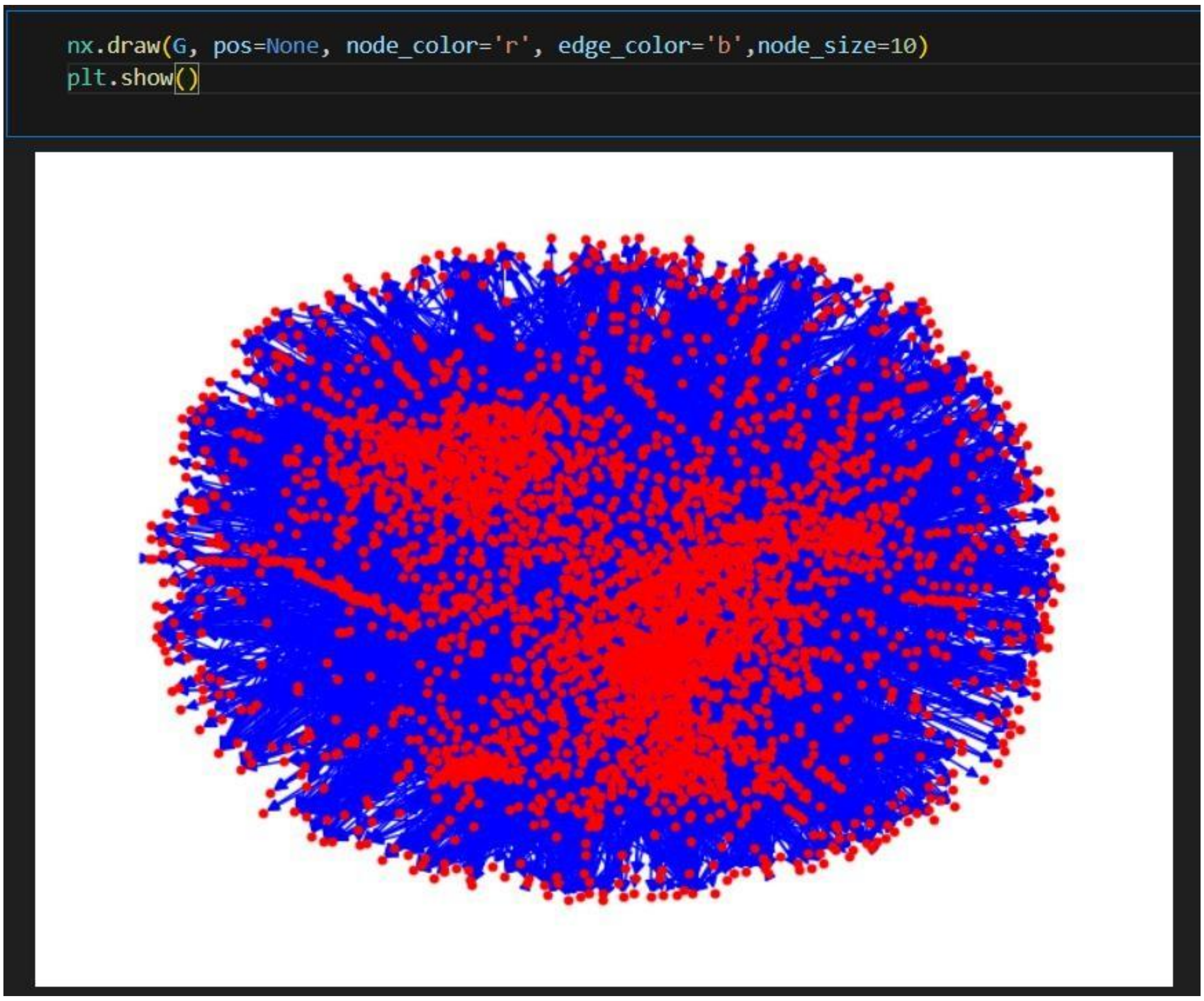
Maximum Degree: 1045

Minimum Degree: 1

Average Degree: 43.69

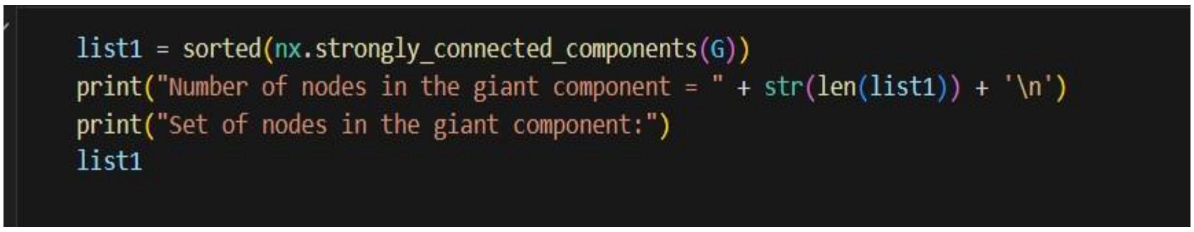
Standard Deviation of Degree: 52.41

Network Graph Visualization:

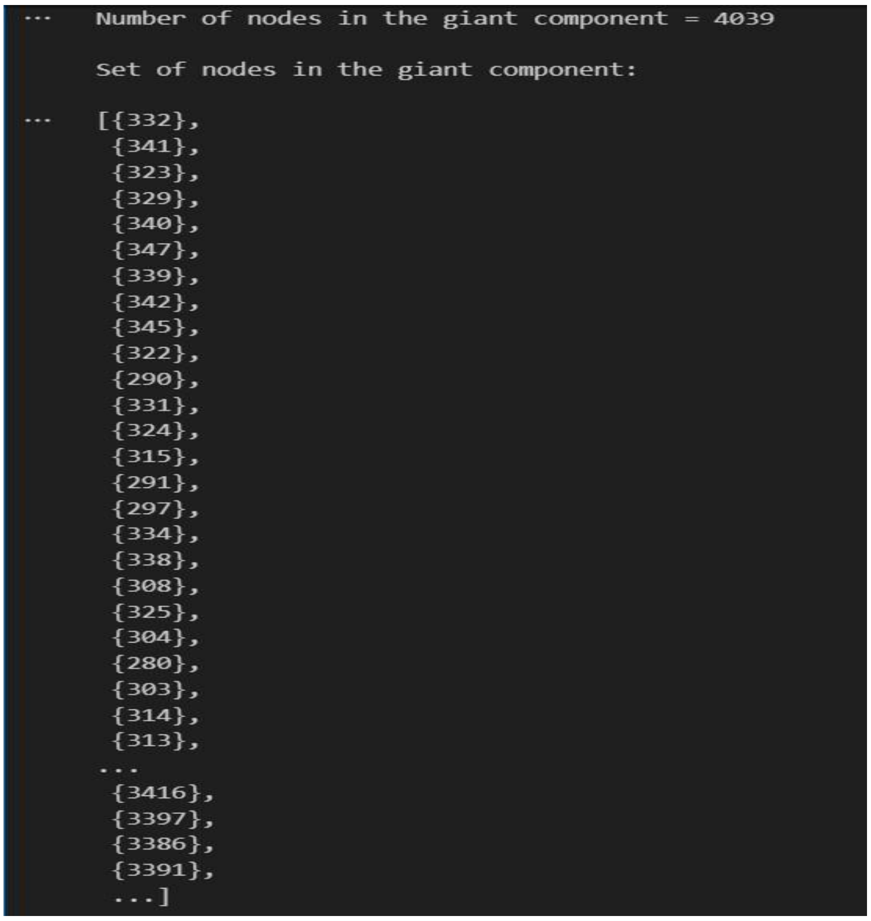


```
nx.draw(G, pos=None, node_color='r', edge_color='b',node_size=10)
plt.show()
```

This code snippet utilizes NetworkX's draw function to visualize the graph G. It sets the node color to red (node_color='r'), edge color to blue (edge_color='b'), and node size to 10 (node_size=10). Finally, it displays the plot using plt.show().



```
list1 = sorted(nx.strongly_connected_components(G))
print("Number of nodes in the giant component = " + str(len(list1)) + '\n')
print("Set of nodes in the giant component:")
list1
```



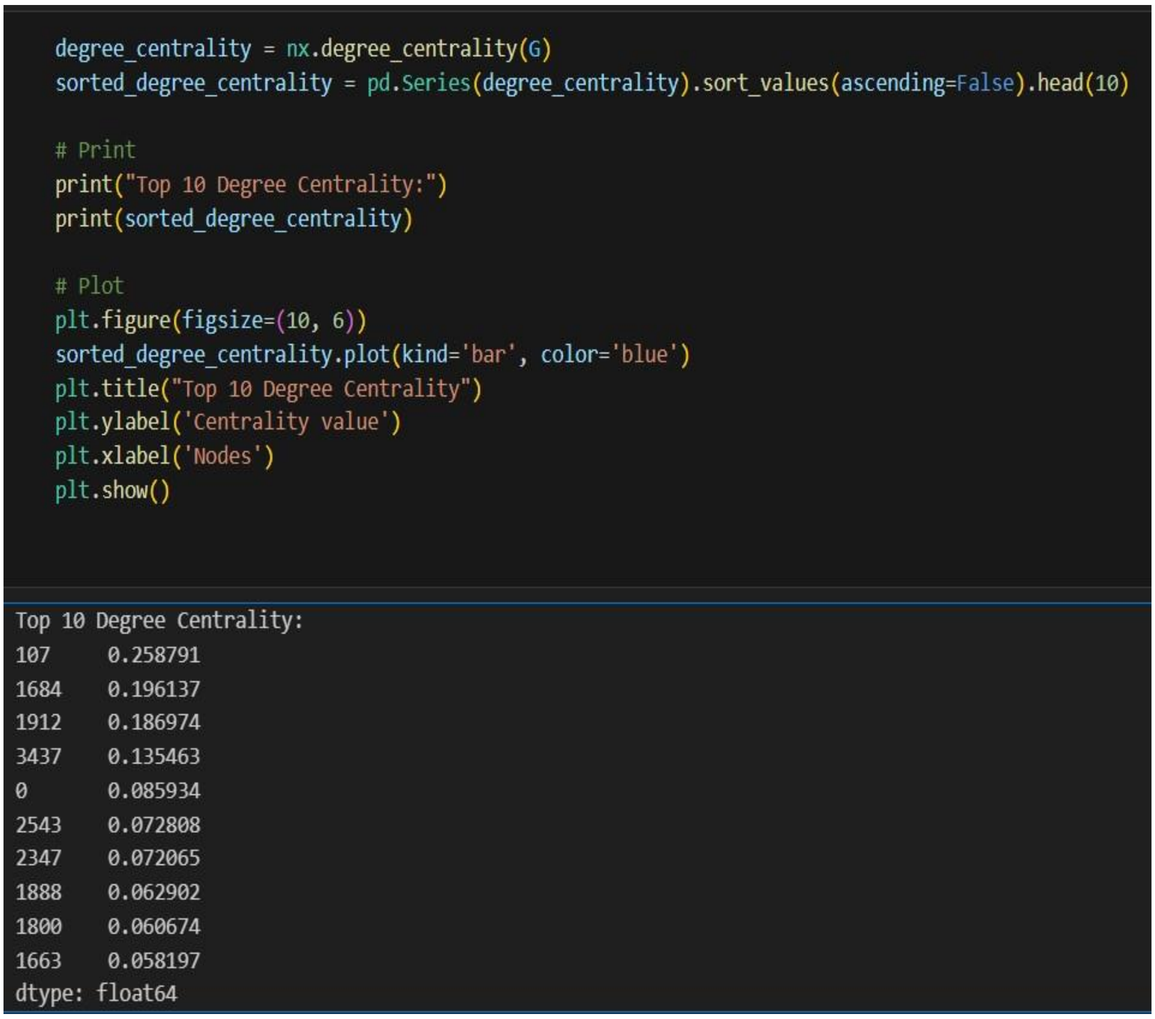
```
...   Number of nodes in the giant component = 4039

      Set of nodes in the giant component:

...   [{332},
       {341},
       {323},
       {329},
       {340},
       {347},
       {339},
       {342},
       {345},
       {322},
       {290},
       {331},
       {324},
       {315},
       {291},
       {297},
       {334},
       {338},
       {308},
       {325},
       {304},
       {280},
       {303},
       {314},
       {313},
      ...
       {3416},
       {3397},
       {3386},
       {3391},
       ...]
```

This code snippet calculates the strongly connected components of the graph G using NetworkX's strongly_connected_components function and sorts them. Then, it prints the number of nodes in the giant component and the set of nodes in the giant component.

Here's a breakdown of the output:

Number of nodes in the giant component: This line displays the total number of nodes in the largest strongly connected component of the graph G.
Set of nodes in the giant component: It lists the nodes present in the giant component.
This information helps in understanding the structure of the graph and the extent of its connectivity. Strongly connected components represent subsets of nodes where each node is reachable from every other node within the subset. The largest strongly connected component, often

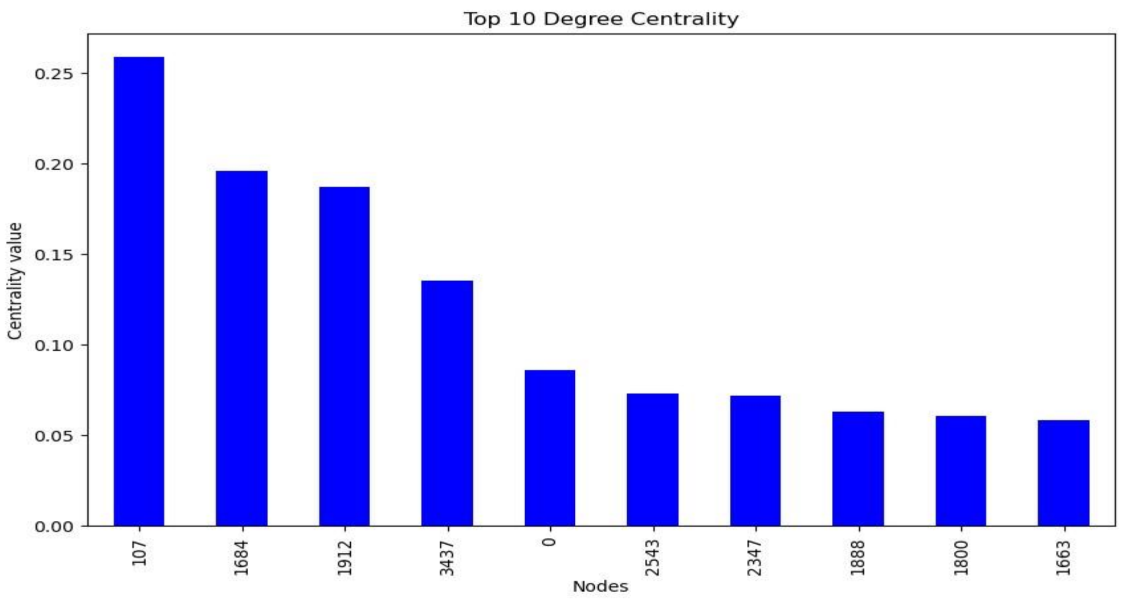
referred to as the giant component, is of particular interest in analyzing the overall connectivity and resilience of the network.



```python
degree_centrality = nx.degree_centrality(G)
sorted_degree_centrality = pd.Series(degree_centrality).sort_values(ascending=False).head(10)

# Print
print("Top 10 Degree Centrality:")
print(sorted_degree_centrality)

# Plot
plt.figure(figsize=(10, 6))
sorted_degree_centrality.plot(kind='bar', color='blue')
plt.title("Top 10 Degree Centrality")
plt.ylabel('Centrality value')
plt.xlabel('Nodes')
plt.show()
```

```
Top 10 Degree Centrality:
107     0.258791
1684    0.196137
1912    0.186974
3437    0.135463
0       0.085934
2543    0.072808
2347    0.072065
1888    0.062902
1800    0.060674
1663    0.058197
dtype: float64
```

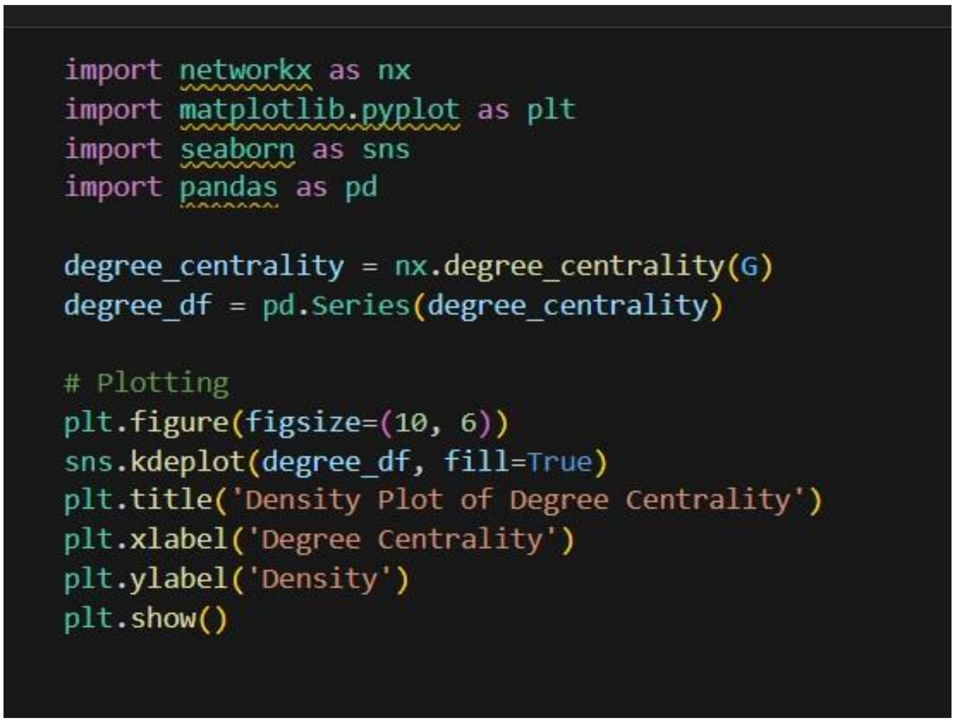
Plot of Top 10 Degree Centralities:



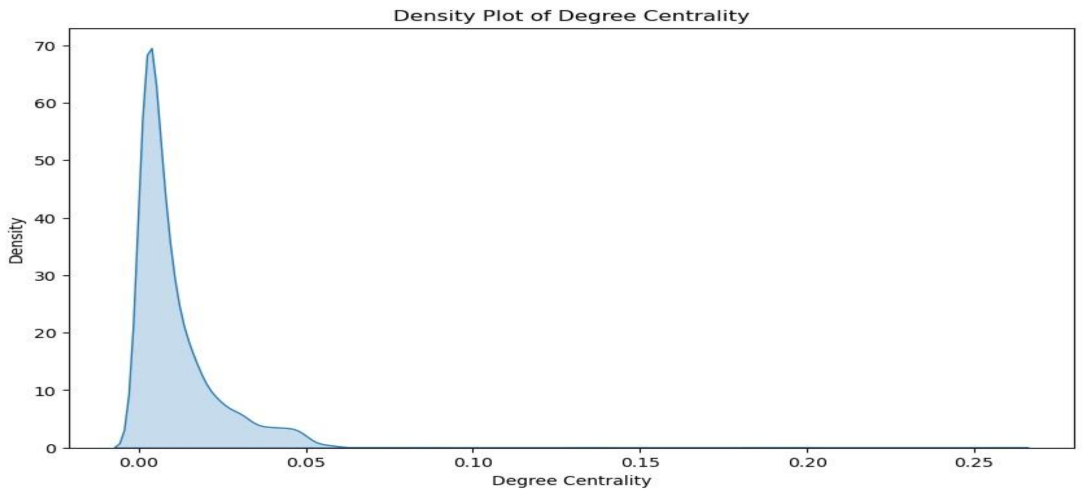
Inferences:

- **Popular users:** Users on the far right side of the graph, with the highest degree centrality, are likely the most popular on Facebook. They have a very large number of friends.
- **Average users:** The center of the graph likely represents the majority of Facebook users who have a typical number of friends.
- **Less connected users:** Users on the left side of the graph, with low degree centrality, have a smaller number of friends on Facebook.

