Twitch Social Network Analysis

Introduction to Social Networks Project

Team Members

Name	ID
أحمد محمد محمود محمود محمدي	22010038
محمد محمود محمد حسانين	22011680
عمر حافظ مأمون محمد حسن محمد	22011562
الحسين ياسر إبراهيم السيد	22010056
فارس صلاح الدين عبدالفتاح النحاس	22010183
طارق مصطفى محمد عزالدين زين	22010124
إبراهيم محمد عبدالغني علي	22010010

Introduction

This report analyzes the Twitch social networks dataset obtained from Stanford's SNAP repository. The analysis includes statistical metrics and visualizations generated using **Gephi**. These insights help in understanding the meaning of the dataset.

Problem Description

This project aims to analyze the **Twitch streamer-streamer social network** to uncover patterns and relationships within the platform's community. By calculating basic and advanced network measures—such as **connectivity, centrality,** and **visualizations**—we aim to provide insights into the structure of the network. The ultimate goal is to identify how these network properties correlate with a key metric: **streamer viewership**.

Data Overview

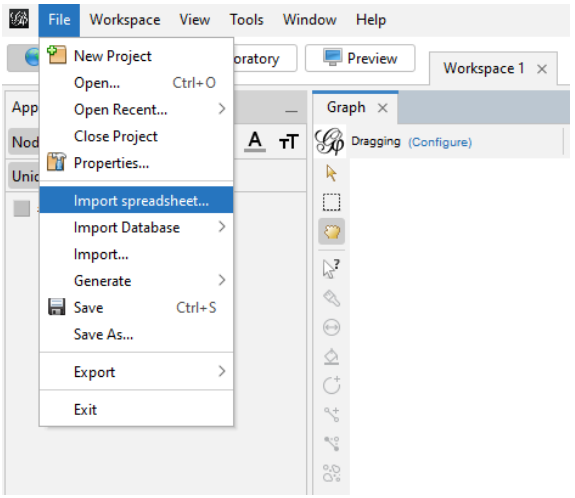
The dataset consists of various files detailing node and edge statistics, as well as network metrics like **in, out, and total degree, clustering coefficient** and **different centralities**.

A dataset of Twitch users with **7,126 nodes, 35,324 edges.**

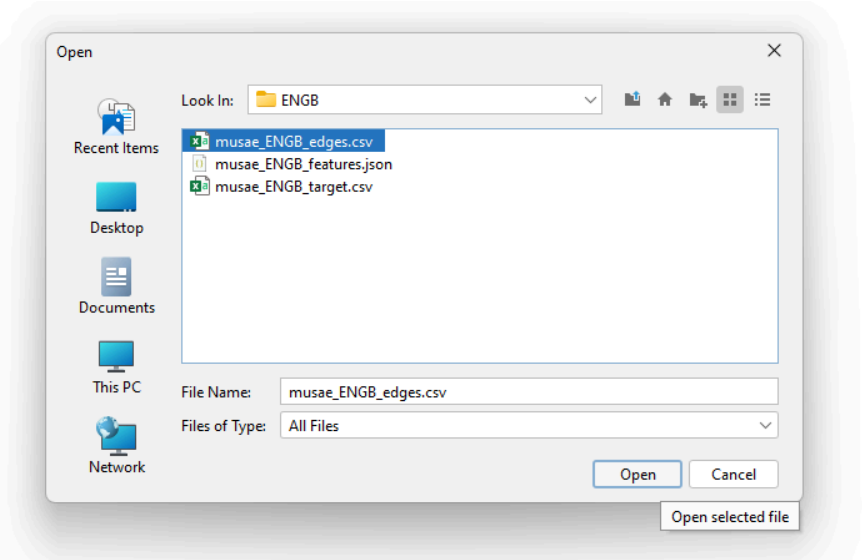
Nodes represent the streamers, while edges indicate a follow relationship from one streamer to another.

Project Steps

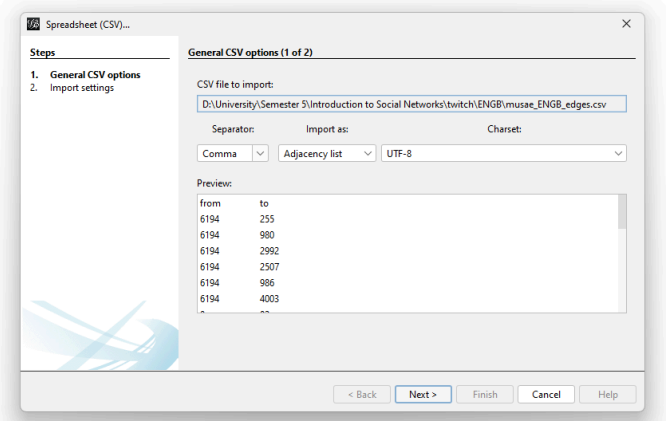
Step 1:
Import the dataset spreadsheet.



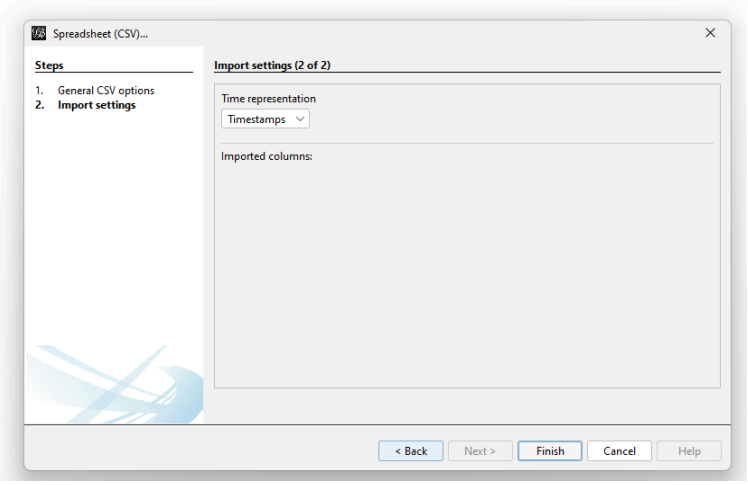
Step 2:
Navigate to the file location on your device.



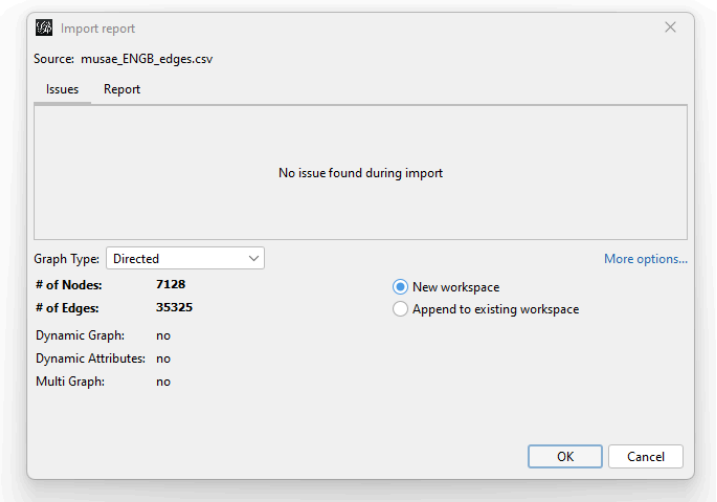
Step 3:
Set your preferred options and press next.



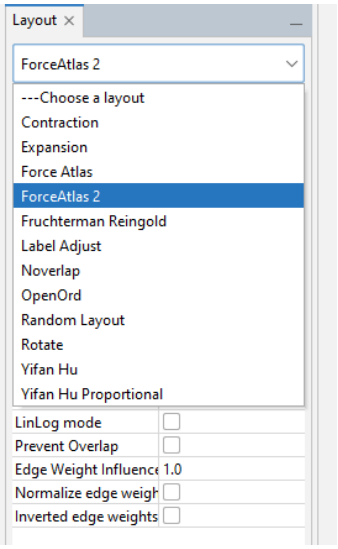
Step 4:
Choose the time representation and press next.



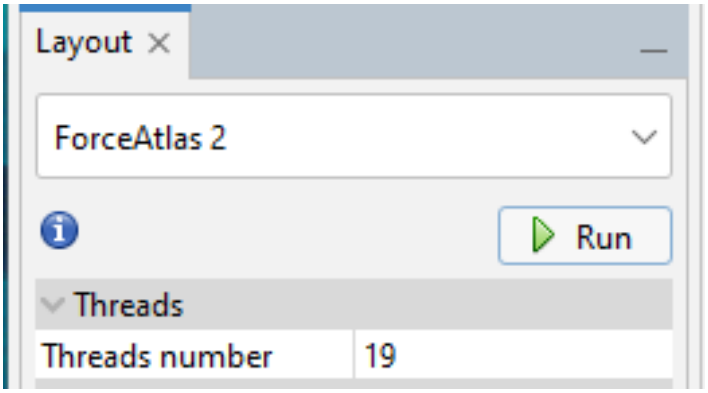
Step 5:
Choose the graph type and press OK.



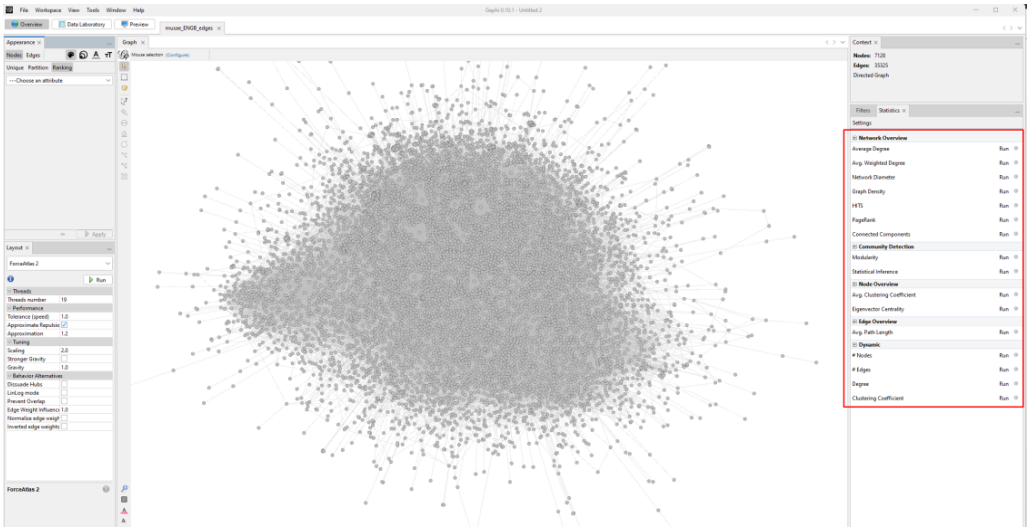
Step 6:
Select your preferred layout type.



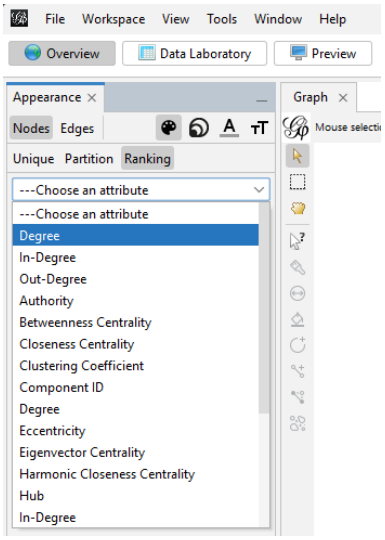
Step 7:
Press Run to create the layout.



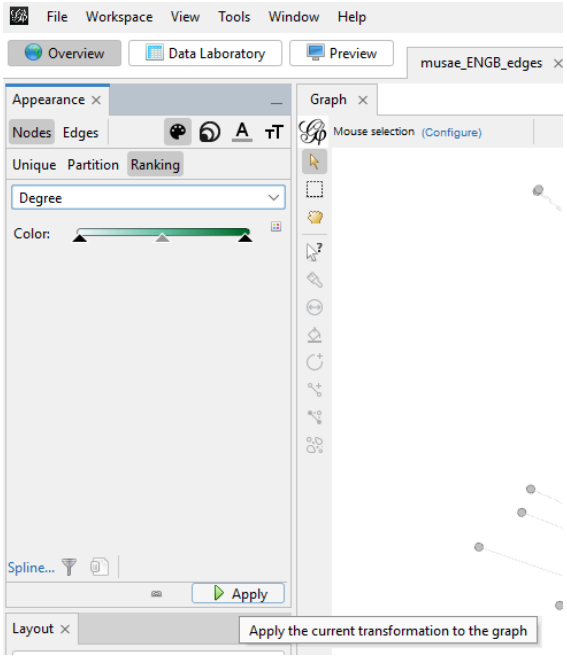
Step 8:
Now that we have created the layout, we can calculate different statistics from the menu on the right.



Step 9:
From the Ranking tab, you can visualize different attributes.