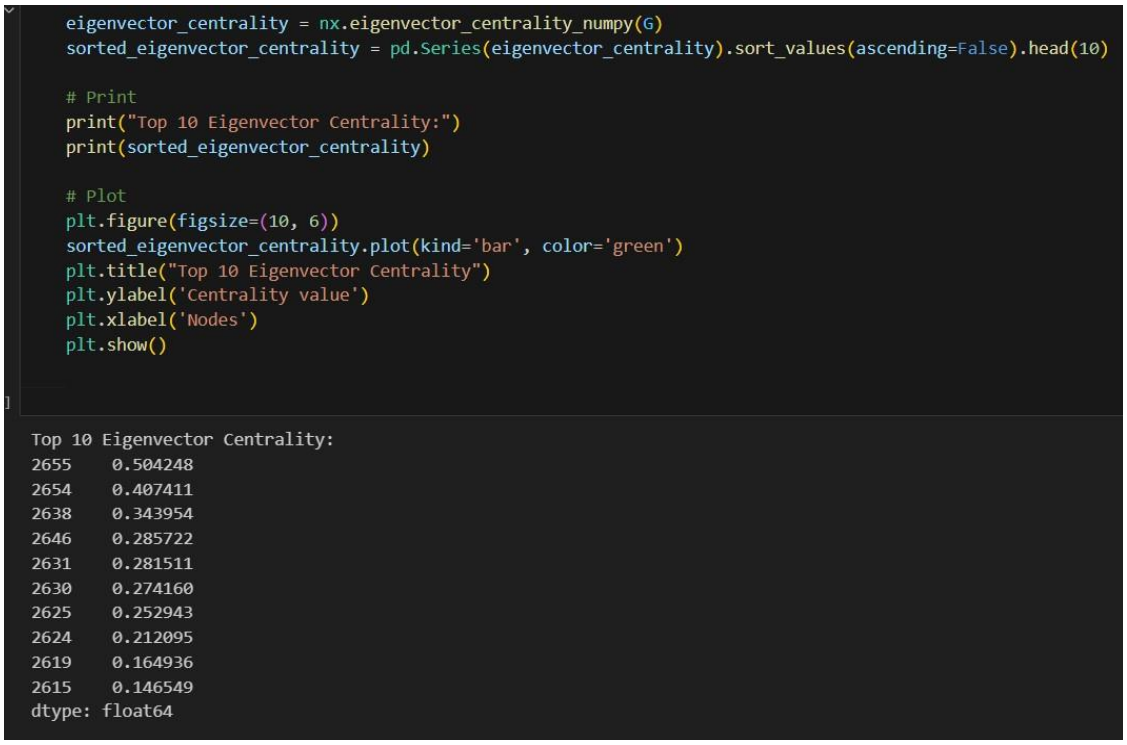
Density Plot of Degree Centrality:



```python
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

degree_centrality = nx.degree_centrality(G)
degree_df = pd.Series(degree_centrality)

# Plotting
plt.figure(figsize=(10, 6))
sns.kdeplot(degree_df, fill=True)
plt.title('Density Plot of Degree Centrality')
plt.xlabel('Degree Centrality')
plt.ylabel('Density')
plt.show()
```

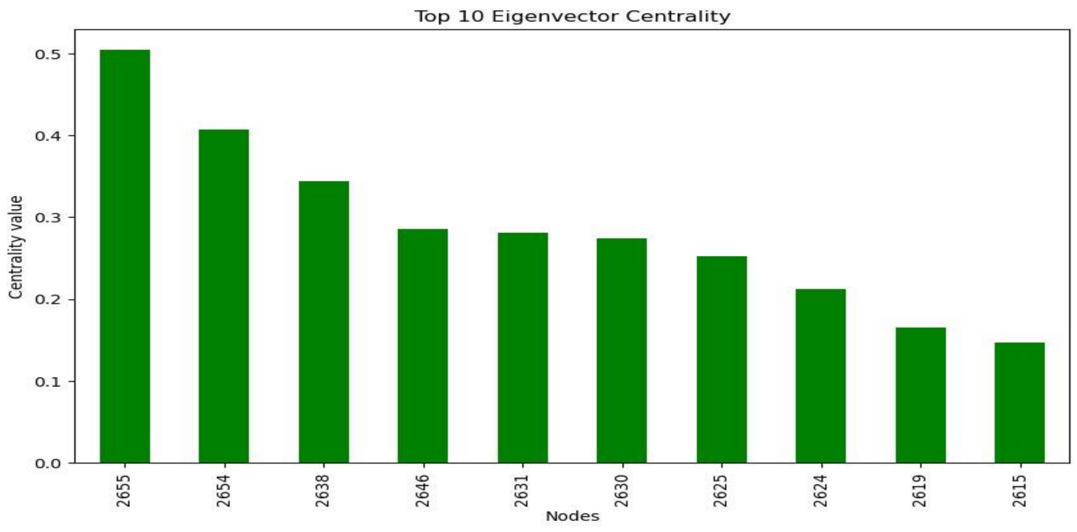


Top 10 Eigen Vector Centralities:



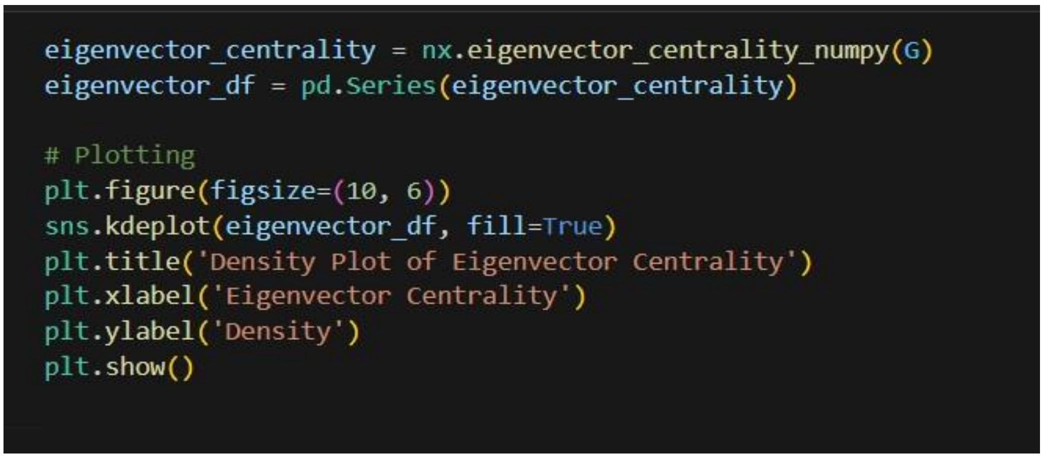
```
eigenvector_centrality = nx.eigenvector_centrality_numpy(G)
sorted_eigenvector_centrality = pd.Series(eigenvector_centrality).sort_values(ascending=False).head(10)

# Print
print("Top 10 Eigenvector Centrality:")
print(sorted_eigenvector_centrality)

# Plot
plt.figure(figsize=(10, 6))
sorted_eigenvector_centrality.plot(kind='bar', color='green')
plt.title("Top 10 Eigenvector Centrality")
plt.ylabel('Centrality value')
plt.xlabel('Nodes')
plt.show()
```

```
Top 10 Eigenvector Centrality:
2655    0.504248
2654    0.407411
2638    0.343954
2646    0.285722
2631    0.281511
2630    0.274160
2625    0.252943
2624    0.212095
2619    0.164936
2615    0.146549
dtype: float64
```

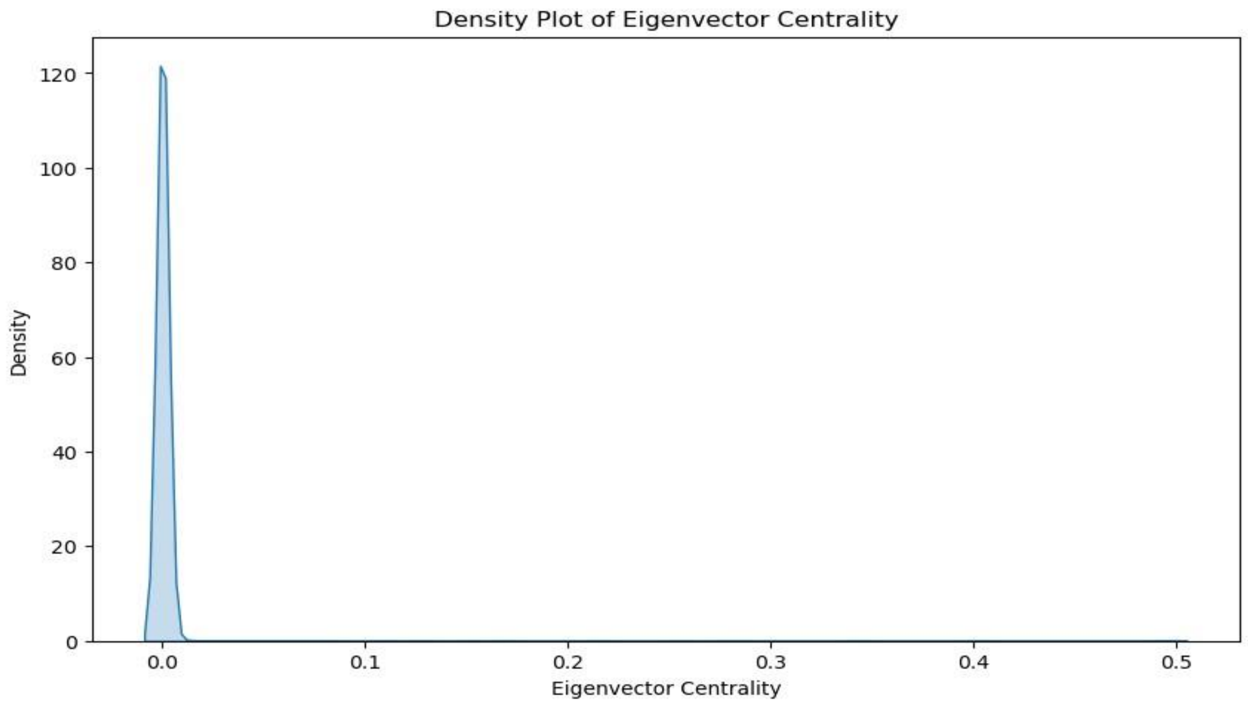


Inferences:

Users on the far right side of the graph, with the highest eigenvector centrality, are likely the most influential on Facebook. But unlike degree centrality (which focuses on the number of friends), eigenvector centrality goes beyond just popularity. These influential users are connected to other influential people, forming a network of significant impact. Imagine a celebrity who befriends other celebrities and public figures – they'd likely score high on eigenvector centrality.



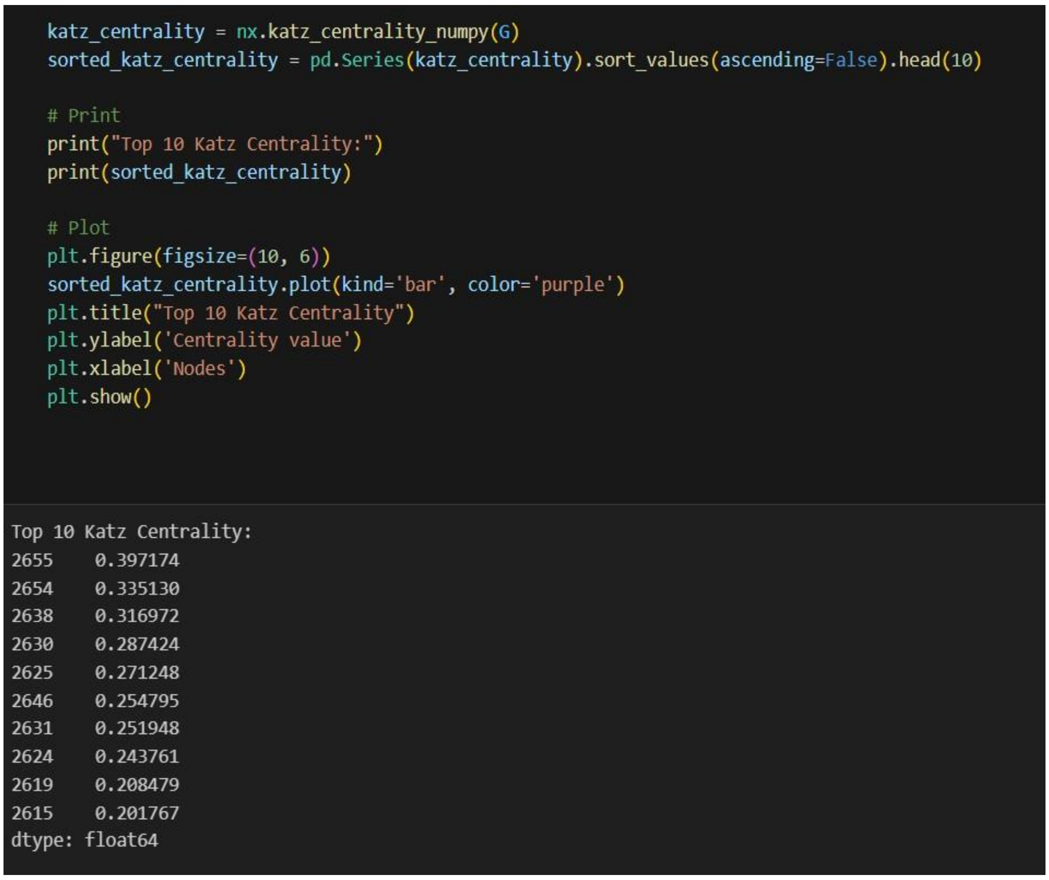
```
eigenvector_centrality = nx.eigenvector_centrality_numpy(G)
eigenvector_df = pd.Series(eigenvector_centrality)

# Plotting
plt.figure(figsize=(10, 6))
sns.kdeplot(eigenvector_df, fill=True)
plt.title('Density Plot of Eigenvector Centrality')
plt.xlabel('Eigenvector Centrality')
plt.ylabel('Density')
plt.show()
```



Inferences:

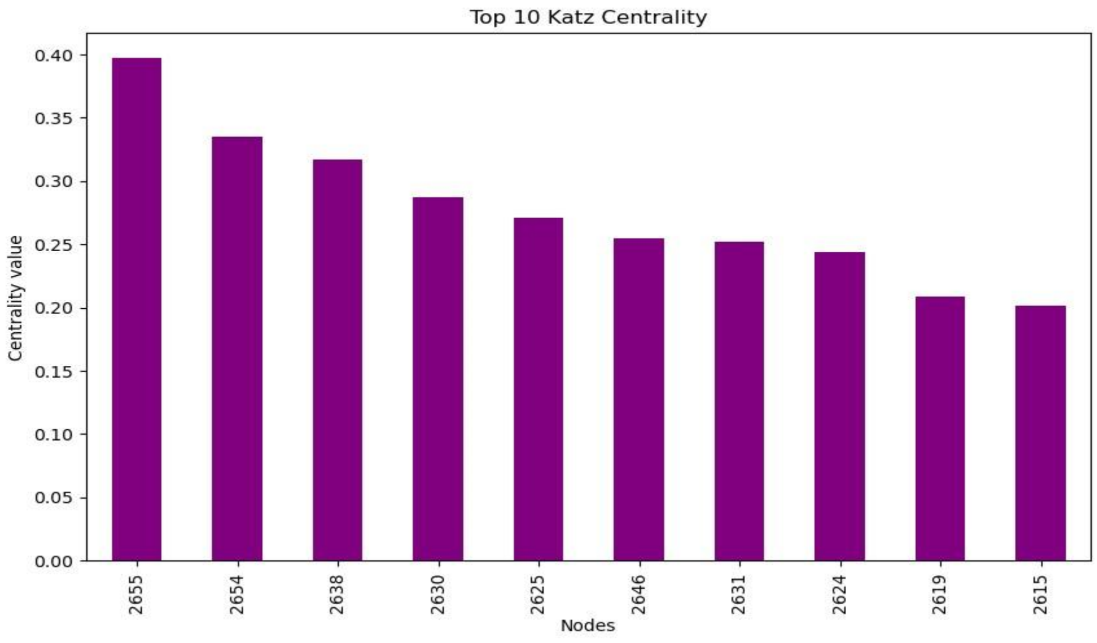
- **Most users have low to moderate influence:** The majority of users likely fall in the center of the distribution, with eigenvector centrality scores between 0.1 and 0.2. This suggests that they are connected to some moderately influential users, but don't wield a huge amount of influence themselves.
- **There's a small number of highly influential users:** The tail of the distribution on the right side suggests a small number of users with very high eigenvector centrality scores. These are the heavy hitters on Facebook, likely celebrities, public figures, or brands with a massive network of connections to other influential users.
- **There might be niche influencers:** The plot doesn't reveal specific communities, but bumps or clusters of users in the distribution could indicate communities or hubs of users with high centrality around specific topics or niches. These users may not be widely known but hold significant influence within their particular areas of interest.

Top 10 Katz Centralities:

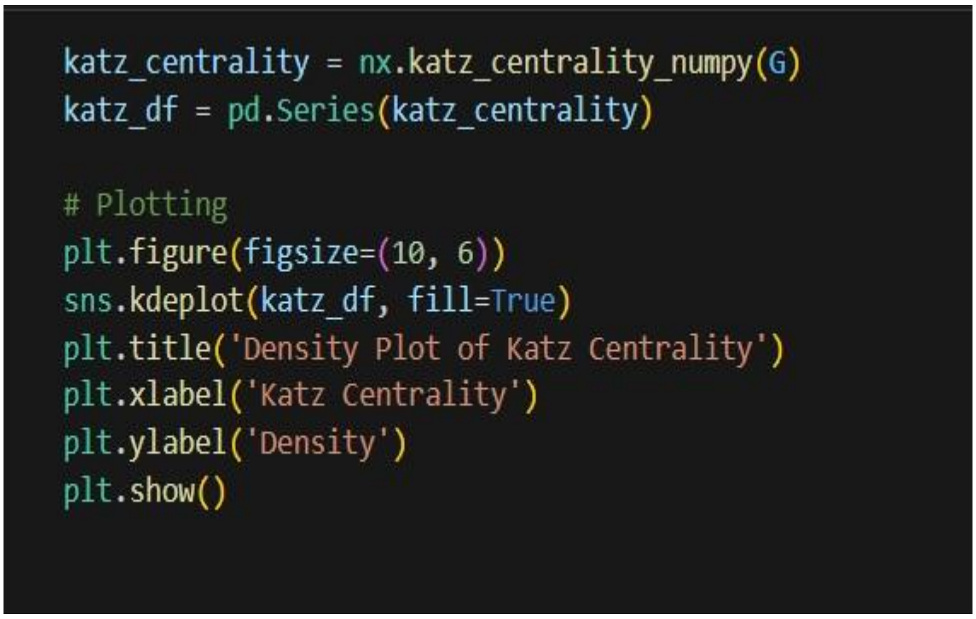


```
katz_centrality = nx.katz_centrality_numpy(G)
sorted_katz_centrality = pd.Series(katz_centrality).sort_values(ascending=False).head(10)

# Print
print("Top 10 Katz Centrality:")
print(sorted_katz_centrality)

# Plot
plt.figure(figsize=(10, 6))
sorted_katz_centrality.plot(kind='bar', color='purple')
plt.title("Top 10 Katz Centrality")
plt.ylabel('Centrality value')
plt.xlabel('Nodes')
plt.show()
```

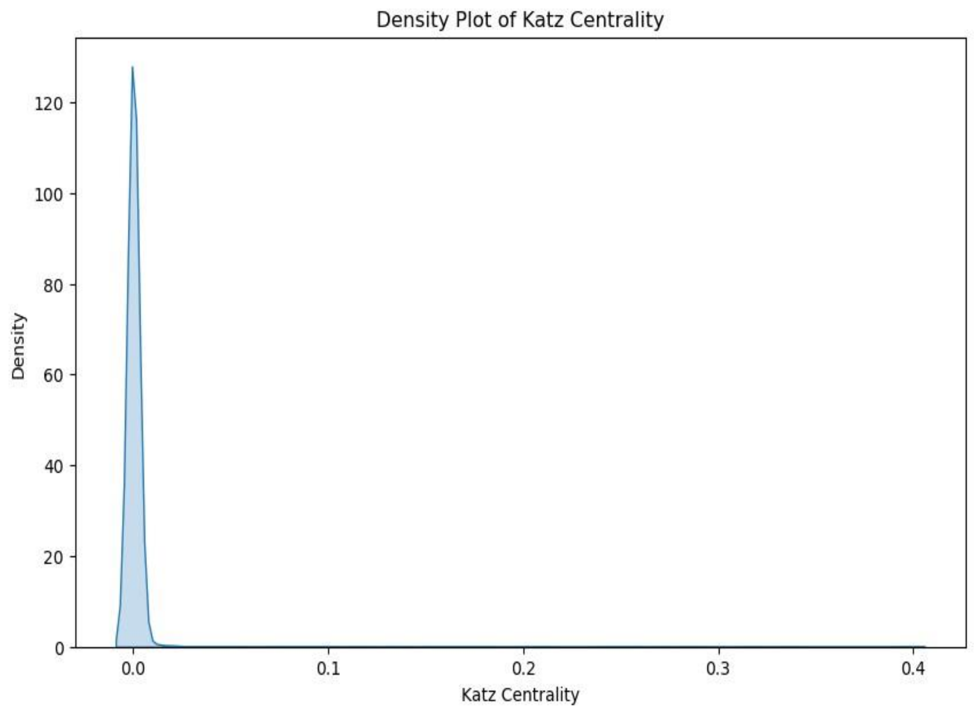
```
Top 10 Katz Centrality:
2655    0.397174
2654    0.335130
2638    0.316972
2630    0.287424
2625    0.271248
2646    0.254795
2631    0.251948
2624    0.243761
2619    0.208479
2615    0.201767
dtype: float64
```



Density Plot:



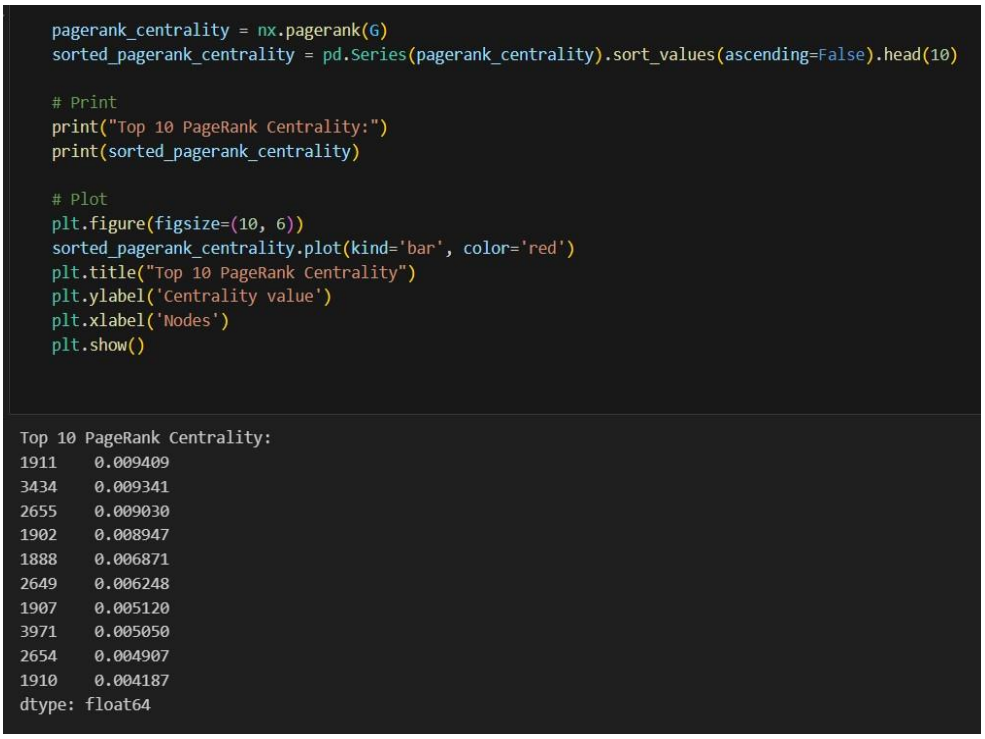
```python
katz_centrality = nx.katz_centrality_numpy(G)
katz_df = pd.Series(katz_centrality)

# Plotting
plt.figure(figsize=(10, 6))
sns.kdeplot(katz_df, fill=True)
plt.title('Density Plot of Katz Centrality')
plt.xlabel('Katz Centrality')
plt.ylabel('Density')
plt.show()
```



Inferences:

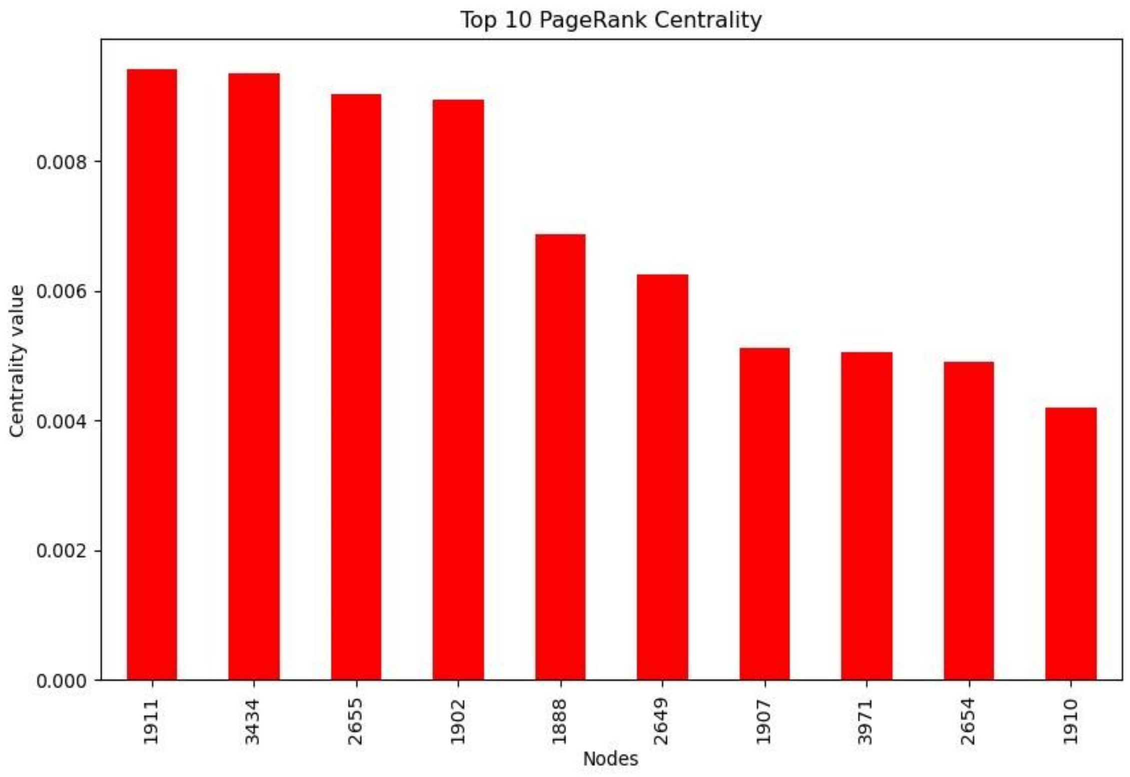
By analyzing the Katz centrality plot of a Facebook network, we gain insights into the distribution of connectivity and reach among users. It reveals a spectrum, with many users having limited reach and a smaller group acting as potential information hubs due to their extensive connections. This knowledge can be valuable for understanding information flow and spread on the platform.

Top 10 PageRank Centralities:

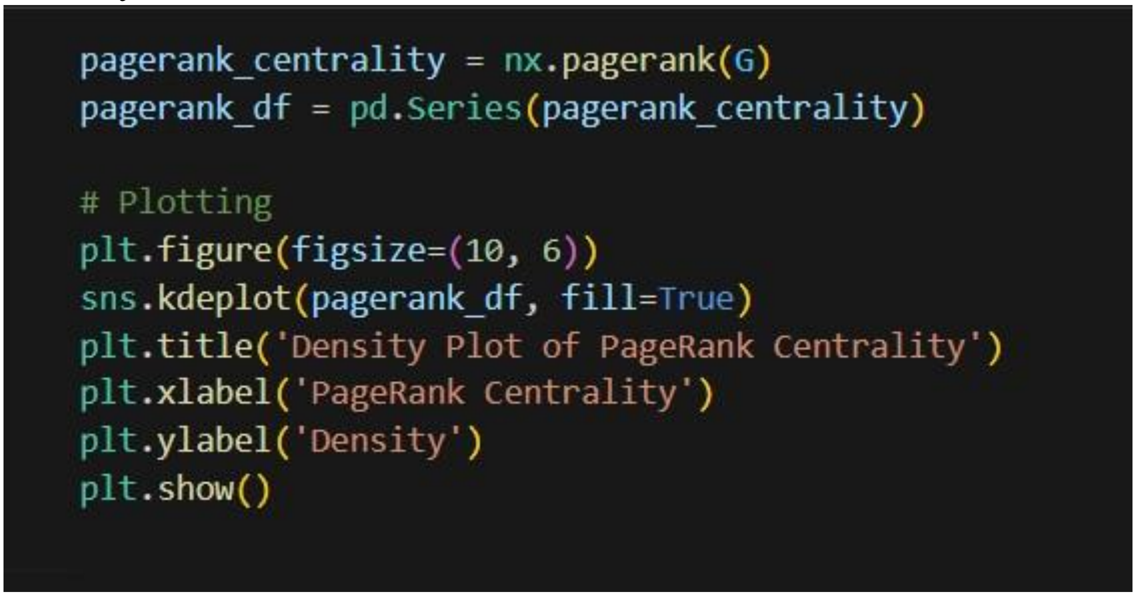


```
pagerank_centrality = nx.pagerank(G)
sorted_pagerank_centrality = pd.Series(pagerank_centrality).sort_values(ascending=False).head(10)

# Print
print("Top 10 PageRank Centrality:")
print(sorted_pagerank_centrality)

# Plot
plt.figure(figsize=(10, 6))
sorted_pagerank_centrality.plot(kind='bar', color='red')
plt.title("Top 10 PageRank Centrality")
plt.ylabel('Centrality value')
plt.xlabel('Nodes')
plt.show()
```

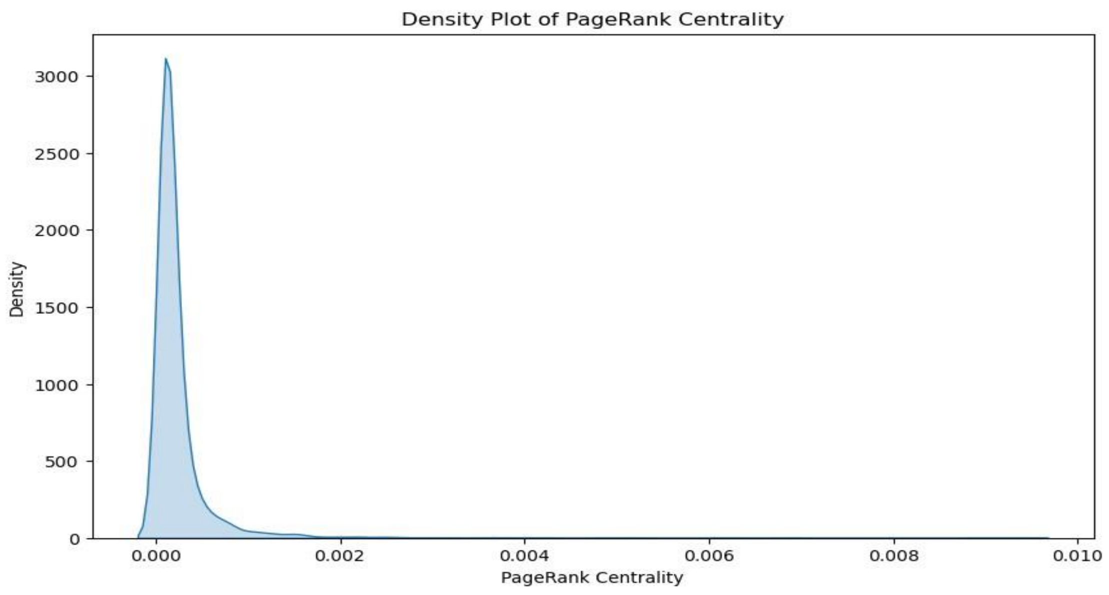
```
Top 10 PageRank Centrality:
1911    0.009409
3434    0.009341
2655    0.009030
1902    0.008947
1888    0.006871
2649    0.006248
1907    0.005120
3971    0.005050
2654    0.004907
1910    0.004187
dtype: float64
```



Density Plot:



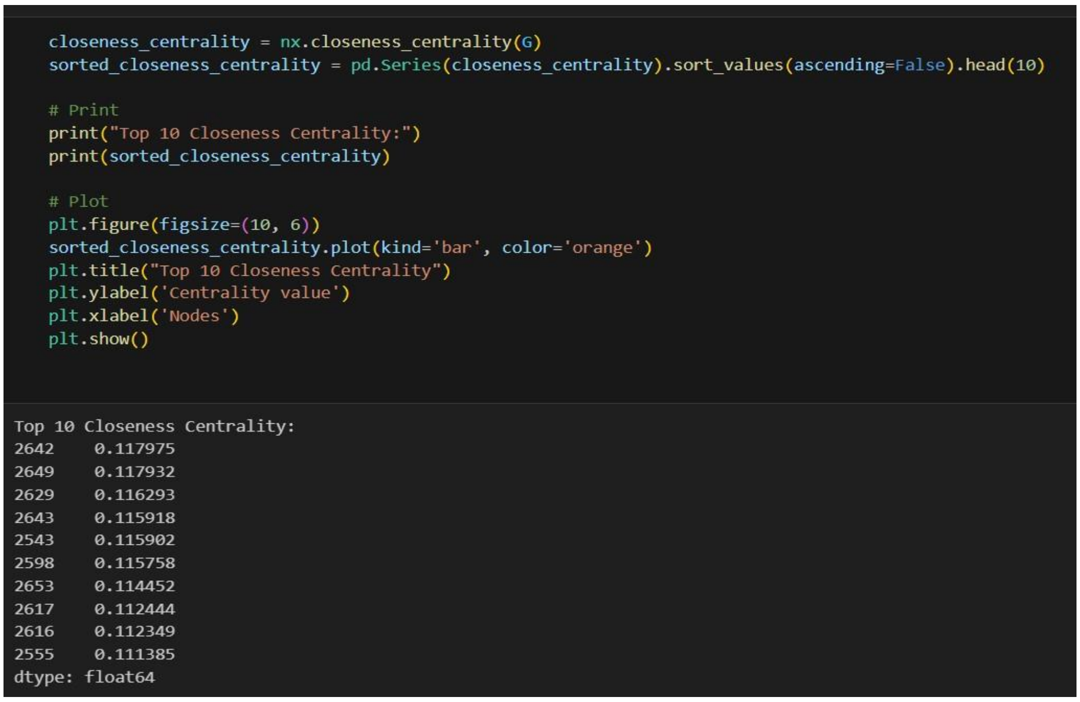
```
pagerank_centrality = nx.pagerank(G)
pagerank_df = pd.Series(pagerank_centrality)

# Plotting
plt.figure(figsize=(10, 6))
sns.kdeplot(pagerank_df, fill=True)
plt.title('Density Plot of PageRank Centrality')
plt.xlabel('PageRank Centrality')
plt.ylabel('Density')
plt.show()
```



Inferences:

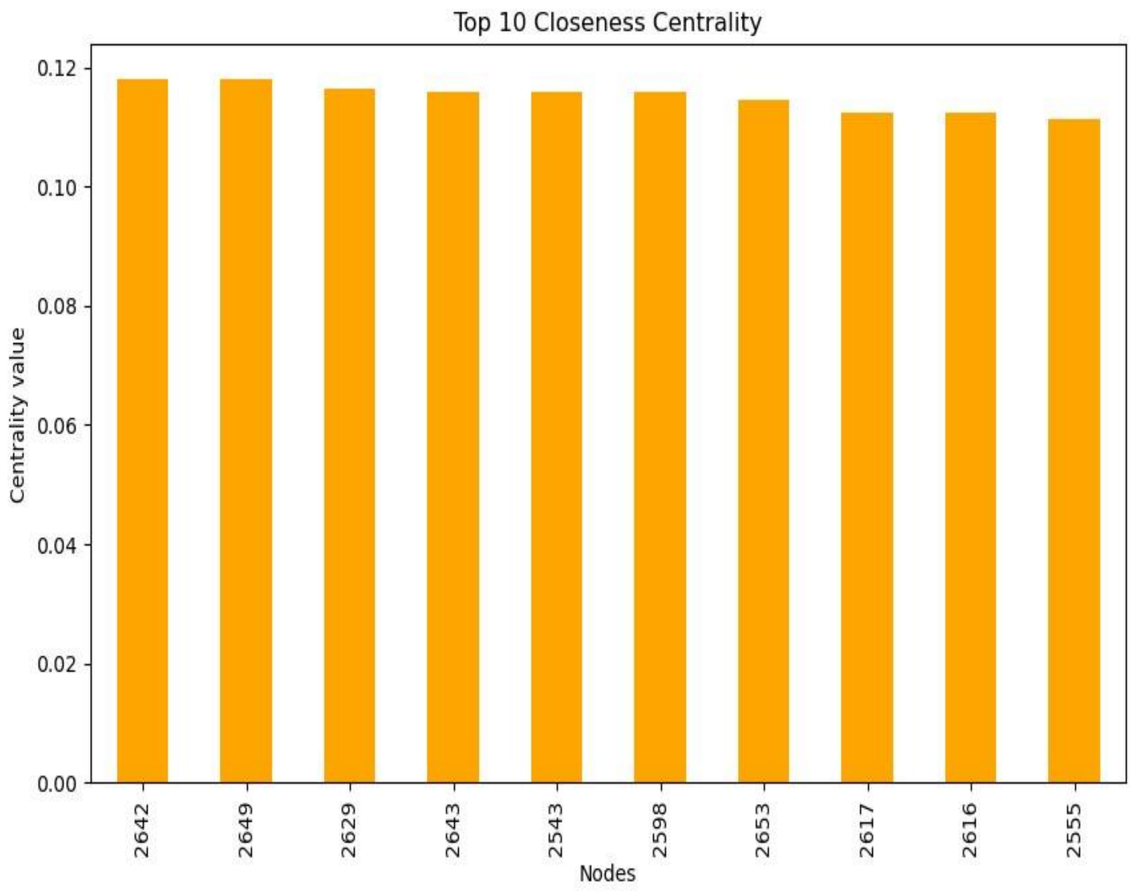
- **Many users have a small number of friends:** The density appears to be highest at the lower end of the x-axis, suggesting that a large portion of Facebook users have a low number of connections (friends).
- **There's a smaller number of users with a large number of friends:** The tail on the right side of the distribution suggests a smaller number of users with a high degree centrality. These are likely the most popular users on Facebook, with a very large number of friends.

Top 10 Closeness Centralities:

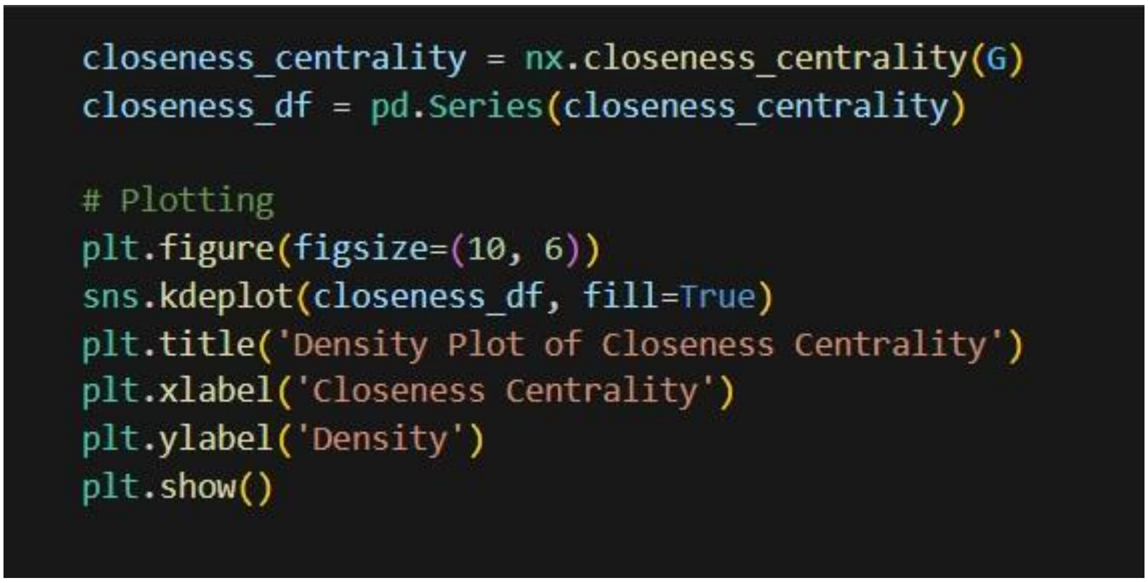


```
closeness_centrality = nx.closeness_centrality(G)
sorted_closeness_centrality = pd.Series(closeness_centrality).sort_values(ascending=False).head(10)

# Print
print("Top 10 Closeness Centrality:")
print(sorted_closeness_centrality)

# Plot
plt.figure(figsize=(10, 6))
sorted_closeness_centrality.plot(kind='bar', color='orange')
plt.title("Top 10 Closeness Centrality")
plt.ylabel('Centrality value')
plt.xlabel('Nodes')
plt.show()
```