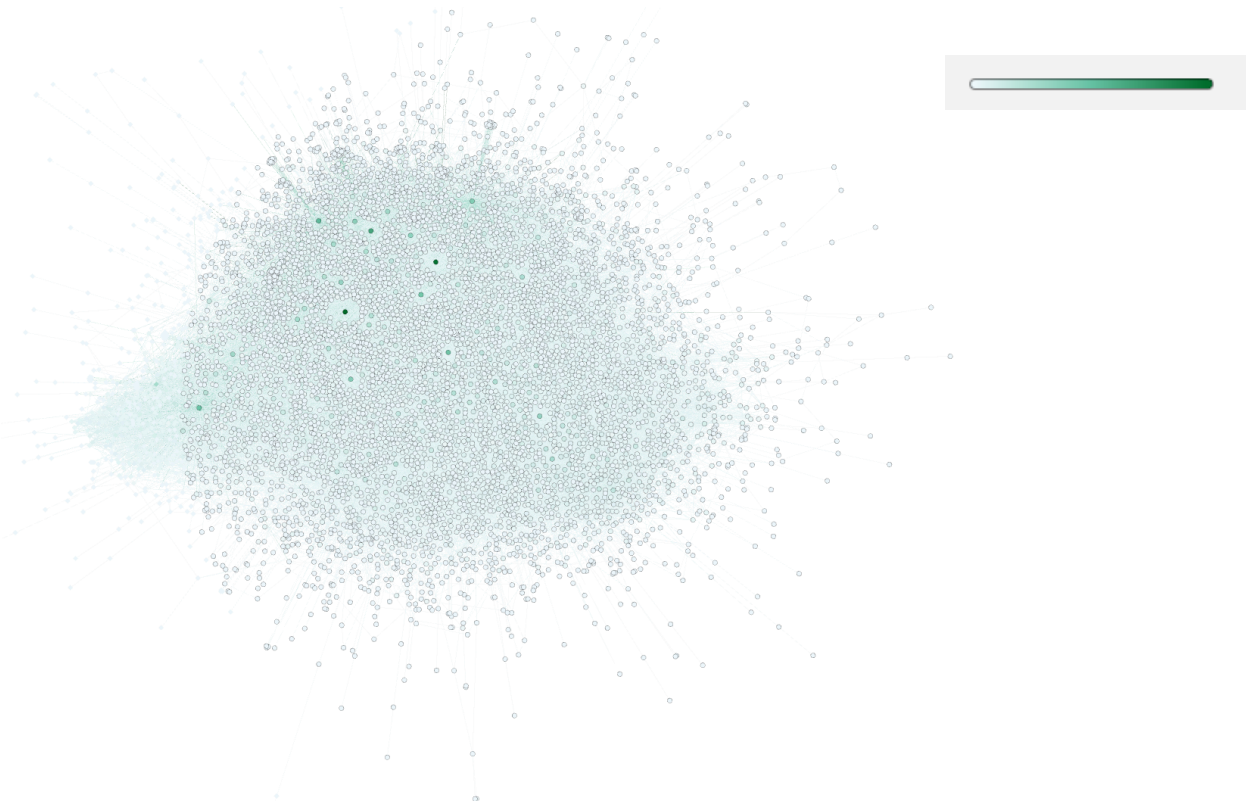
Step 10:
Afterwards choose your preferred color palette and press Apply.