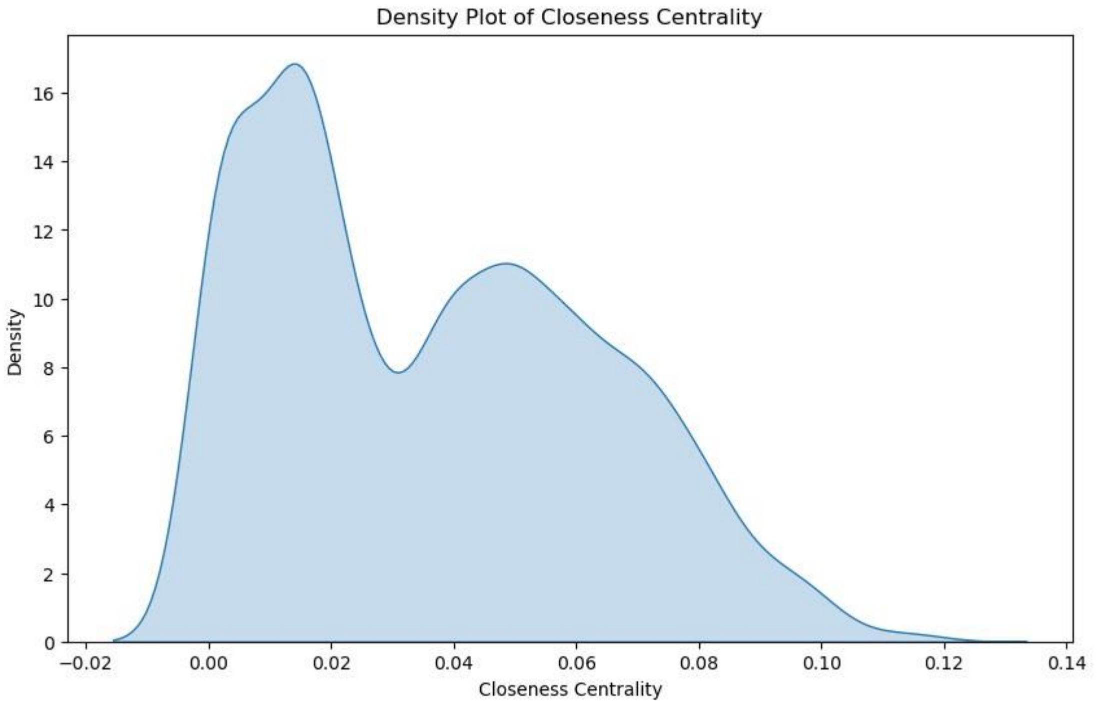
```
Top 10 Closeness Centrality:
2642    0.117975
2649    0.117932
2629    0.116293
2643    0.115918
2543    0.115902
2598    0.115758
2653    0.114452
2617    0.112444
2616    0.112349
2555    0.111385
dtype: float64
```



Density Plot:



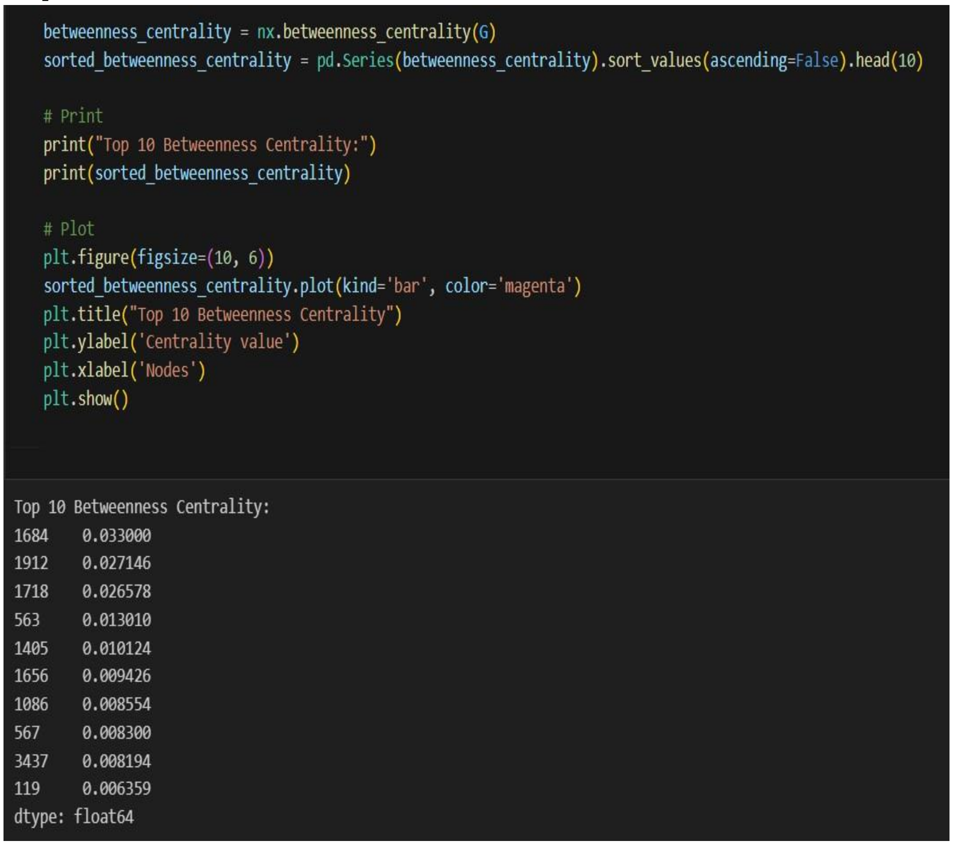
```
closeness_centrality = nx.closeness_centrality(G)
closeness_df = pd.Series(closeness_centrality)

# Plotting
plt.figure(figsize=(10, 6))
sns.kdeplot(closeness_df, fill=True)
plt.title('Density Plot of Closeness Centrality')
plt.xlabel('Closeness Centrality')
plt.ylabel('Density')
plt.show()
```



Inferences:

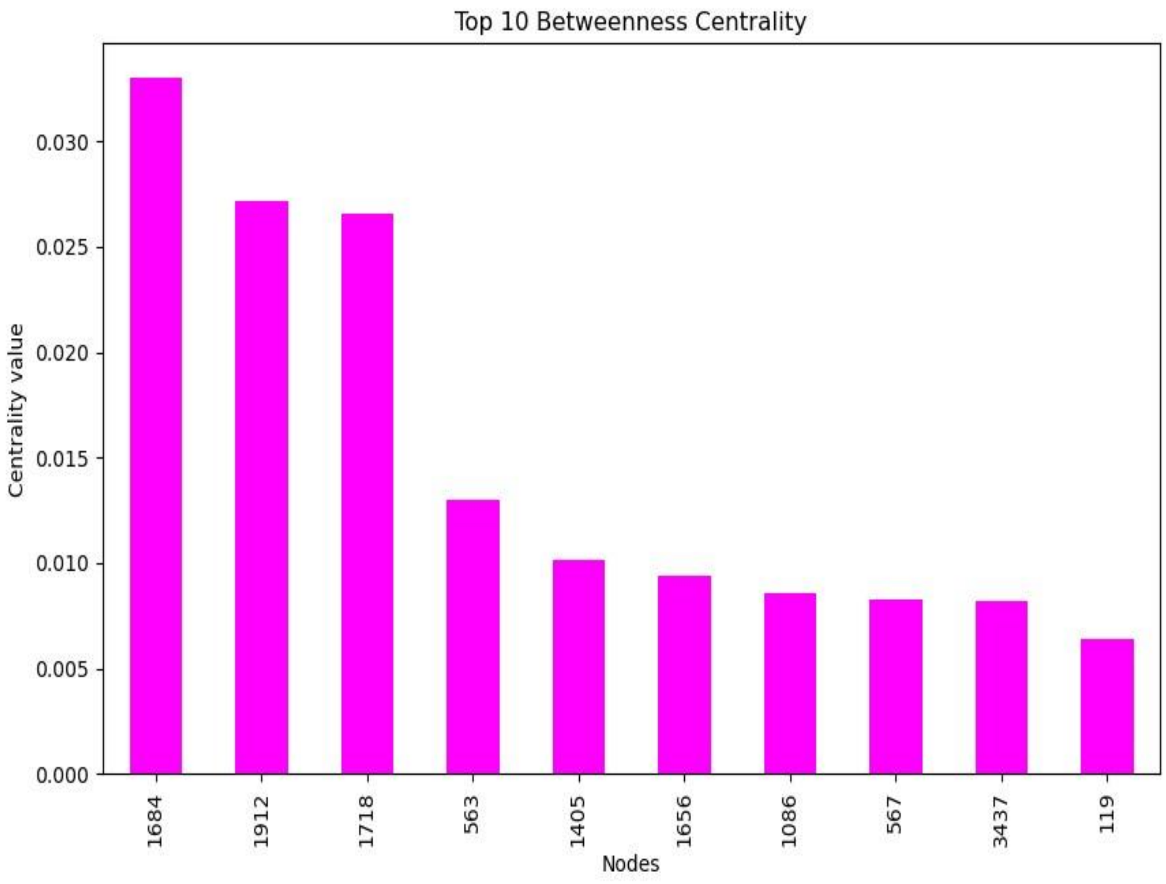
Overall, the density plot of closeness centrality provides valuable insights into how close users are to each other on the Facebook network. It reveals a spectrum, with many users having moderate closeness centrality, and smaller groups at either extreme. This knowledge can be helpful for understanding how information and influence flow on the platform.

Top 10 Betweenness Centralities:

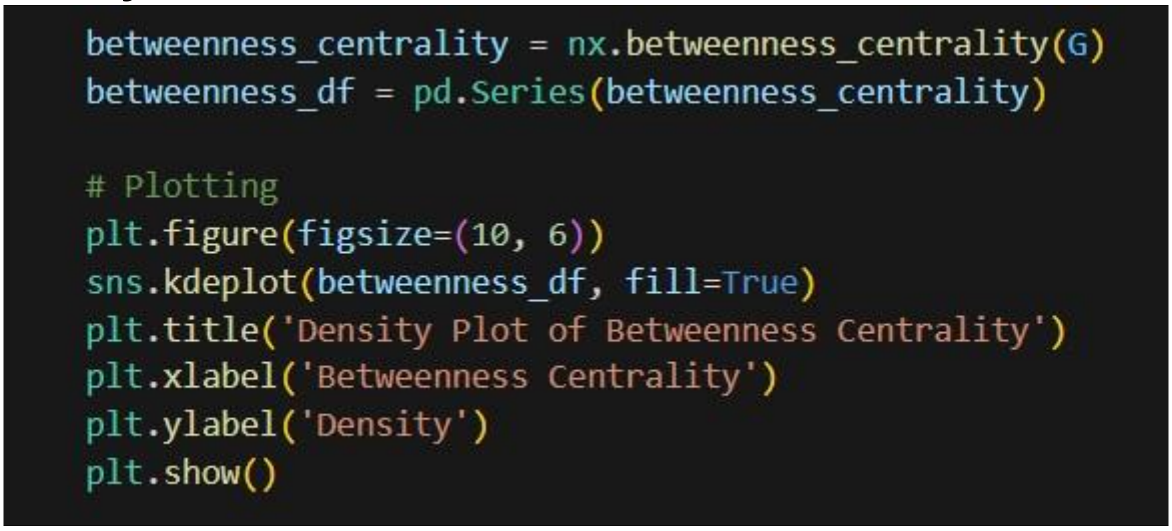


```
betweenness_centrality = nx.betweenness_centrality(G)
sorted_betweenness_centrality = pd.Series(betweenness_centrality).sort_values(ascending=False).head(10)

# Print
print("Top 10 Betweenness Centrality:")
print(sorted_betweenness_centrality)

# Plot
plt.figure(figsize=(10, 6))
sorted_betweenness_centrality.plot(kind='bar', color='magenta')
plt.title("Top 10 Betweenness Centrality")
plt.ylabel('Centrality value')
plt.xlabel('Nodes')
plt.show()
```

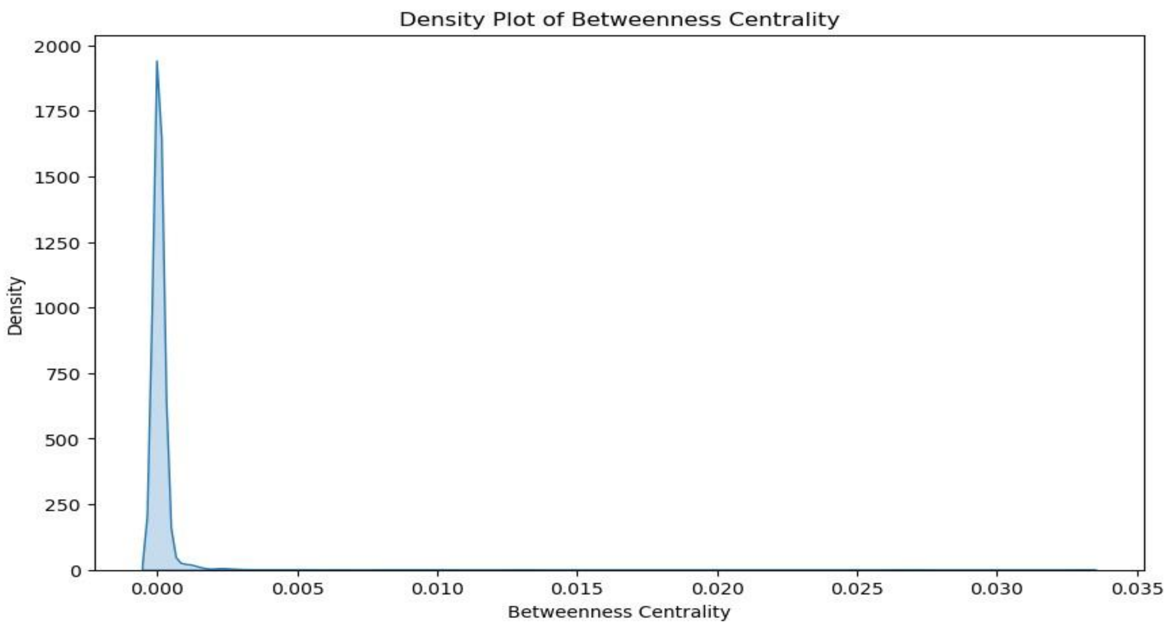
```
Top 10 Betweenness Centrality:
1684    0.033000
1912    0.027146
1718    0.026578
563     0.013010
1405    0.010124
1656    0.009426
1086    0.008554
567     0.008300
3437    0.008194
119     0.006359
dtype: float64
```



Density Plot:



```
betweenness_centrality = nx.betweenness_centrality(G)
betweenness_df = pd.Series(betweenness_centrality)

# Plotting
plt.figure(figsize=(10, 6))
sns.kdeplot(betweenness_df, fill=True)
plt.title('Density Plot of Betweenness Centrality')
plt.xlabel('Betweenness Centrality')
plt.ylabel('Density')
plt.show()
```

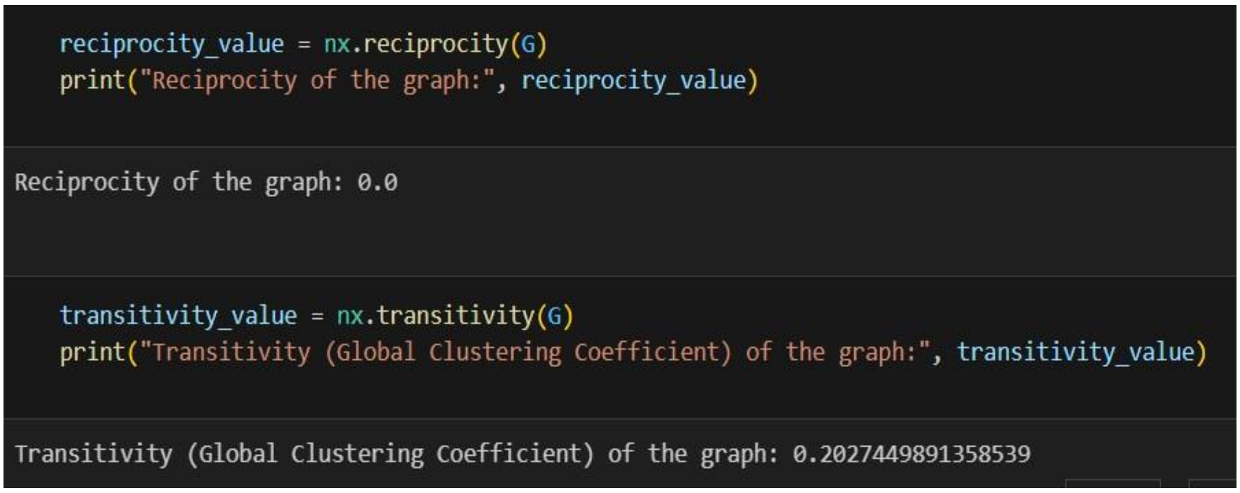


Inferences:

The density plot likely shows a peak in the middle with tails on either side. The central peak represents users with moderate betweenness centrality. These users act as bridges between different communities or groups within Facebook. Information or trends might flow through them as they connect various parts of the network.

The tail on the left side (low betweenness centrality) likely represents users who are on the outskirts of the network.

The tail on the right side (high betweenness centrality) represents a smaller group of influential users who lie on the shortest paths between many other users.



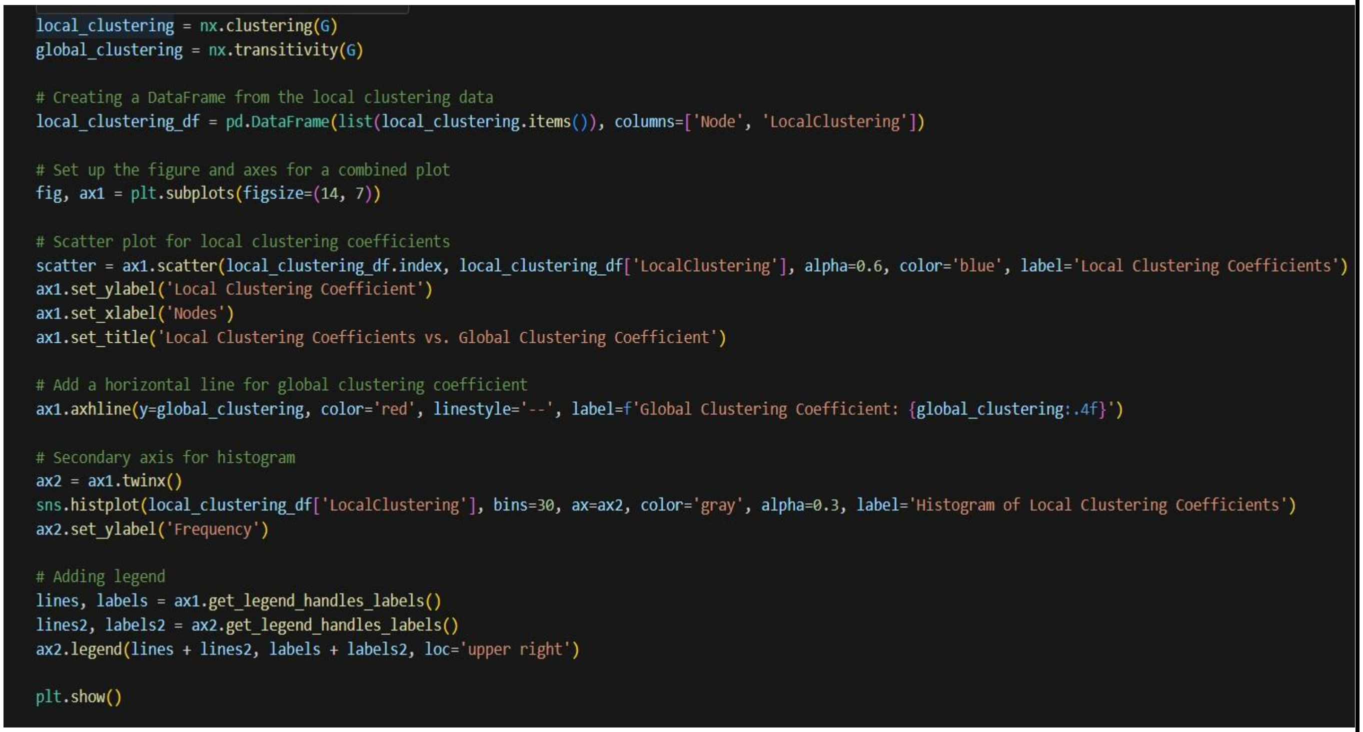
```python
reciprocity_value = nx.reciprocity(G)
print("Reciprocity of the graph:", reciprocity_value)
```

```
Reciprocity of the graph: 0.0
```

```python
transitivity_value = nx.transitivity(G)
print("Transitivity (Global Clustering Coefficient) of the graph:", transitivity_value)
```

```
Transitivity (Global Clustering Coefficient) of the graph: 0.2027449891358539
```

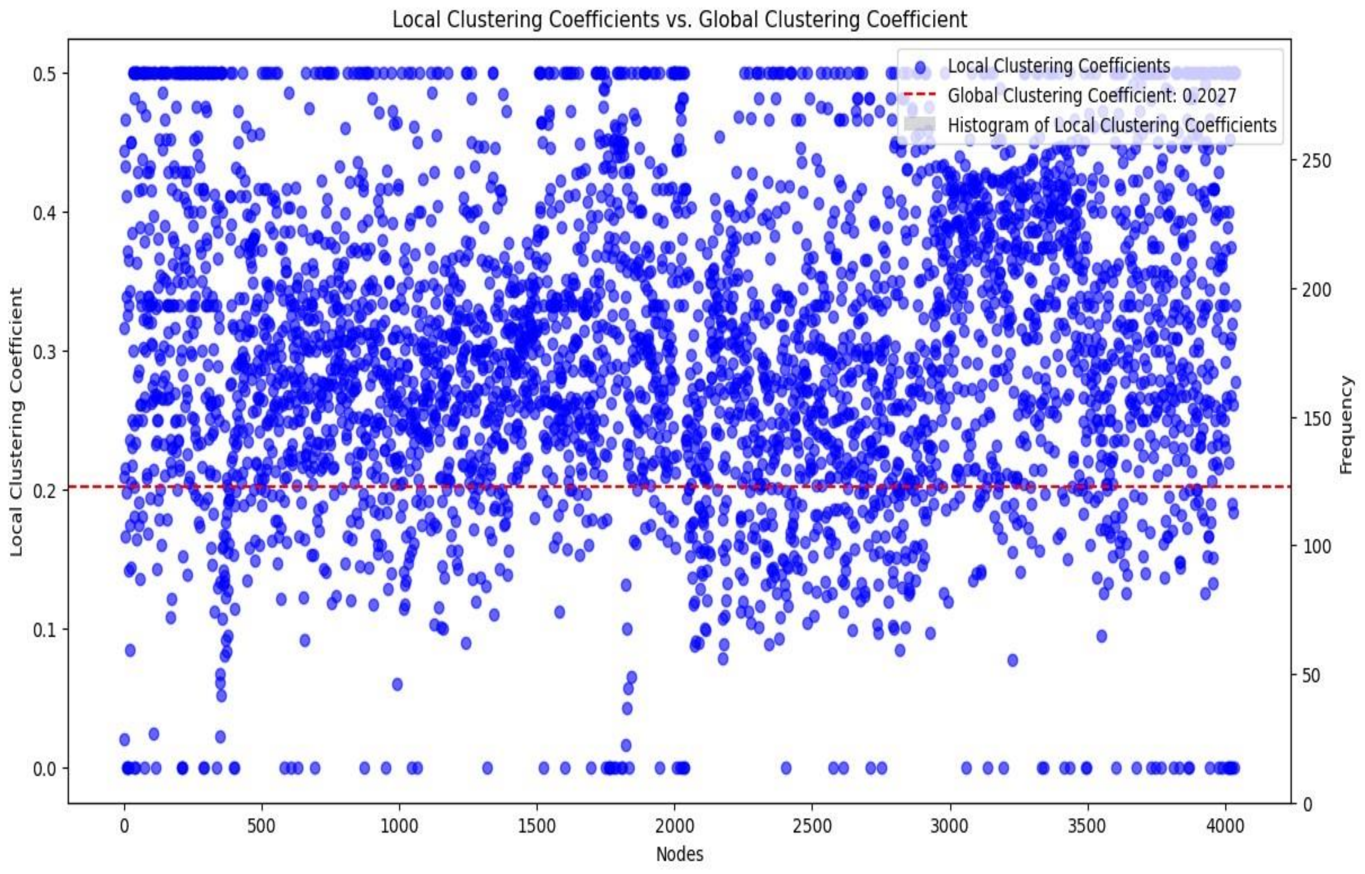
Local Clustering & Global Clustering:



```python
local_clustering = nx.clustering(G)
global_clustering = nx.transitivity(G)

# Creating a DataFrame from the local clustering data
local_clustering_df = pd.DataFrame(list(local_clustering.items()), columns=['Node', 'LocalClustering'])

# Set up the figure and axes for a combined plot
fig, ax1 = plt.subplots(figsize=(14, 7))

# Scatter plot for local clustering coefficients
scatter = ax1.scatter(local_clustering_df.index, local_clustering_df['LocalClustering'], alpha=0.6, color='blue', label='Local Clustering Coefficients')
ax1.set_ylabel('Local Clustering Coefficient')
ax1.set_xlabel('Nodes')
ax1.set_title('Local Clustering Coefficients vs. Global Clustering Coefficient')

# Add a horizontal line for global clustering coefficient
ax1.axhline(y=global_clustering, color='red', linestyle='--', label=f'Global Clustering Coefficient: {global_clustering:.4f}')

# Secondary axis for histogram
ax2 = ax1.twinx()
sns.histplot(local_clustering_df['LocalClustering'], bins=30, ax=ax2, color='gray', alpha=0.3, label='Histogram of Local Clustering Coefficients')
ax2.set_ylabel('Frequency')

# Adding legend
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper right')

plt.show()
```

- Local clustering might be used to identify groups of friends who frequently interact with each other, share similar posts, or belong to the same groups.

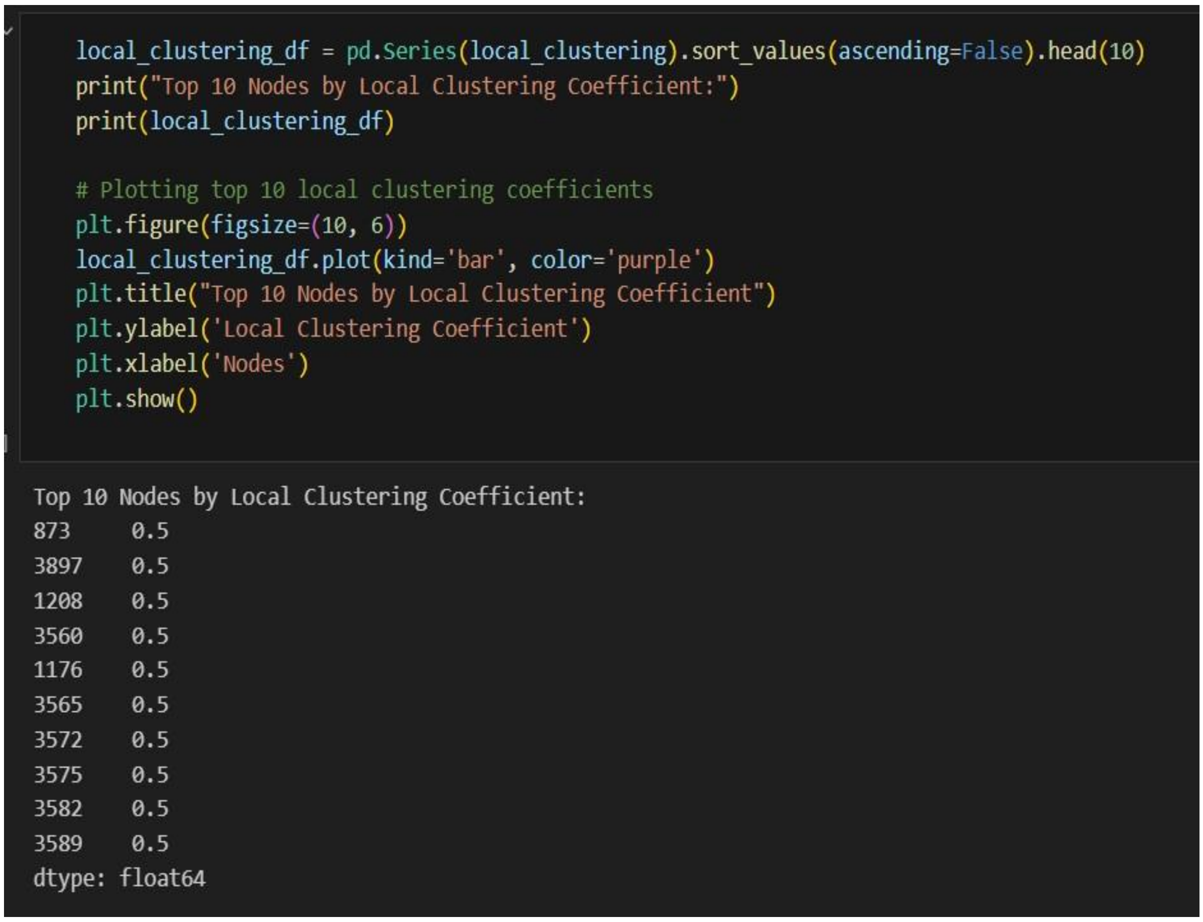
- Global clustering could be used to identify communities based on interests (e.g., sports fans, music lovers), demographics (e.g., parents of young children, professionals in a specific industry), or online behavior (e.g., news sharers, content creators).



Inferences:

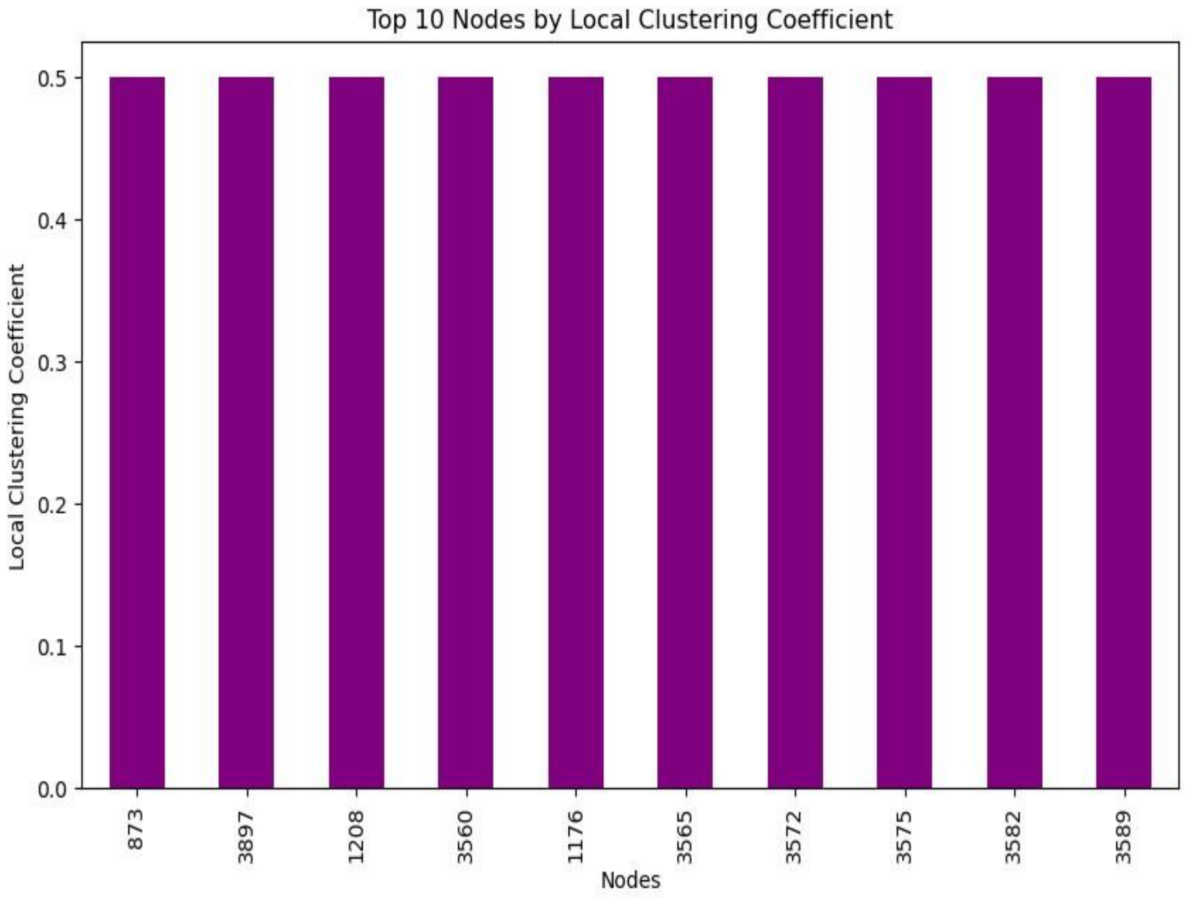
- **Generally higher local clustering coefficient:** Most of the data points appear to be above the diagonal line, suggesting that for most users, their friends (local connections) also tend to be friends with each other (high local clustering coefficient). This indicates that the network is made up of many small communities or groups where users tend to be well-connected to each other.

- **Variation in global clustering coefficient:** There seems to be more spread in the data points along the y-axis (global clustering coefficient) compared to the x-axis (local clustering coefficient). This suggests that while most users are in locally clustered communities, the network itself might not be globally well-clustered. In other words, these local communities may not be strongly interconnected with each other.



```
local_clustering_df = pd.Series(local_clustering).sort_values(ascending=False).head(10)
print("Top 10 Nodes by Local Clustering Coefficient:")
print(local_clustering_df)

# Plotting top 10 local clustering coefficients
plt.figure(figsize=(10, 6))
local_clustering_df.plot(kind='bar', color='purple')
plt.title("Top 10 Nodes by Local Clustering Coefficient")
plt.ylabel('Local Clustering Coefficient')
plt.xlabel('Nodes')
plt.show()
```

```
Top 10 Nodes by Local Clustering Coefficient:
873     0.5
3897    0.5
1208    0.5
3560    0.5
1176    0.5
3565    0.5
3572    0.5
3575    0.5
3582    0.5
3589    0.5
dtype: float64
```

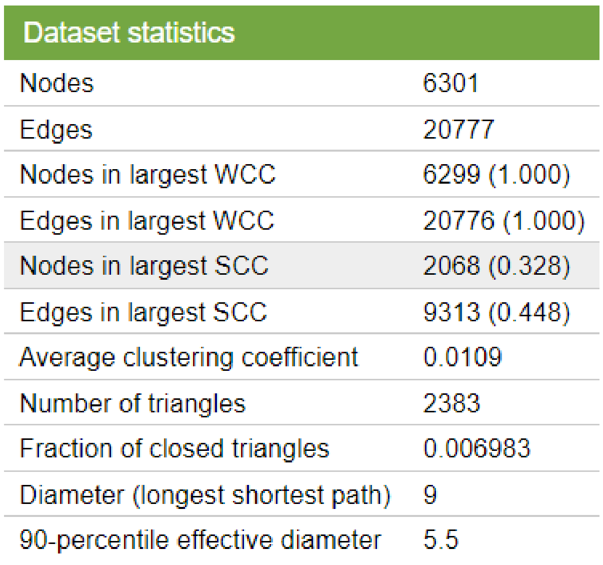


Dataset 2: Gnutella peer-to-peer network

A sequence of snapshots of the Gnutella peer-to-peer file-sharing network from August 2002. There are a total of 9 snapshots of the Gnutella network collected in August 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts.

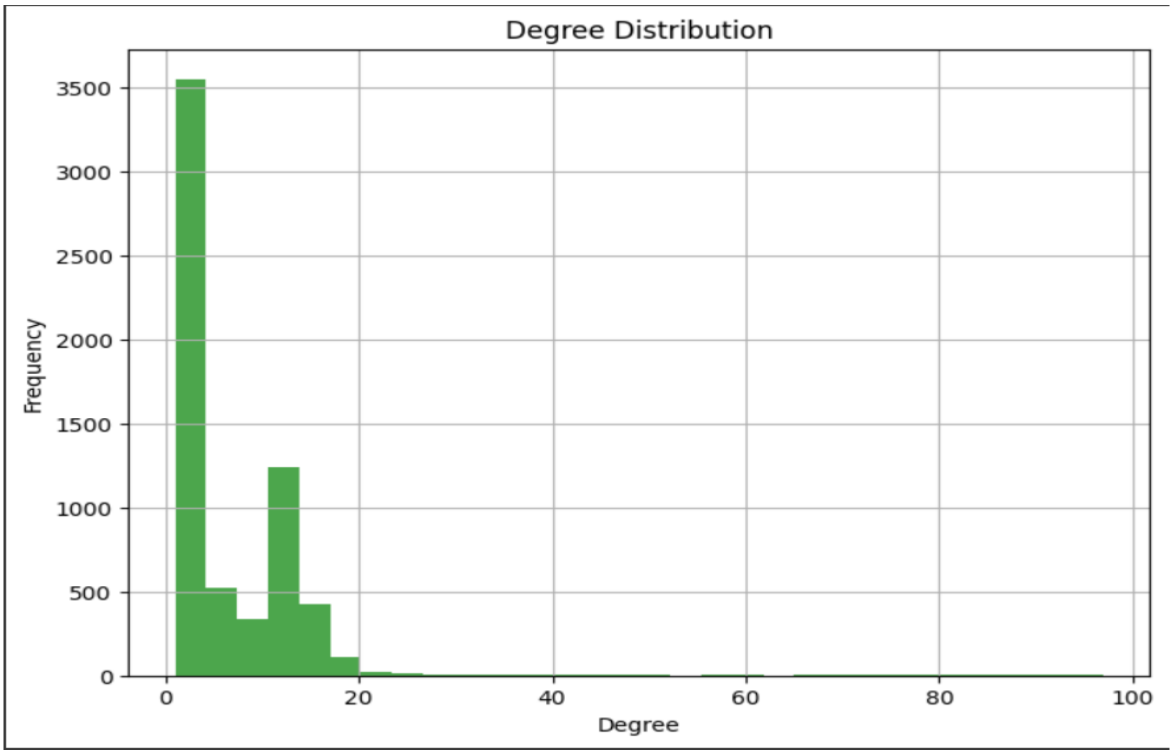
What is Gnutella?

Gnutella was designed to be a file-sharing system based on an unstructured P2P overlay that allows for the decentralized, scalable, reliable, and anonymous sharing of files between participating nodes.



Dataset statistics	
Nodes	6301
Edges	20777
Nodes in largest WCC	6299 (1.000)
Edges in largest WCC	20776 (1.000)
Nodes in largest SCC	2068 (0.328)
Edges in largest SCC	9313 (0.448)
Average clustering coefficient	0.0109
Number of triangles	2383
Fraction of closed triangles	0.006983
Diameter (longest shortest path)	9
90-percentile effective diameter	5.5

Degree of distribution



Inference

Maximum Degree: 97

Minimum Degree: 1

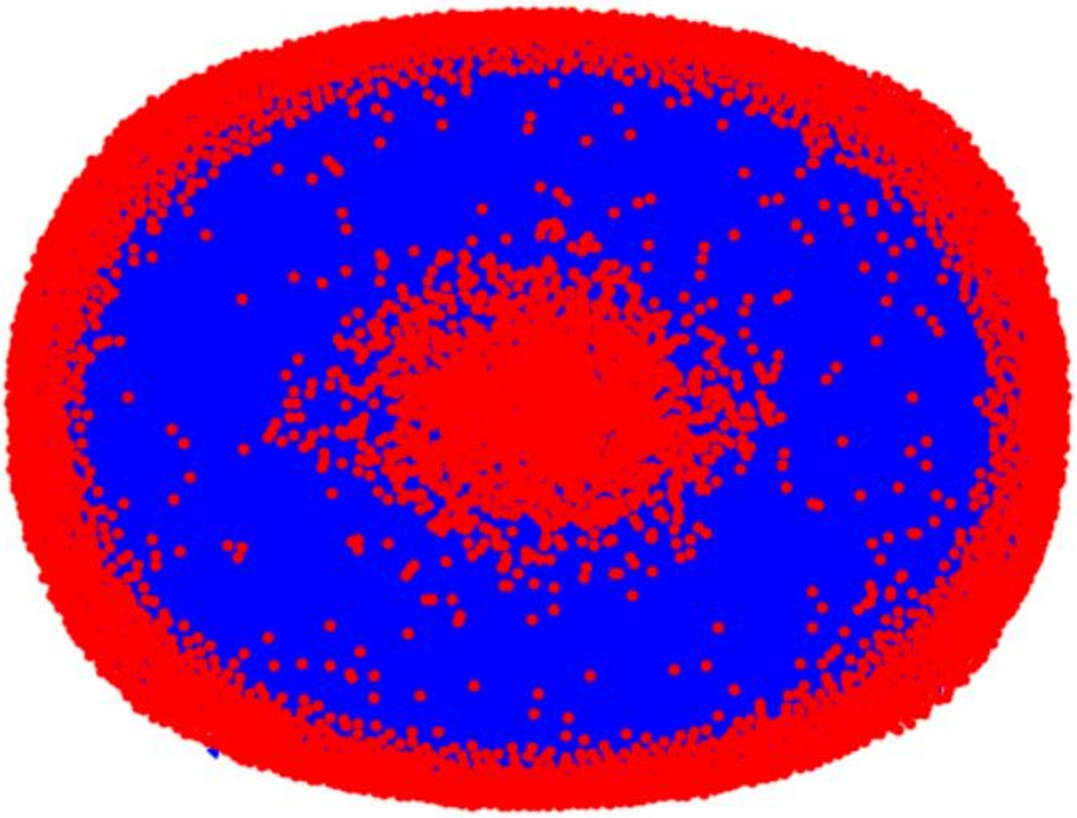
Average Degree: 6.59

Standard Deviation of Degree: 8.54

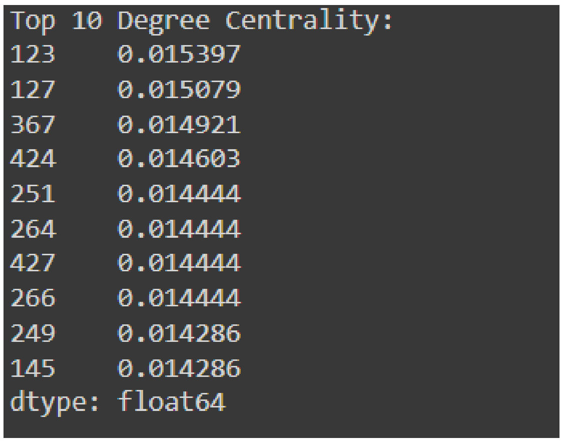
Comparison with Dataset-1

- Both graphs share the same trend of having more low-degree nodes and fewer high-degree nodes.
- The specific values may differ, but the overall shape remains consistent.