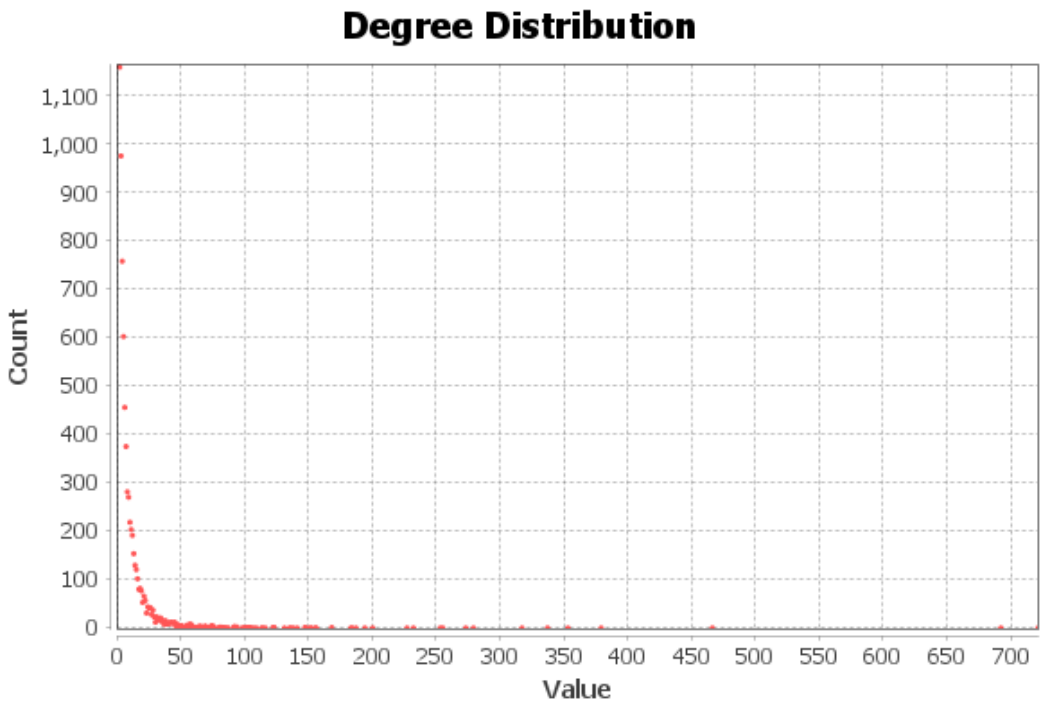


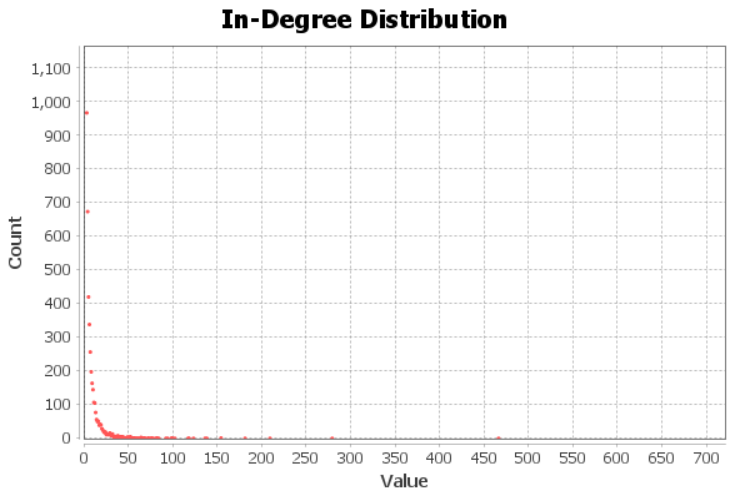
Visualizations

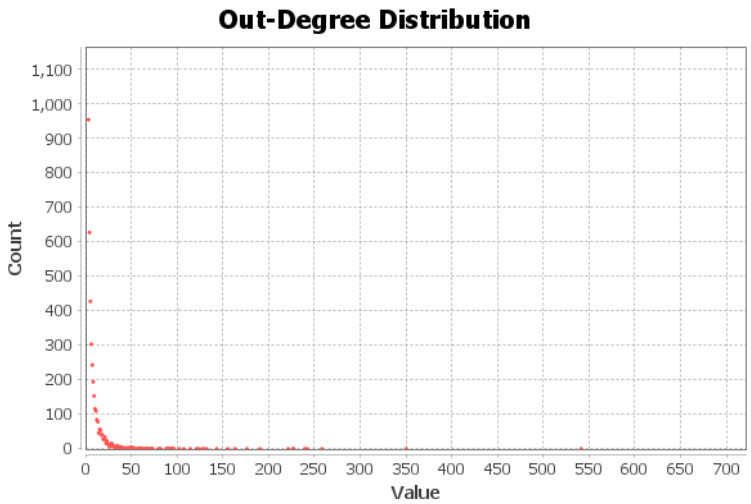
Below are visualizations of each statistic using a graph representation, with darker shades representing higher values and a scatterplot for the distribution of each statistic.

Degree Distribution:



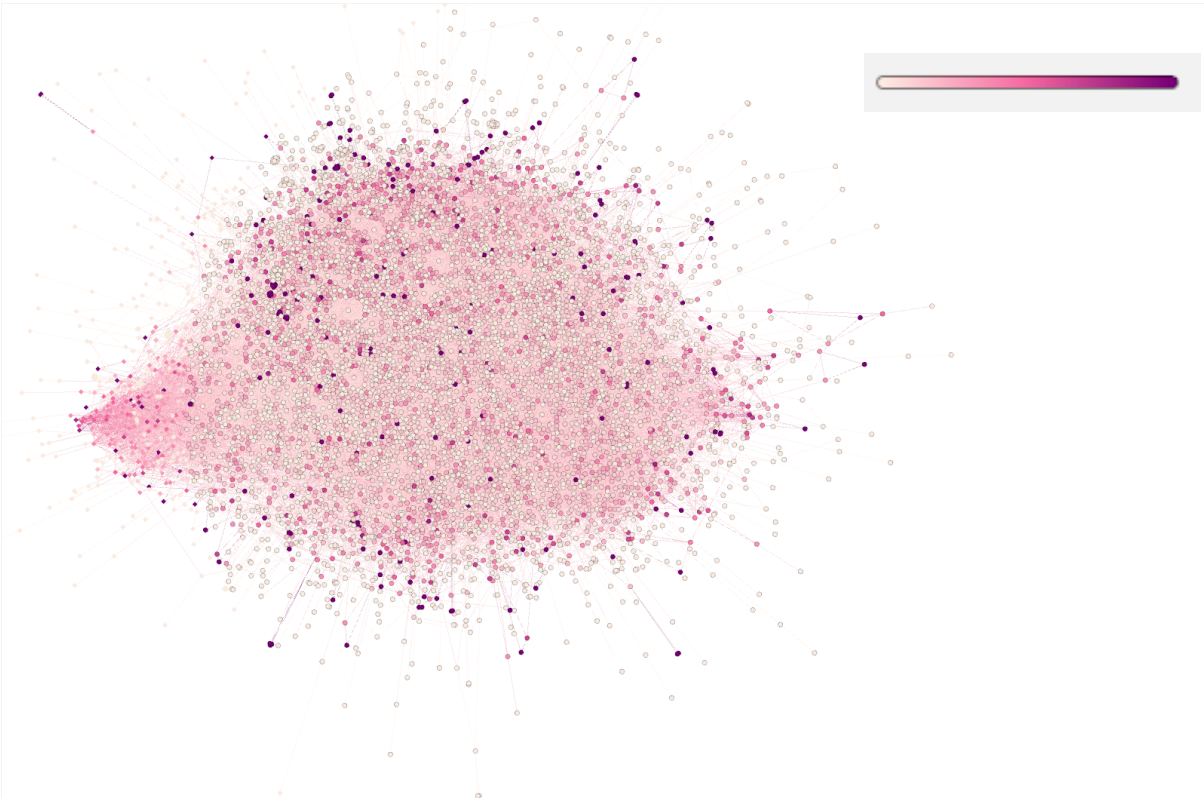


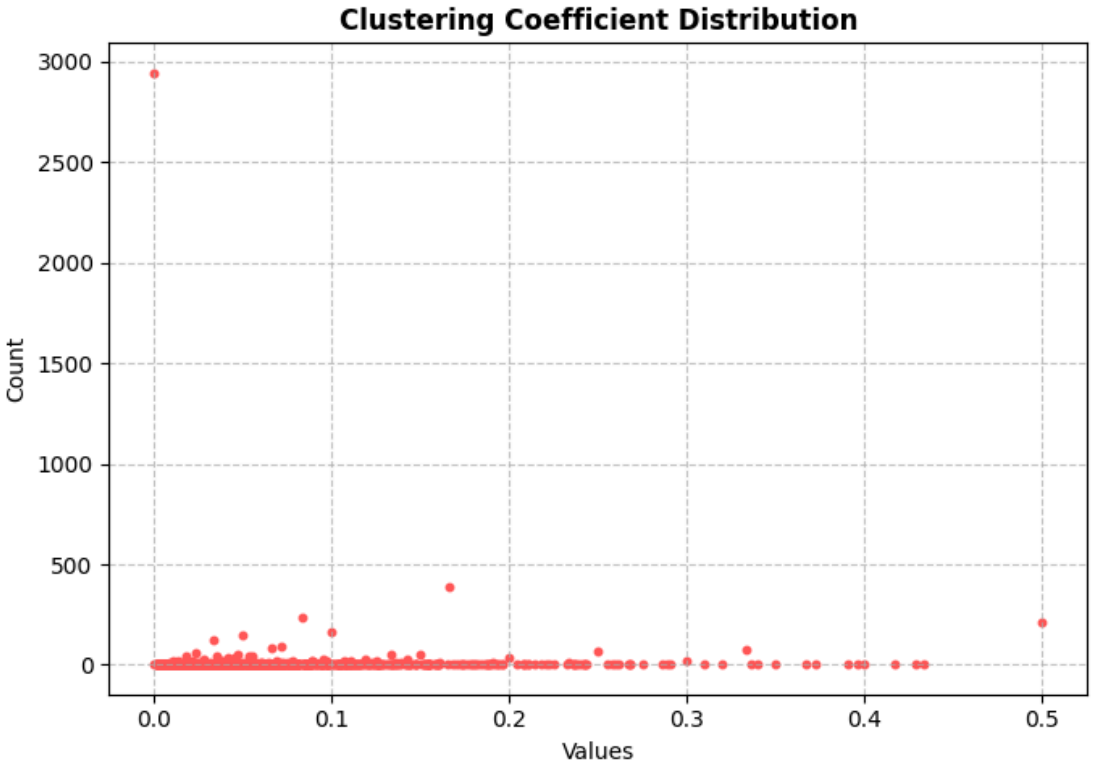




Most of the nodes have a low in, out, and total degrees indicating that most streamers do not follow many other streamers.

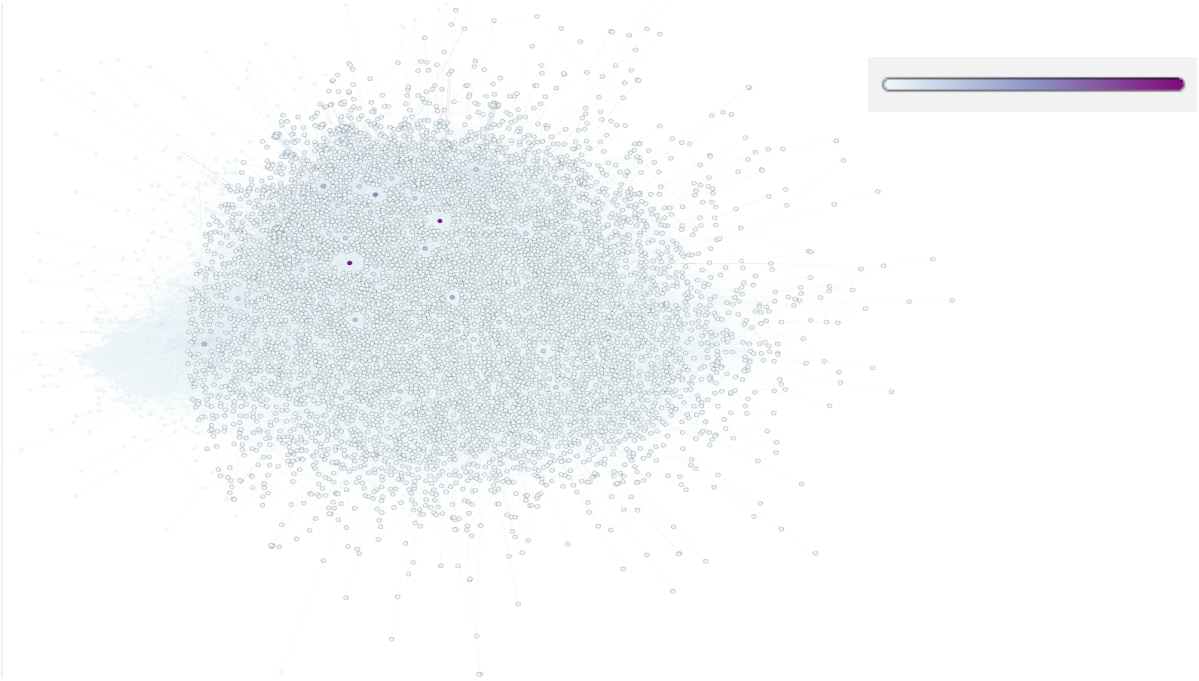
Average Clustering Coefficient:

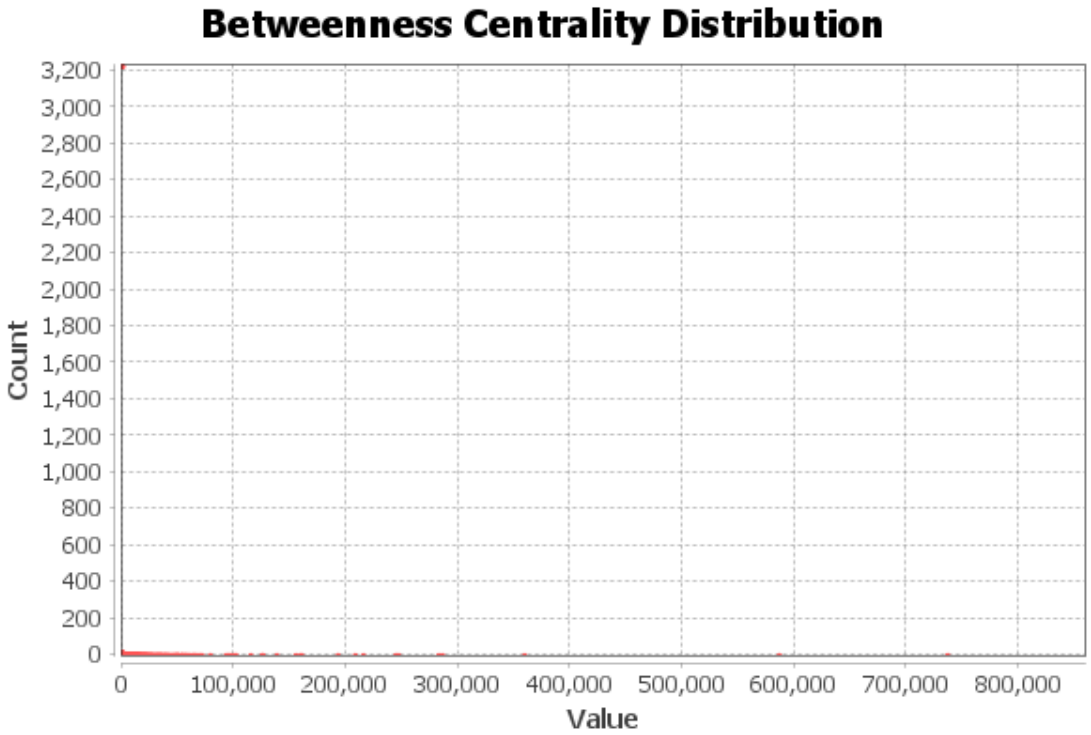




Most of the nodes have a low clustering coefficient indicating that there are not many strongly connected components.