Network graph visualization



Top 10 degree of centrality of dataset -2

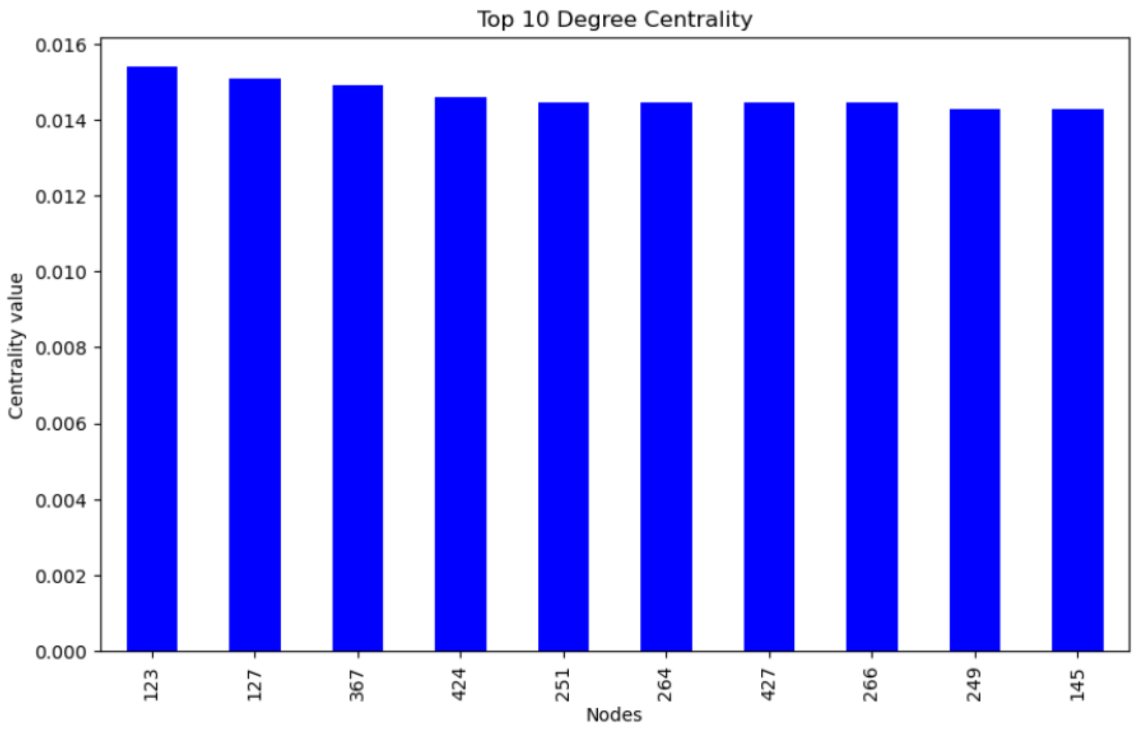


```
Top 10 Degree Centrality:
123    0.015397
127    0.015079
367    0.014921
424    0.014603
251    0.014444
264    0.014444
427    0.014444
266    0.014444
249    0.014286
145    0.014286
dtype: float64
```

1. Nodes with higher degree centrality values play more influential roles in the network.
2. These top 10 nodes likely have significant connections or interactions with other nodes.

Comparison with the Dataset-1

The previous table had higher degree centrality values, while the current table shows lower values.



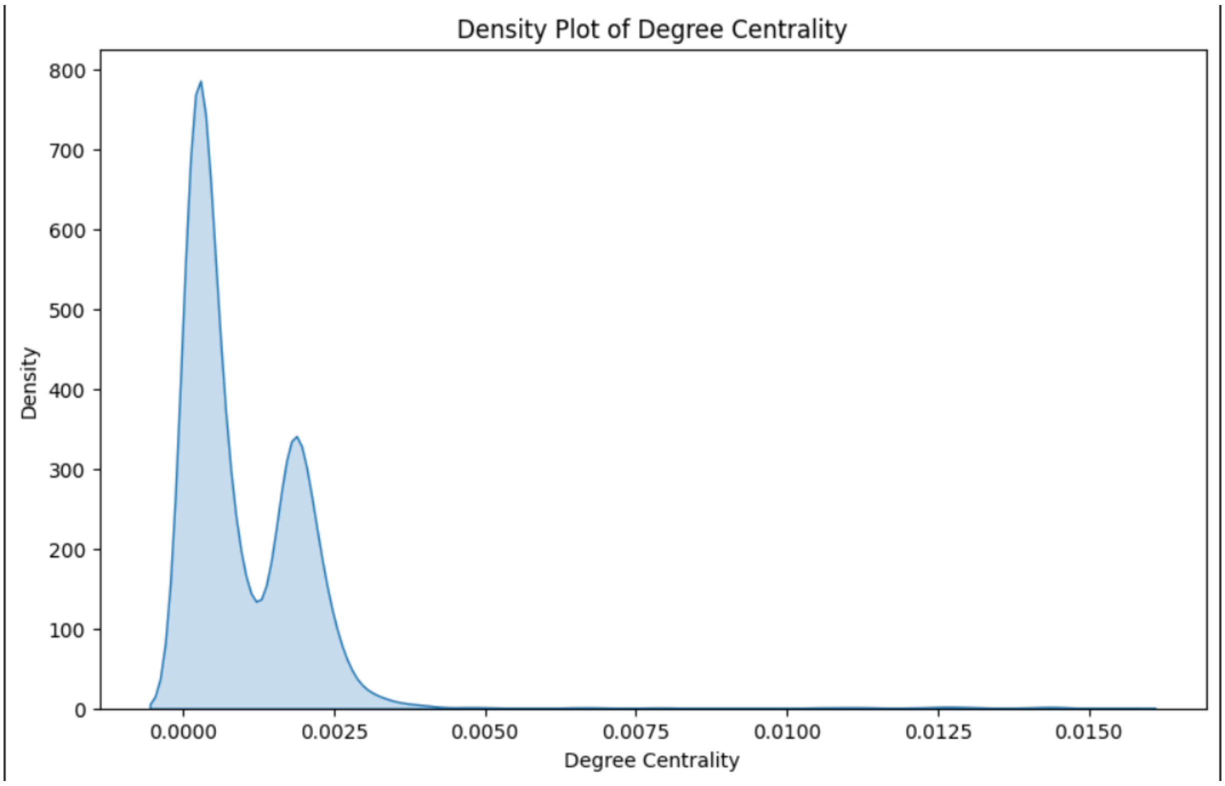
Inference

The graph represents the **top 10 nodes** within a network based on their **degree of centrality**. Here are the key takeaways:

1. **Node 123**: Has a degree centrality of approximately **0.015397**.
2. **Node 127**: Also has a degree centrality of approximately **0.015079**.
3. **Node 367**: Exhibits a degree centrality value of approximately **0.014921**.
4. **Node 424**: Shows a degree centrality of approximately **0.014603**.
5. **Node 251**: Has a degree centrality value of approximately **0.014444**.

These nodes play influential roles in the network due to their connections. The degree centrality values are relatively close, indicating similar importance among these top 10 nodes.

Density plot of degree of centrality

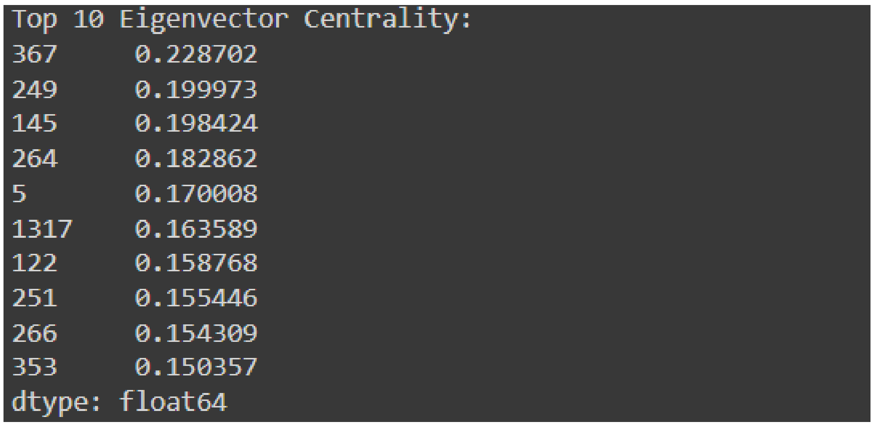


Comparison with dataset-1

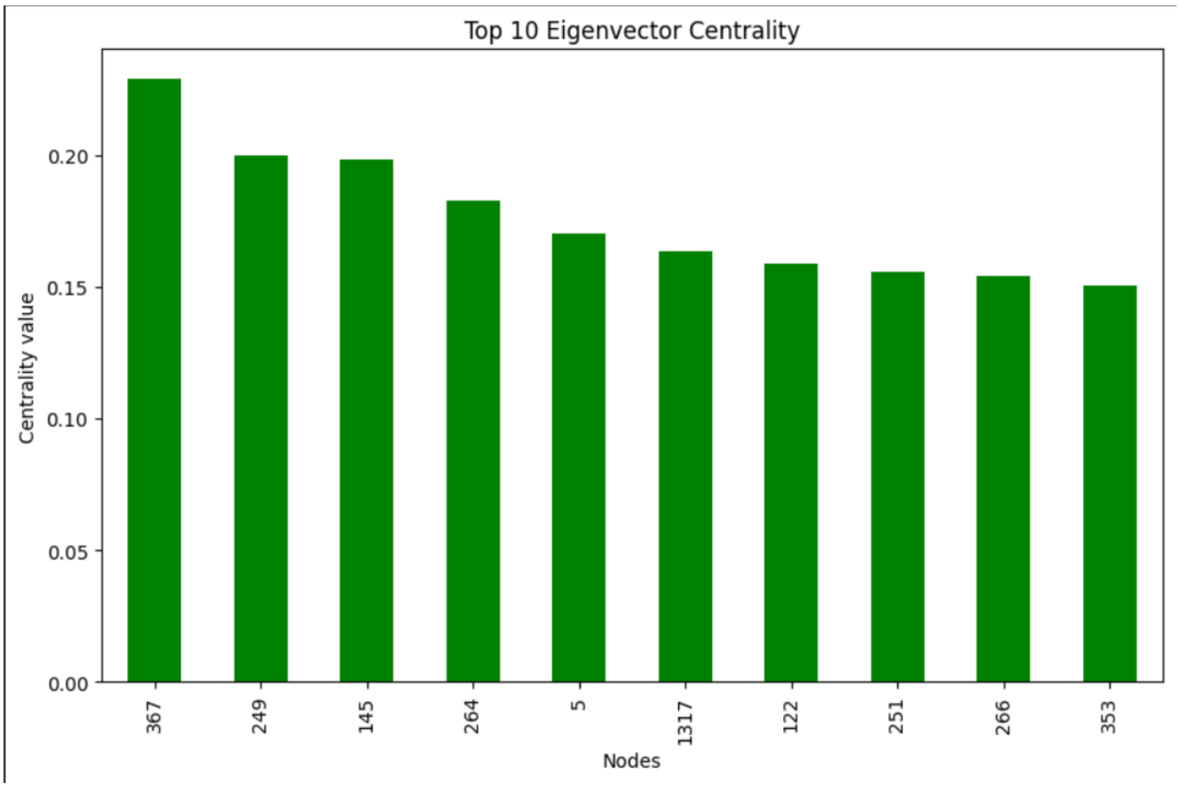
The current distribution is more concentrated around a higher degree of centrality value.

Top 10 Eigen Vector Centrality

Dataset-2



```
Top 10 Eigenvector Centrality:
367     0.228702
249     0.199973
145     0.198424
264     0.182862
5       0.170008
1317    0.163589
122     0.158768
251     0.155446
266     0.154309
353     0.150357
dtype: float64
```



Inference

The graph of eigenvector centrality reveals the influence of nodes within a network. Here are the key insights:

1. Node 367: Has the highest eigenvector centrality value, approximately 0.20. This node is most central within the network according to this measure.
2. Other nodes (249, 145, 264, 5, 1317, 122, 251, 256, and 353) also exhibit moderate to high eigenvector centrality values, ranging from approximately 0.15 to 0.20.

3. These nodes likely play crucial roles in information flow and influence propagation within the network.

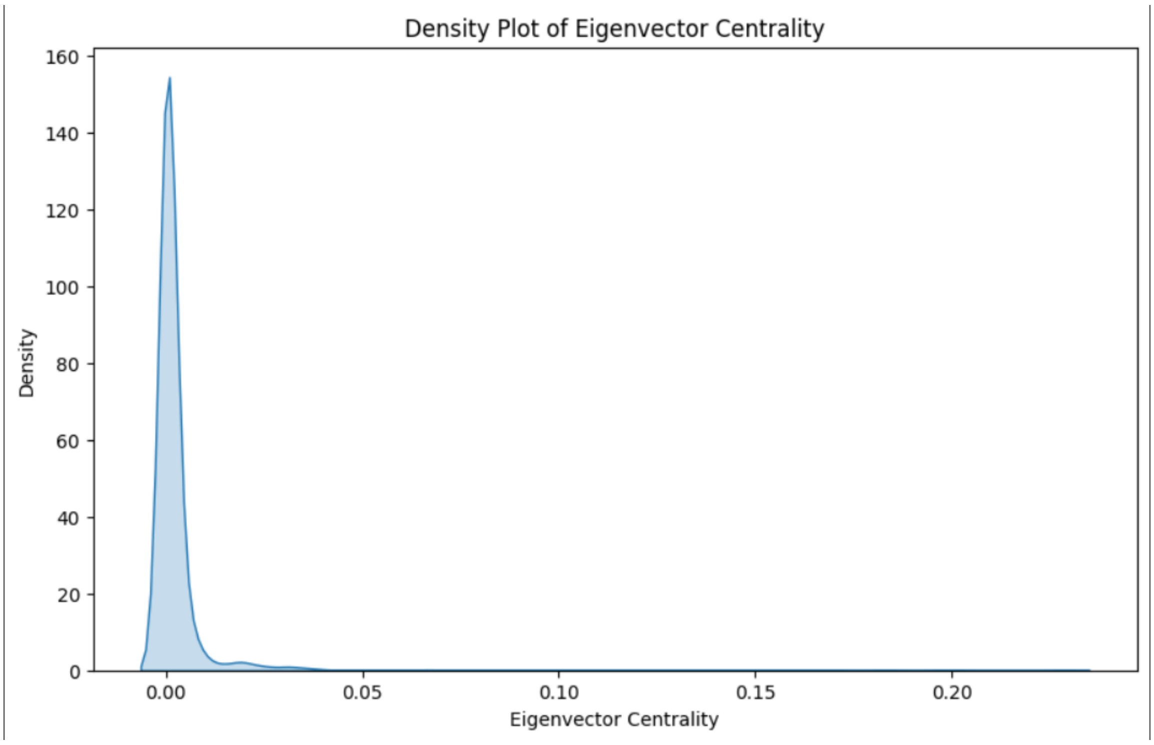
Remember that eigenvector centrality emphasizes both the node’s direct connections and the centrality of its neighbors. These influential nodes can significantly impact the network dynamics.

Comparison with dataset-1

- The previous table had higher eigenvector centrality values, while the the table shows lower values.
- Both tables highlight influential nodes within their respective networks.

Density plot of Eigenvector centrality

Plot for dataset -2



Inference

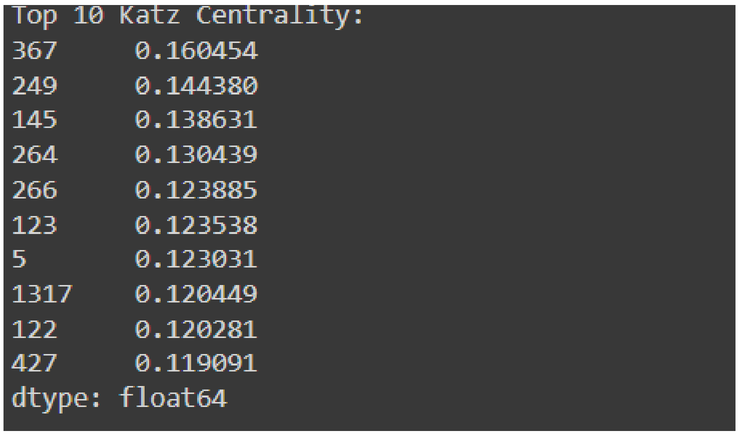
The graph of eigenvector centrality reveals the influence of nodes within a network. Here are the key insights:

1. **Node 367**: Has the highest eigenvector centrality value of approximately **0.228702**. This node is the most central within the network according to this measure.
2. Other nodes (249, 145, 264, 5, 1317, 122, 251, 256, and 353) also exhibit moderate to high eigenvector centrality values, ranging from approximately **0.15** to **0.20**.
3. These nodes likely play crucial roles in information flow and influence propagation within the network.

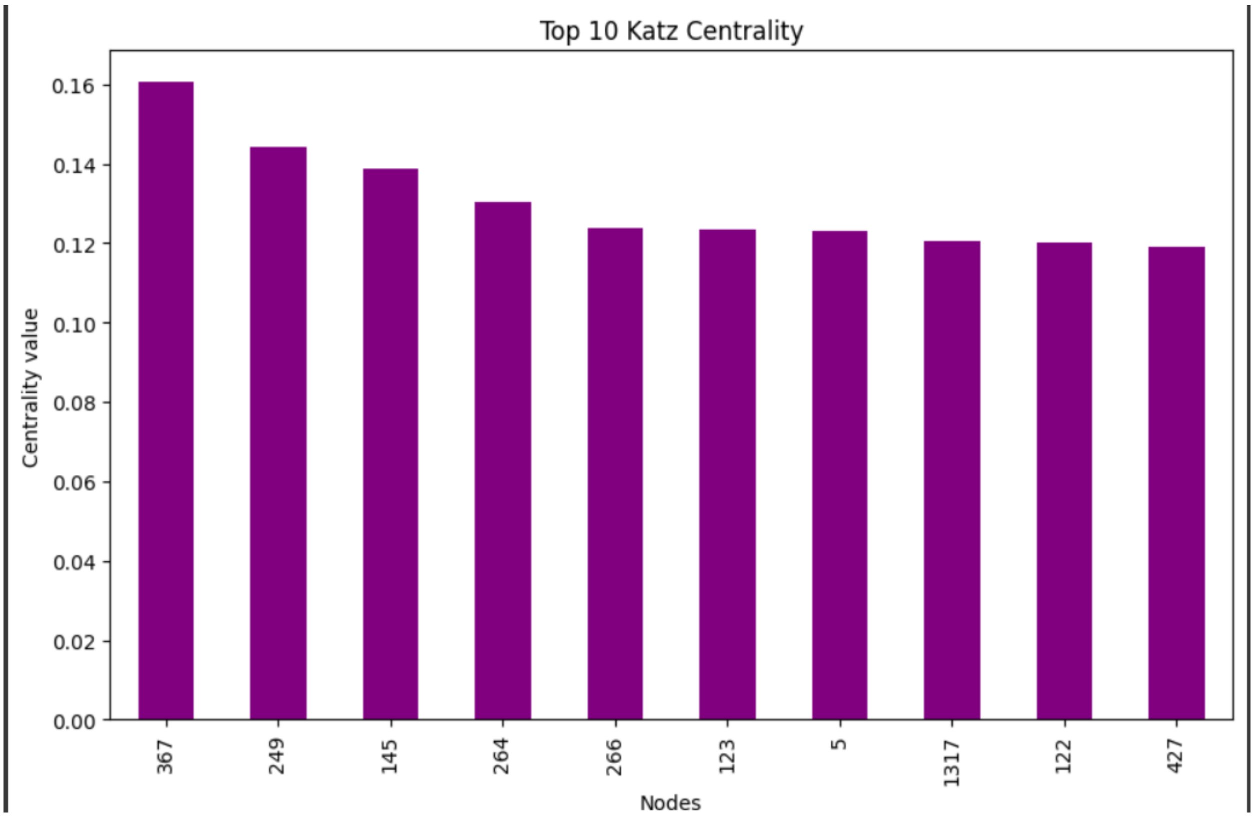
Remember that eigenvector centrality emphasizes both the node's direct connections and the centrality of its neighbors. These influential nodes can significantly impact the network dynamics.