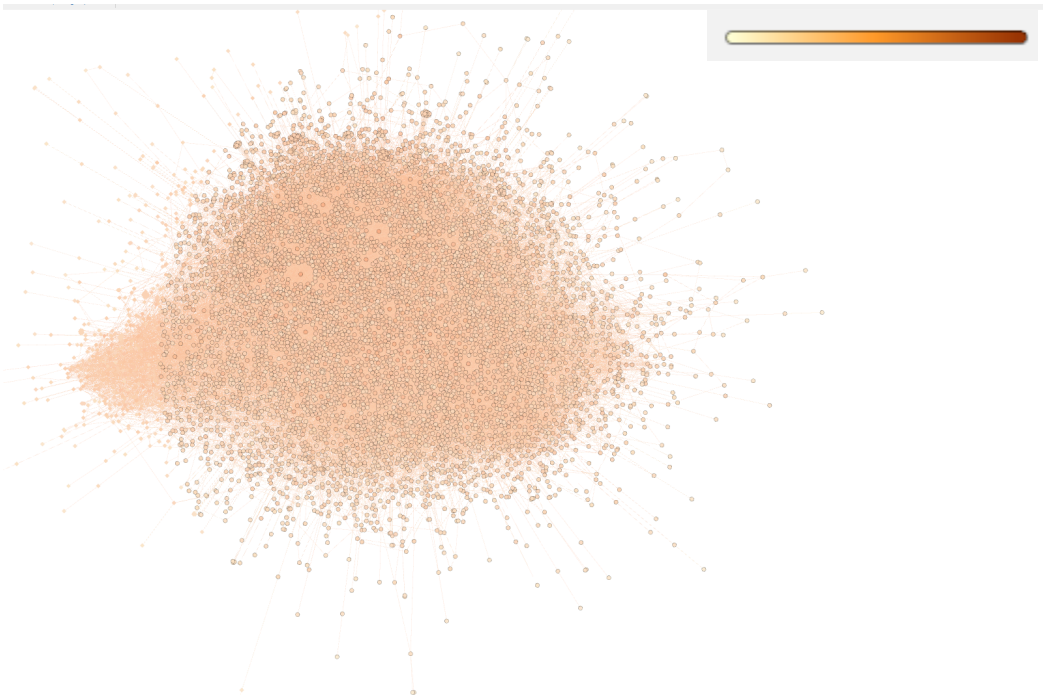
Betweenness Centrality:

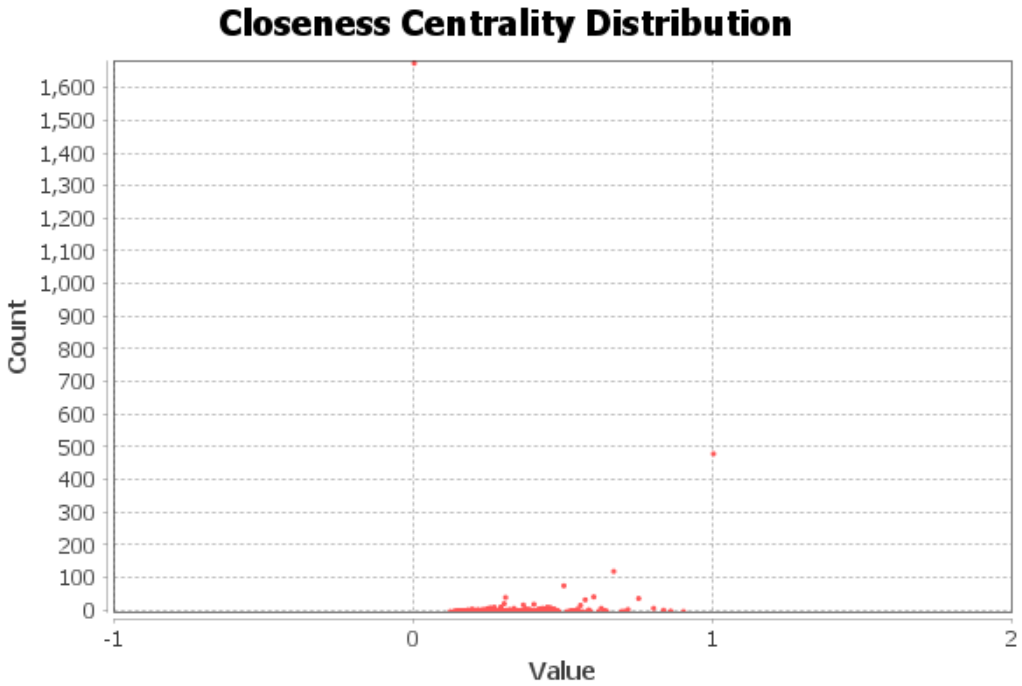




About half of the nodes have near-zero betweenness, indicating minimal control over the network, while the other half show moderate to high values, suggesting a more significant role in connecting different parts of the network with. With some nodes having extremely high betweenness, likely representing top global streamers.