Top 10 Katz Centrality

Dataset-2



```
Top 10 Katz Centrality:
367     0.160454
249     0.144380
145     0.138631
264     0.130439
266     0.123885
123     0.123538
5       0.123031
1317    0.120449
122     0.120281
427     0.119091
dtype: float64
```

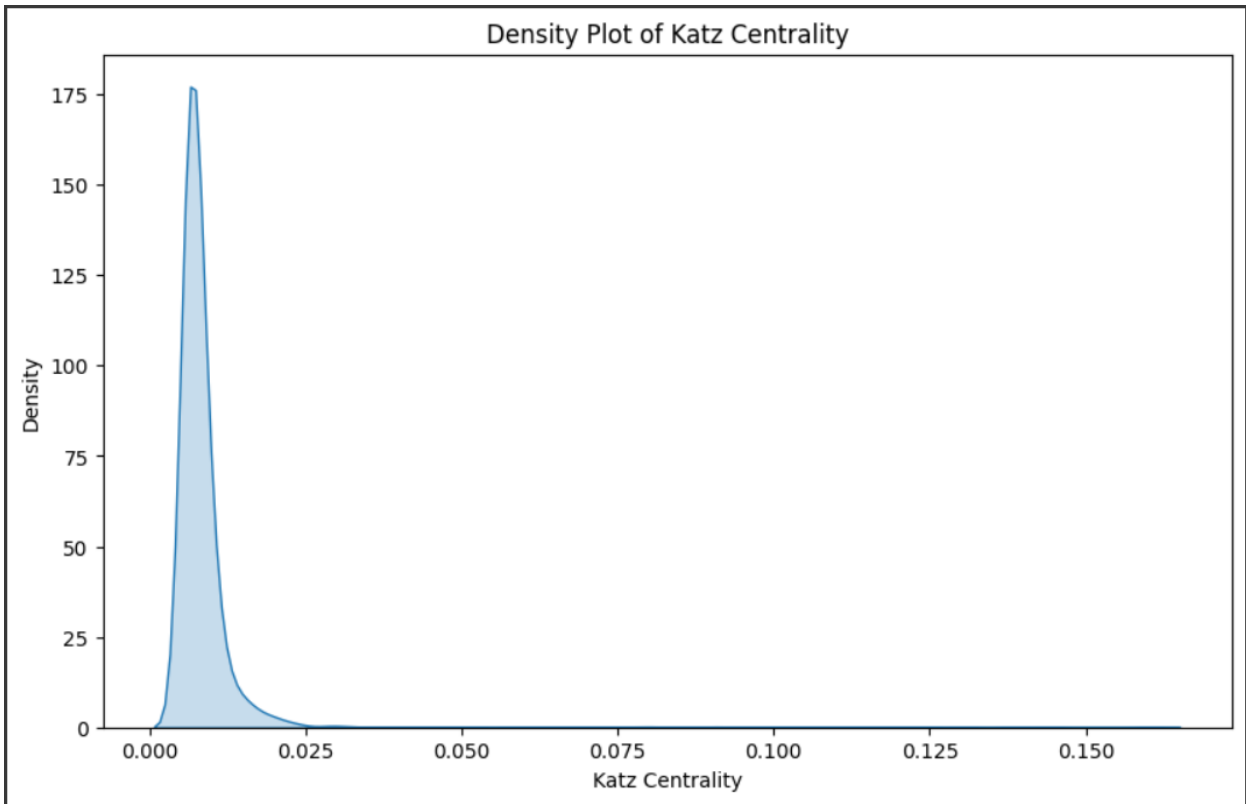


Comparison with dataset-1

1. The previous table had higher Katz centrality values, while the current table shows lower values.
2. Nodes with higher Katz centrality are more influential within the network.
3. Both tables highlight influential nodes based on their connectivity and influence propagation.

Remember that Katz's centrality considers both direct connections and the influence of neighbors, making it a valuable measure for understanding node importance.

Density plot



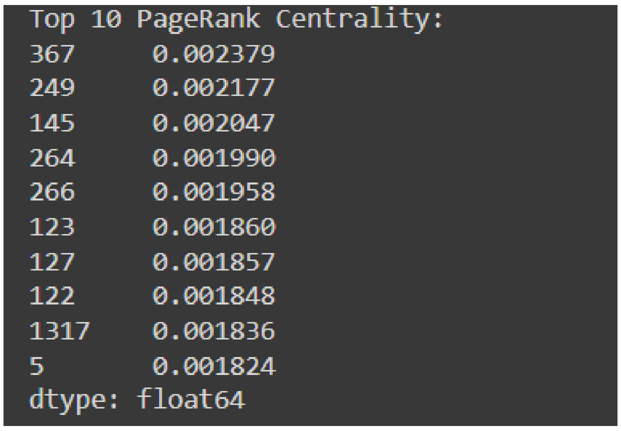
Inference for the output

The density plot of Katz's Centrality provides insights into the distribution of centrality values within a network. Here are the key observations:

- The x-axis represents Katz Centrality values, ranging from 0 to approximately 0.15.
- The y-axis represents density, indicating how many nodes fall within a specific Katz Centrality intervals.
- The plot shows a peak in density near a Katz Centrality of 0, suggesting that many nodes have low centrality.
- As Katz's Centrality increases, the density sharply decreases, indicating fewer nodes with higher centrality values.

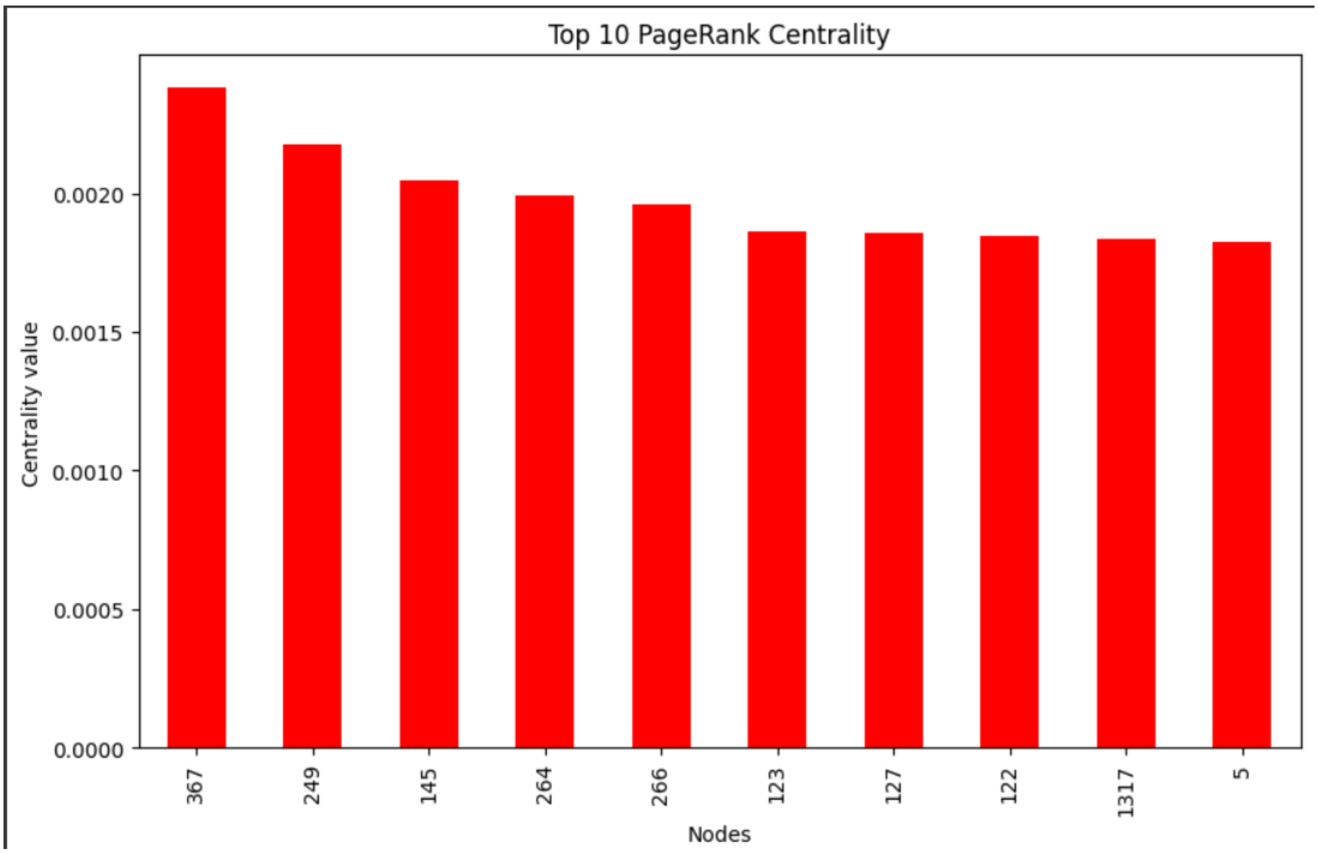
In summary, this density plot highlights the prevalence of low centrality nodes and the rapid decline in density as centrality values rise. It provides valuable information about the distribution of influence within the network.

Top 10 PageRank centrality



```
Top 10 PageRank Centrality:
367     0.002379
249     0.002177
145     0.002047
264     0.001990
266     0.001958
123     0.001860
127     0.001857
122     0.001848
1317    0.001836
5       0.001824
dtype: float64
```

For Dataset-2

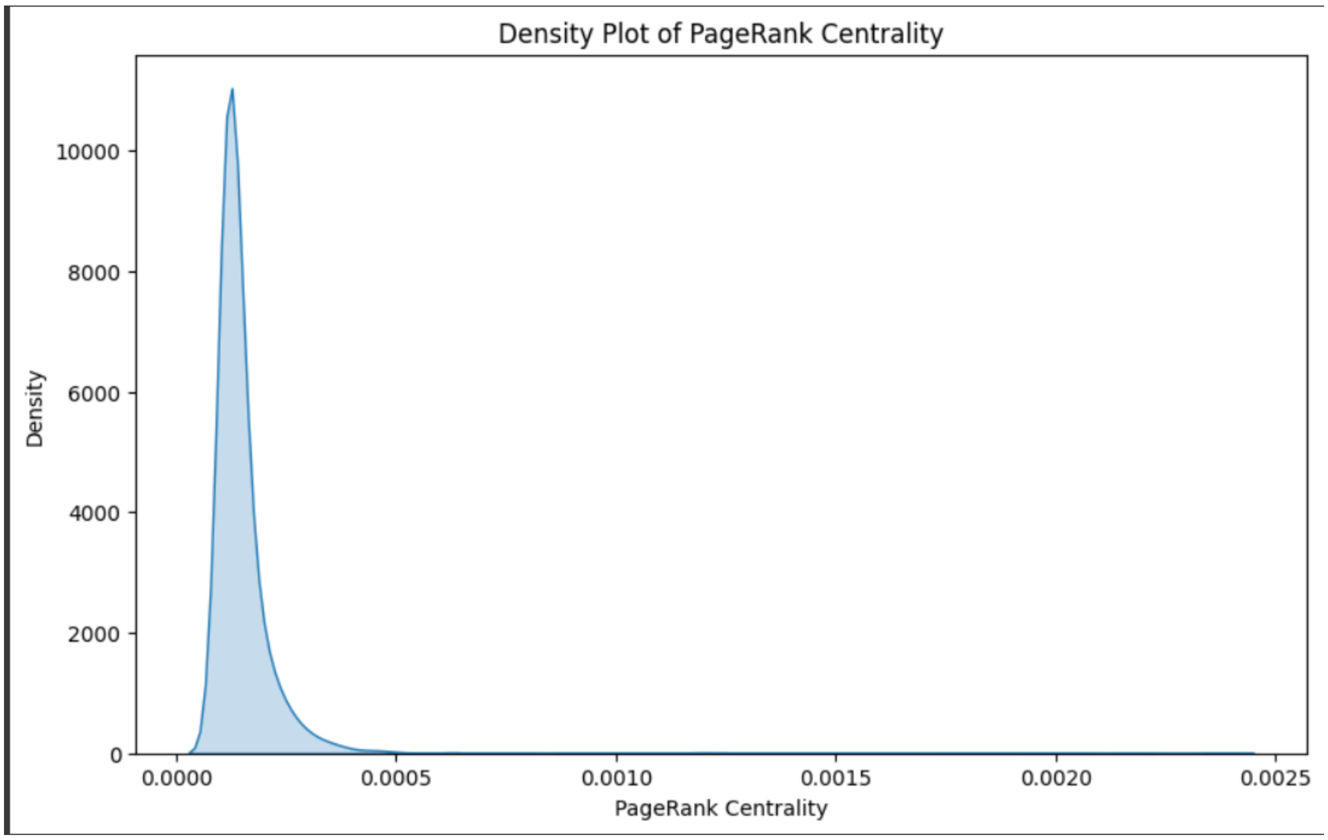


Comparison with dataset-1

1. The PageRank Centrality values in the current network are significantly lower than those in the previous network.
2. Nodes with higher PageRank Centrality are more influential within their respective networks.
3. The current network seems to have less overall influence compared to the previous one.

PageRank Centrality is a valuable measure for understanding node importance in various contexts, such as web pages, social networks, and citation networks

Density Plot graph



Inference

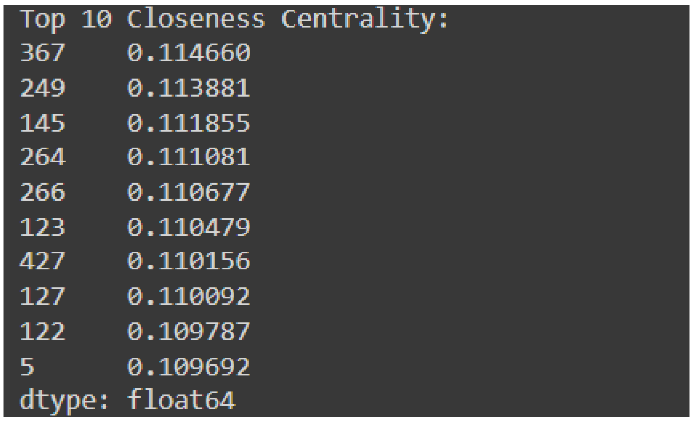
The plot shows a sharp peak in density near a very low PageRank Centrality value (close to zero), suggesting that a large number of nodes have low centrality.

As PageRank Centrality increases, the density rapidly decreases, indicating fewer nodes with higher centrality values.

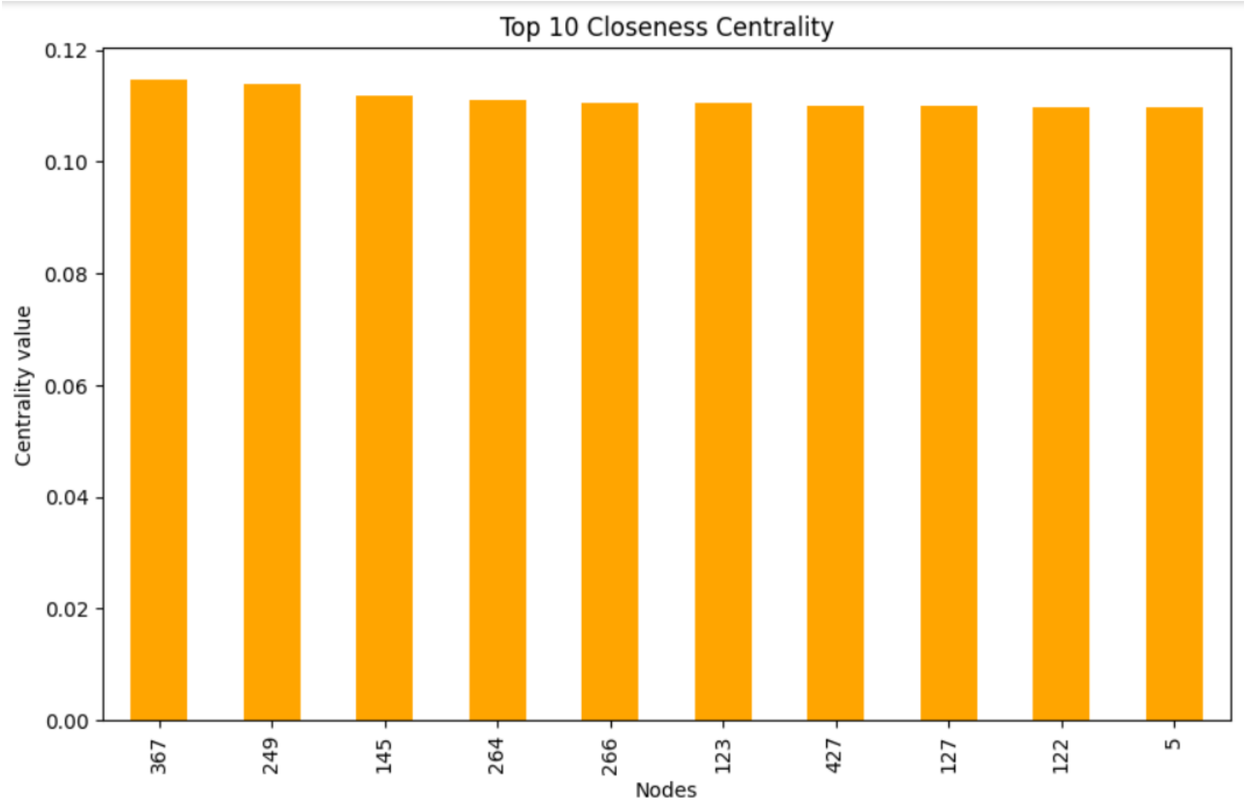
In summary, this density plot highlights the prevalence of low centrality nodes and the scarcity of highly influential nodes within the network**.**

Top 10 Closeness centrality

Dataset-2



```
Top 10 Closeness Centrality:
367    0.114660
249    0.113881
145    0.111855
264    0.111081
266    0.110677
123    0.110479
427    0.110156
127    0.110092
122    0.109787
5      0.109692
dtype: float64
```



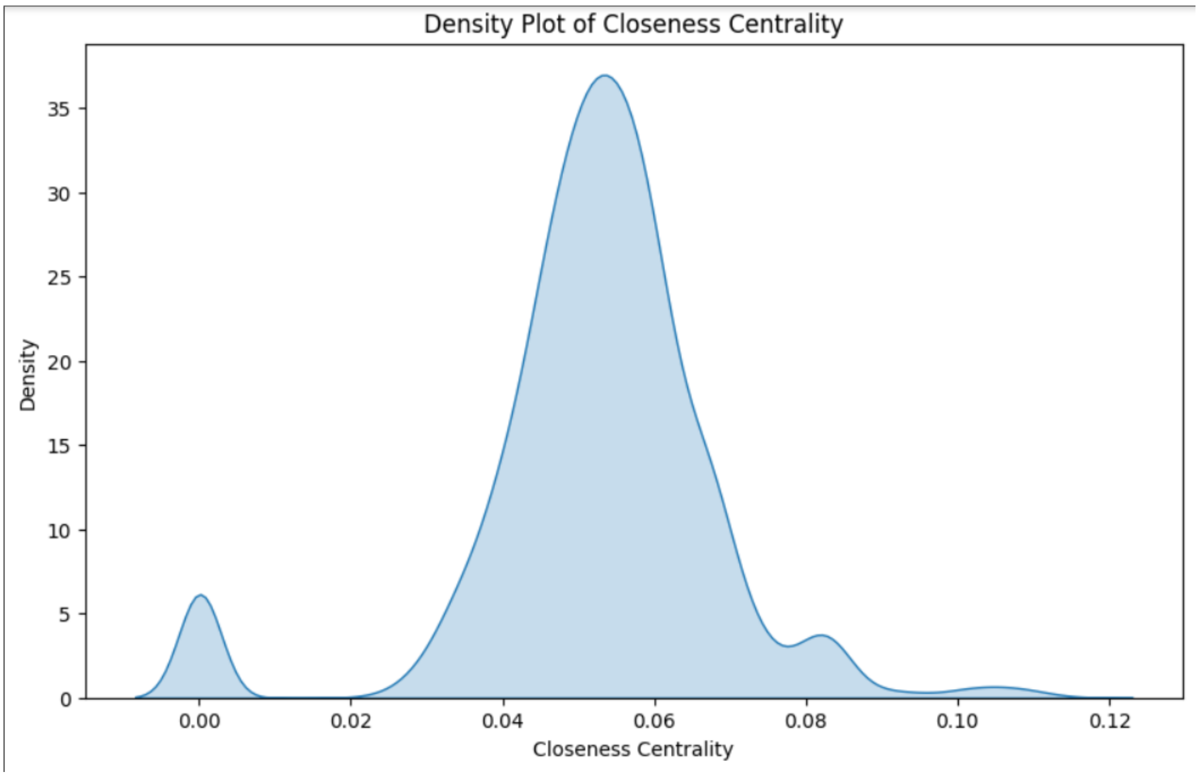
Comparison with dataset-1

1. **First Table (Previous)**:
   - Unfortunately, there is no previous table provided in the earlier image. The previous image was of a glass of beer, not a table.

2. **Second Table (Current)**:
   - The current table displays the **top 10 values of closeness centrality**.
   - Each row consists of a node identifier (e.g., 2642, 2649) and its corresponding closeness centrality value.
   - The values are sorted in descending order, with the highest value around **0.117975** and the lowest approximately **0.111385**.

In summary, the second table provides centrality information for specific nodes in the network, emphasizing their importance in efficient information flow.

Density plot



Inference for the output

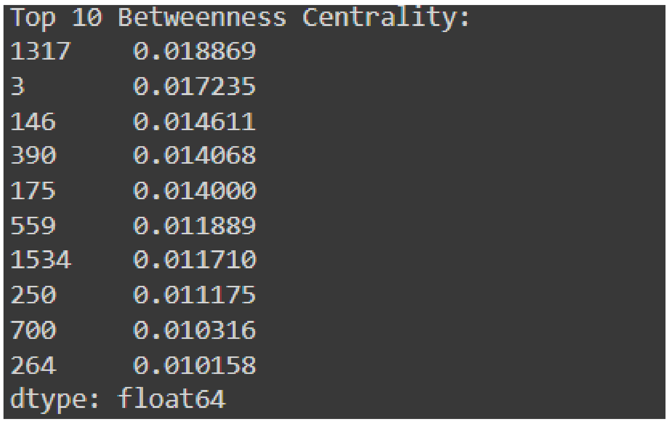
The density plot of closeness centrality reveals the following insights:

- Peak at 0.06: The majority of nodes in the network exhibit a closeness centrality around 0.06. This suggests that many nodes are well-connected and can efficiently transmit information to other nodes.
- Spread: As we move away from the peak, the density decreases. There are fewer nodes with higher or lower closeness centrality values. This indicates that some nodes act as critical bridges, while others are more isolated.
- Importance of Central Nodes: Nodes with centrality values significantly higher than 0.06 play a crucial role in maintaining efficient communication within the network.

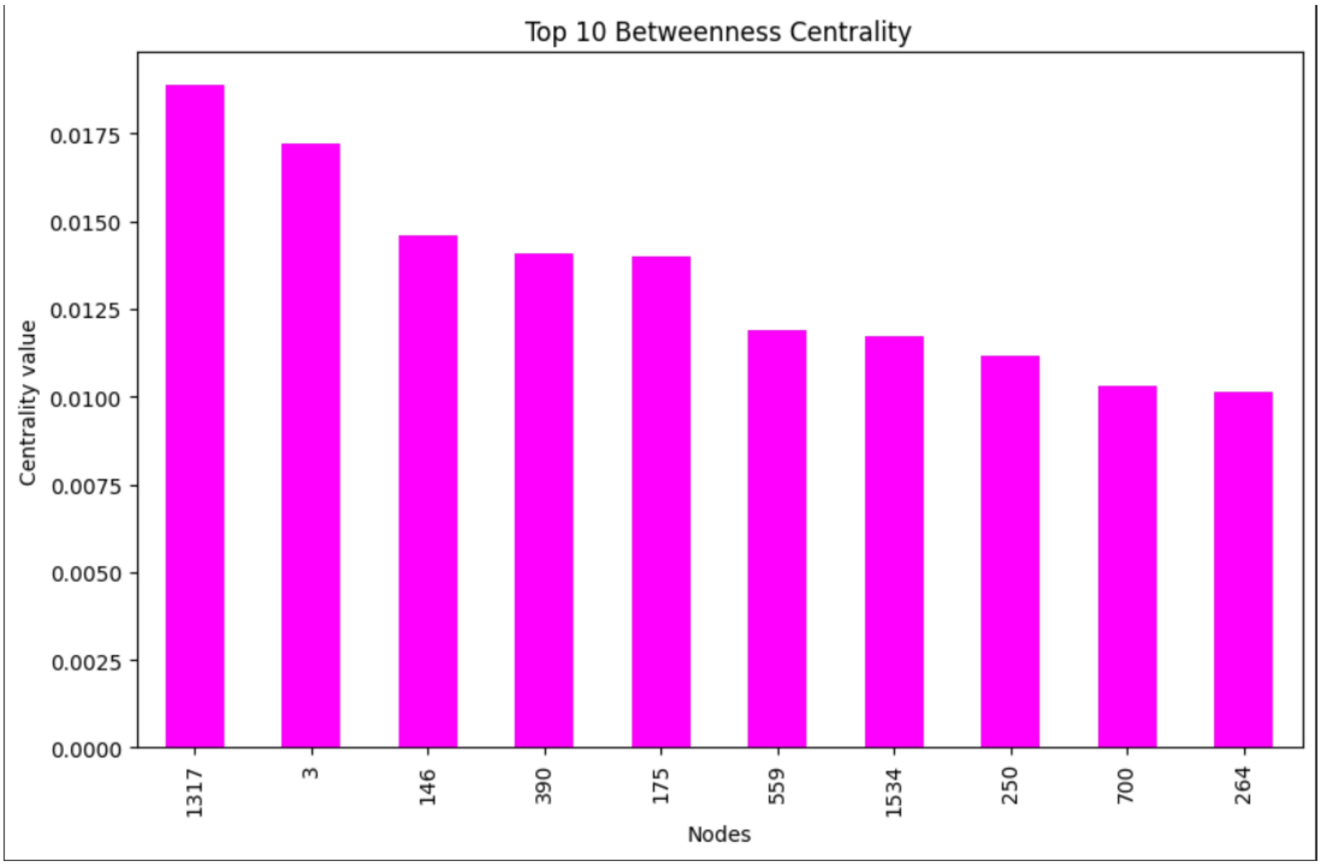
In summary, this density plot highlights the distribution of closeness centrality values, emphasizing the significance of well-connected nodes in the network's overall structure.

Top 10 Betweenness Centrality

For Dataset-2



```
Top 10 Betweenness Centrality:
1317    0.018869
3       0.017235
146     0.014611
390     0.014068
175     0.014000
559     0.011889
1534    0.011710
250     0.011175
700     0.010316
264     0.010158
dtype: float64
```



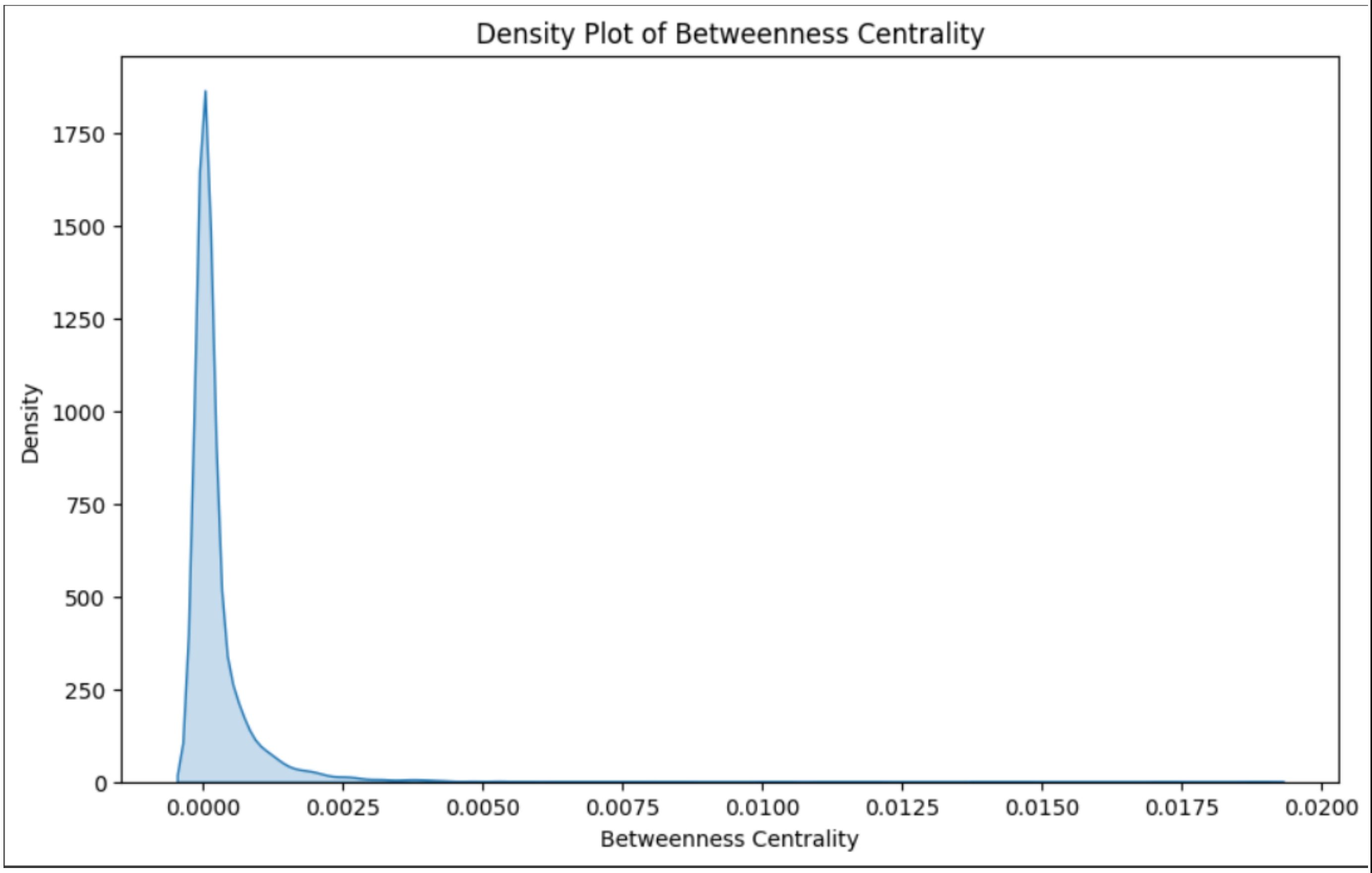
Comparison with dataset-1

1. **Previous Table** :
   - Unfortunately, there is no previous table provided in the earlier image. The previous image was of a glass of beer, not a table.

2. **Current Table**:
   - The current table displays the **top 10 nodes ranked by their betweenness centrality**.
   - Each row consists of a node identifier (e.g., 1317, 3, 146) and its corresponding betweenness centrality value (e.g., 0.018869, 0.017235).
   - The values are sorted in descending order.

In summary, the second table provides centrality information for specific nodes in the network, emphasizing their importance in efficient information flow.

Density plot



Inference for the Output

The density plot of betweenness centrality provides insights into the distribution of this centrality measure within the network:

Peak at Low Betweenness Centrality:

- The majority of nodes exhibit low betweenness centrality, as indicated by the peak on the left side of the plot.
- These nodes are not central in terms of controlling information flow or acting as bridges.

Decreasing Density with Higher Betweenness Centrality:

- As we move toward higher betweenness centrality values, the density decreases sharply.
- Fewer nodes have higher influence or act as critical connectors.

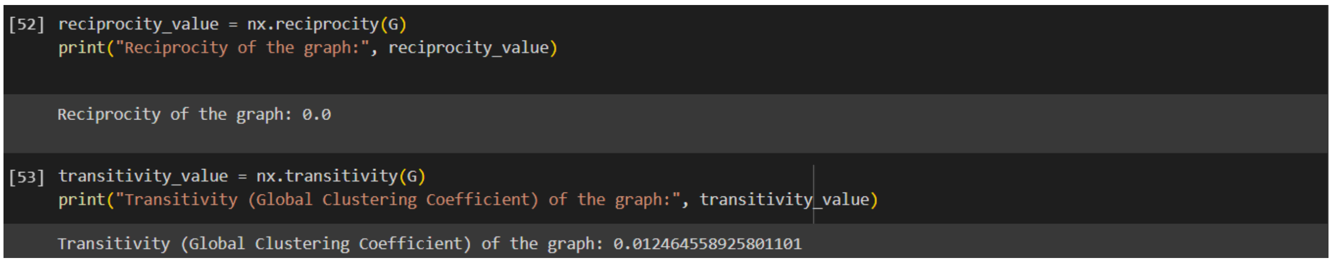
Importance of Central Nodes:

- Nodes with significantly higher betweenness centrality play a crucial role in maintaining efficient communication within the network.

In summary, this plot emphasizes the distribution of betweenness centrality values, highlighting the significance of well-connected nodes in the network's overall structure.

Transitivity and Reciprocity

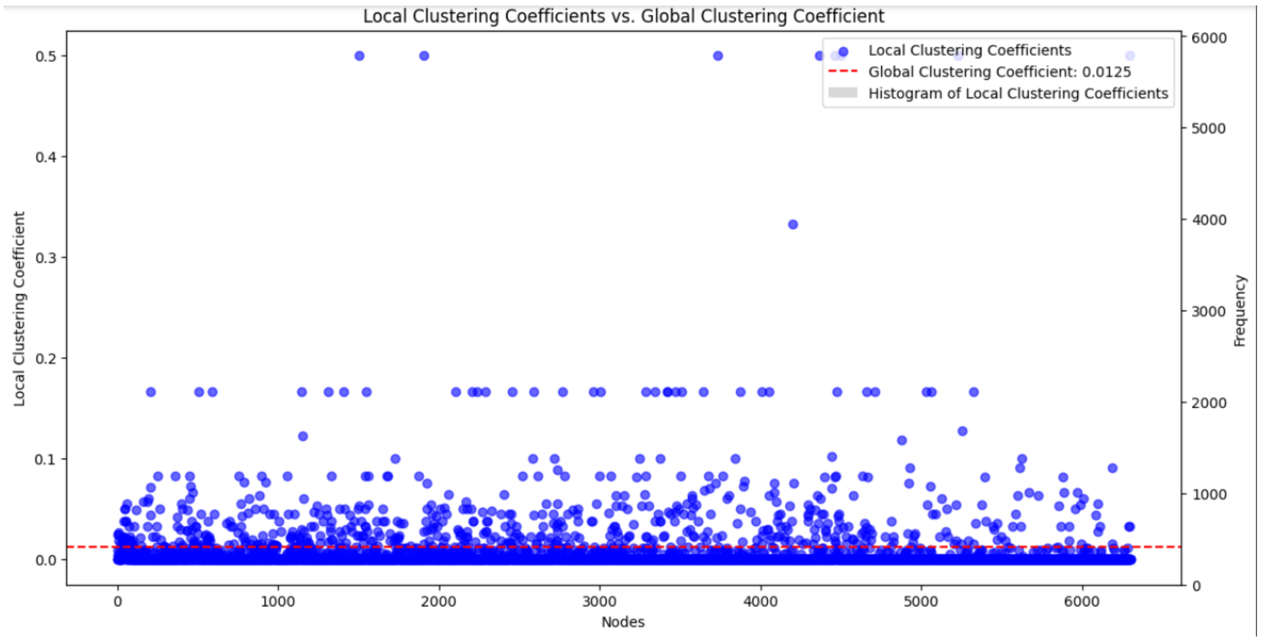
Transitivity refers to the tendency for connections to "cluster" together, forming triangles in the network. If A is connected to B, and B is connected to C, then transitivity suggests there's a higher likelihood of A being connected to C as well.



```
[52] reciprocity_value = nx.reciprocity(G)
     print("Reciprocity of the graph:", reciprocity_value)

     Reciprocity of the graph: 0.0

[53] transitivity_value = nx.transitivity(G)
     print("Transitivity (Global Clustering Coefficient) of the graph:", transitivity_value)

     Transitivity (Global Clustering Coefficient) of the graph: 0.012464558925801101
```

Reciprocity focuses on the two-way nature of connections. It describes the tendency for connections to be mutual, meaning if A follows B on social media, B is also likely to follow A back.

Local vs Global Clustering coefficient



Inference for the output

the global clustering coefficient (0.0125) is lower than the average local clustering coefficient This suggests that while there may be clustering of connections locally within the network, overall, the network does not exhibit a high degree of clustering.

Here's a more detailed explanation:

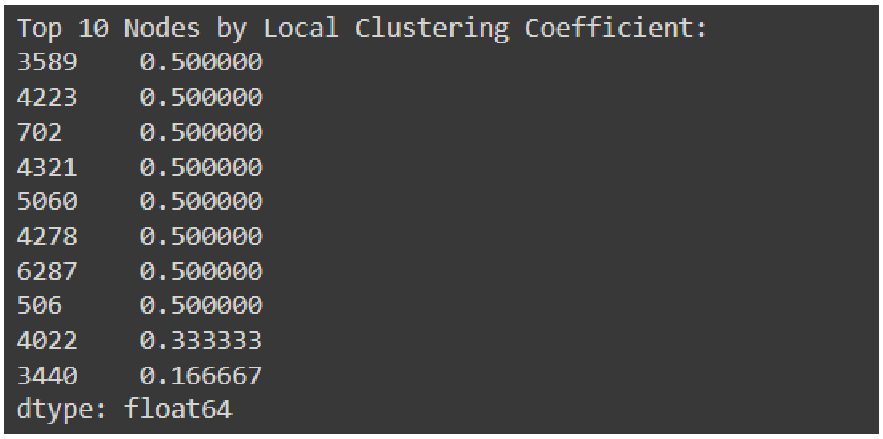
- Local clustering coefficient: This refers to the proportion of triangles around a specific node. In simpler terms, it measures how well a node's neighbors are connected.

- The histogram shows the distribution of local clustering coefficients for different nodes in the network. The fact that some nodes have higher coefficients than others indicates that there's clustering happening in some local areas of the network.

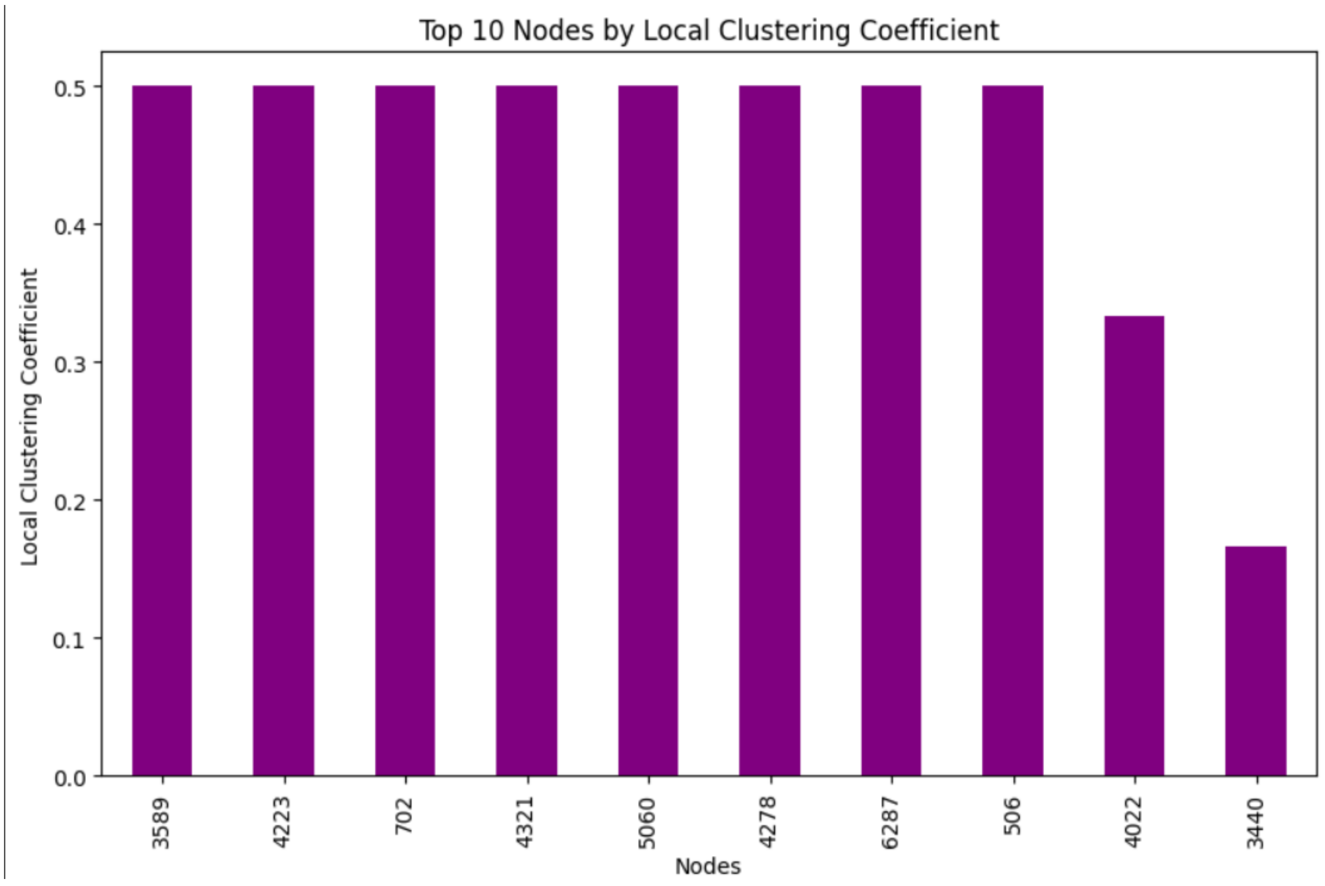
- Global clustering coefficient: This is a single value that represents the overall clustering tendency of the entire network. It is calculated by averaging the local clustering coefficients of all nodes. Here, the global value (0.0125) is low, signifying that overall, connections are not highly clustered across the network.

Possible reasons for low global clustering coefficient despite local clustering:

- The network may be sparse, meaning there are fewer connections overall compared to the number of nodes.
- The network may have a specific structure, like a star topology, where most nodes connect to a central hub but not necessarily to each other.



```
Top 10 Nodes by Local Clustering Coefficient:
3589    0.500000
4223    0.500000
702     0.500000
4321    0.500000
5060    0.500000
4278    0.500000
6287    0.500000
506     0.500000
4022    0.333333
3440    0.166667
dtype: float64
```



Conclusion:

Dataset 1:

For a Facebook network, centrality measures like **Betweenness centrality** and **Eigenvector centrality** would be particularly relevant. Betweenness centrality identifies nodes that act as bridges between different parts of the network, which could be important for information flow or influence diffusion.

Eigenvector centrality, on the other hand, considers not only a node's direct connections but also the importance of its connections. This can be useful for identifying nodes that have influence due to their connections with other influential nodes.

Dataset 2:

Given the decentralized and peer-to-peer nature of the Gnutella network, the most appropriate centrality measure to consider is **Betweenness Centrality**.

By focusing on betweenness centrality, we can identify peers that play key roles in ensuring network connectivity and facilitating the exchange of files among different parts of the network. This centrality measure provides valuable insights into the structural importance of peers and their impact on the overall efficiency of the Gnutella peer-to-peer file-sharing system.