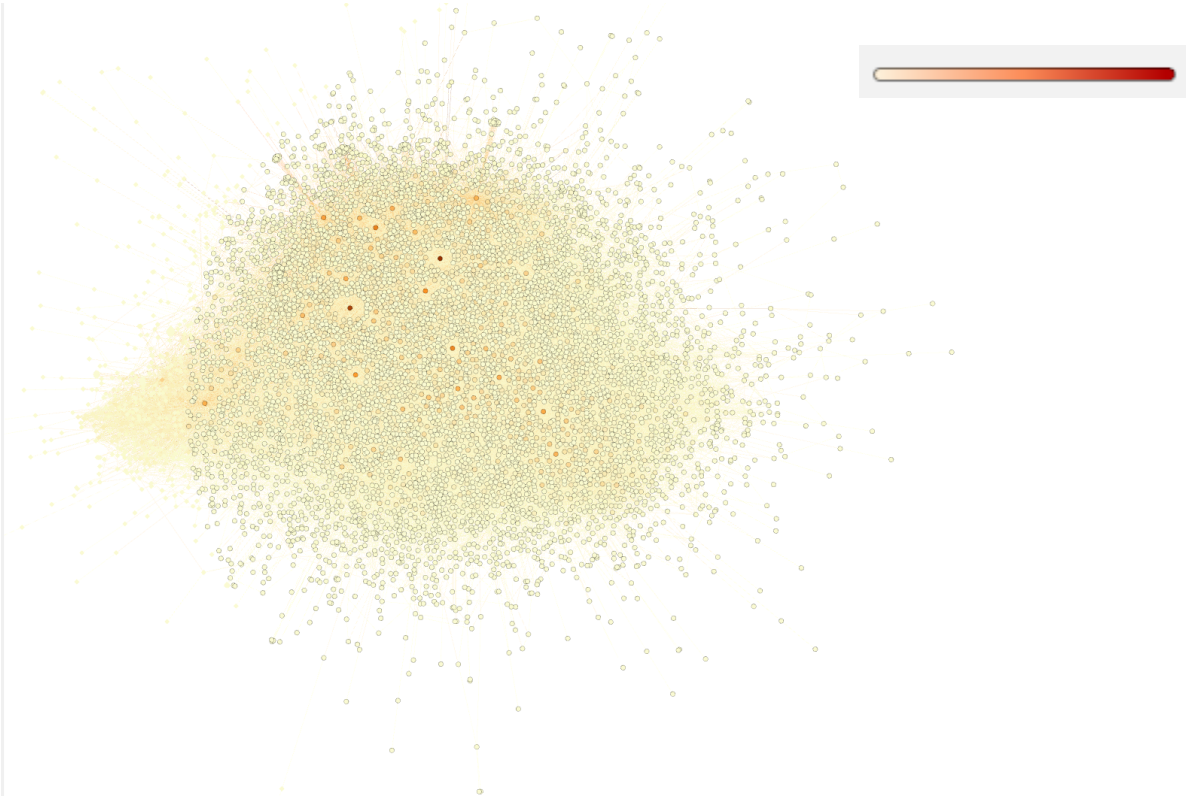
Closeness Centrality:

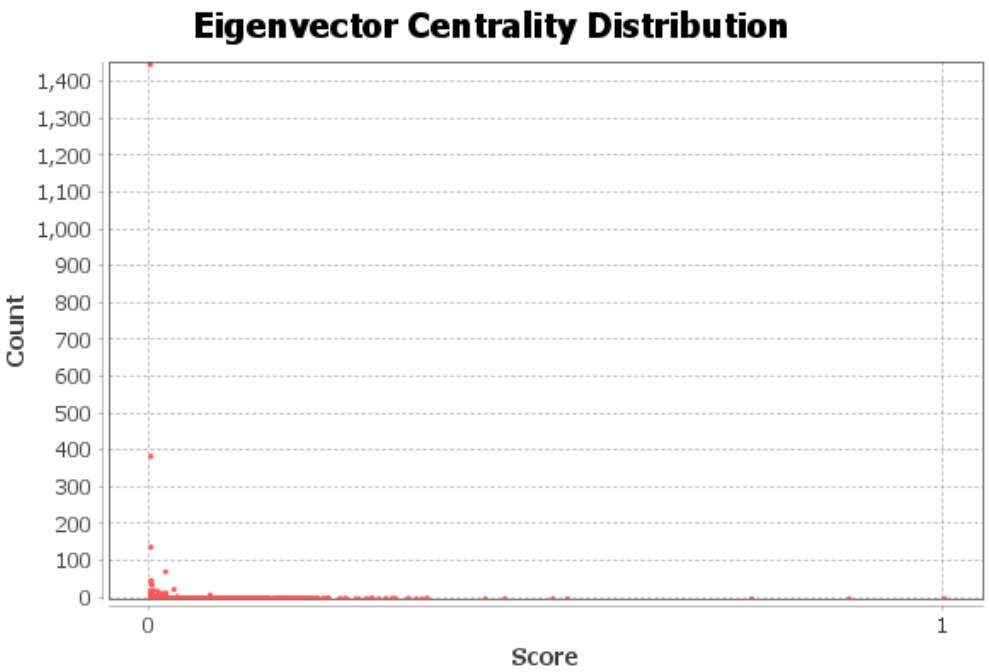




Most of the nodes have a closeness centrality of near-zero indicating that the nodes are far from most other nodes in the network, meaning it takes many steps to reach others. While some nodes have a high closeness centrality indicating that they're well connected with most of the network.

Eigenvector Centrality:





Most nodes have low eigenvector centrality, indicating they lack connections to influential nodes within the network.

Representing Graph

Example of edge list representing:
(6194, 255)
(6194, 980)
(6194, 2992)
(6194, 2507)
(6194, 986)
(6194, 4003)
(7106, 6211)
(7106, 6611)
(7106, 1348)
(7106, 2239)

Example of Adjacency list representing:
6194: 255, 980, 2992, 2507, 986, 4003
255: 6194, 990, 863, 2481, 5498, 4479
2992: 6194, 1598, 2732, 4330

Coding

At first we imported the “pandas” library to store the dataset in a data frame and start processing it.
After that we dropped some columns from the data frame that weren’t needed.

```
df = df.drop(columns=['componentnumber','weighted indegree','weighted outdegree'
                      ,'Weighted Degree','Eccentricity','strongcompnum',
                 'harmonicclosnesscentrality','Authority','Hub','modularity_class','stat_inf_class'])
```

Then, we began to create two separate files one for the node statistics file, and another for the target file which contains the features of the dataset.

Then we sorted them by ID.

```
file1 = pd.read_csv("Nodes_statistics.csv")
file2 = pd.read_csv("musae_ENGB_target.csv")
```

```
file2_new = file2.sort_values(by='id')
file2_new
```

	days	mature	views	partner	id
177	324	True	1678	False	0
3658	1618	False	622427	True	1
5858	2587	True	1465	False	2
1696	1517	False	13141	False	3
3634	1889	True	448	False	4
...	...	...	...	...	...
5903	1884	False	6025	False	7121
2757	1001	True	187417	False	7122
7118	1366	False	1444	False	7123
4383	1119	False	1017	False	7124
895	3604	True	2119	False	7125

7126 rows × 5 columns

```
file1_new.to_csv('file_1.csv', index=False)
file2_new.to_csv('file_2.csv', index=False)
```

```
f1 = pd.read_csv('file_1.csv')
f2 = pd.read_csv('file_2.csv')
```

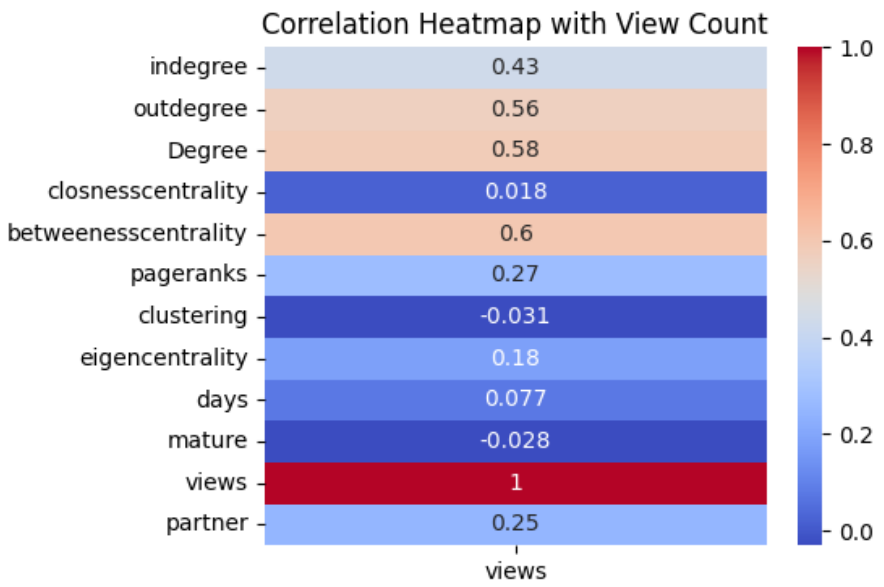
```
f1[['days','mature','views','partner']] = f2[['days','mature','views','partner']]
```

We saved the sorted data in two separate csv files, then we read these csv files again and we added 4 columns to the first file and assigned it to 4 columns from the second file to merge them.

```
# Calculate correlations for 'view_count'
correlation = visul_data.corr()[['views']]

# Plot the heatmap
plt.figure(figsize=(5, 4))
sns.heatmap(correlation, annot=True, cmap='coolwarm', cbar=True)
plt.title('Correlation Heatmap with View Count')
plt.show()
```

Then, we used this code to visualize a correlation between the views and all the other statistics measures.



Then, we made a heatmap which tells us which one of these statistics has high correlation with view count.
The statistic with the highest correlation is the **betweenness centrality**.

```python
def visual_scatter_plot_subplot(data, plots, figsize=(15, 10)):
    rows = 2
    cols = 2
    plt.figure(figsize=figsize)

    for i, (x_col, y_col, title) in enumerate(plots, 1):
        x = data[x_col]
        y = data[y_col]
        plt.subplot(rows, cols, i)
        plt.scatter(x, y, color='#ff5555', marker='o', s=10, alpha=1)
        plt.xlabel(x_col)
        plt.ylabel(y_col)
        plt.title(title, fontweight='bold')
        plt.grid(True, linestyle='--', alpha=0.7)

    plt.tight_layout()
    plt.show()
```

0.0s

```python
plots = [
    ('betweenesscentrality', 'views',
     'betweenesscentrality effect on views'),

    ('Degree', 'views', 'Degree effect on views'),
    ('outdegree', 'views', 'outdegree effect on views')
]
visual_scatter_plot_subplot(final_data, plots)
```

0.4s

This function takes (data, plots, figsize(15, 10)), The data parameter is the dataset , the plots parameter is a tuple contains the x-axis , the y-axis and the title of the plot, the figsize just shape the plot the size we want.

```python
# Calculate the correlation matrix
correlation_matrix = visul_data.corr()

# Plot the heatmap
plt.figure(figsize=(12, 10))  # Adjust figure size as needed
sns.heatmap(correlation_matrix, annot=True,
            cmap='coolwarm', cbar=True, square=True)
plt.title('Correlation Heatmap for All Columns')
plt.show()
```

This code makes a heatmap for the columns demonstrating the correlation between each other.

Results

```
max_indegree = df['indegree'].max()
print(max_indegree)
df[df['indegree'] == max_indegree ]
```

```
465
```

	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
88	4949	465	226	691	0.43852	859191.964214	0.007983	0.003192	1.0

N**ode 4949** is the most followed node in the network, likely representing a highly influential entity or person (e.g., a prominent streamer or account) that many other streamers are connected to.

```
max_outdegree = df['outdegree'].max()
print(max_outdegree)
df[df['outdegree'] ==  max_outdegree]
```

```
540
```

	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
54	1773	180	540	720	0.407119	735562.185811	0.002475	0.001824	0.166374

Node 1773 follows the most streamers in the network. It is likely an account or entity actively engaging with many others, potentially representing an entity with broad interests or a user/network participant seeking to interact with a diverse set of streamers.

```
max_Degree = df['Degree'].max()
print(max_Degree)
df[df['Degree'] ==  max_Degree]
```

```
720
```

	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
54	1773	180	540	720	0.407119	735562.185811	0.002475	0.001824	0.166374

Node 1773 is the most connected node in the network, having a total of **720** connections. This includes:

- **180 incoming connections** (*indegree*), meaning 180 streamers follow this node.
- **540 outgoing connections** (*outdegree*), meaning this node follows 540 other streamers.

This suggests that **node 1773** is both actively following many streamers and also being followed by a notable number, highlighting it as a key participant in the network with significant engagement in both directions.

```python
max_betweenesscentrality = df['betweenesscentrality'].max()
print(max_betweenesscentrality)
df[df['betweenesscentrality'] ==  max_betweenesscentrality]
```

```
859191.964214
```

	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
88	4949	465	226	691	0.43852	859191.964214	0.007983	0.003192	1.0

Node 4949 has the highest betweenness centrality in the network**.** This means it lies on the shortest paths between many other nodes. Essentially, it's a crucial connector, facilitating communication and information flow within the network.This node is considered a key player or influencer, as it can potentially control the spread of information or resources.
In this context, **node 4949** might be a central figure in the streamer community, serving as **a bridge** between different groups or facilitating collaborations.

```python
max_closnesscentrality = df['closnesscentrality'].max()
print(max_closnesscentrality)
df[df['closnesscentrality'] ==  max_closnesscentrality]
```

```
1.0
```

	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
46	6286	10	1	11	1.0	1313.135242	0.000129	0.072727	0.019506
50	8	0	1	1	1.0	0.000000	0.000056	0.000000	0.000000
64	10	0	1	1	1.0	0.000000	0.000056	0.000000	0.000000
71	12	0	2	2	1.0	0.000000	0.000056	0.000000	0.000000
78	3418	22	1	23	1.0	946.162726	0.000429	0.015810	0.045676
...	...	...	...	...	...	...	...	...	...
7108	6230	0	2	2	1.0	0.000000	0.000056	0.000000	0.000000
7109	2289	0	1	1	1.0	0.000000	0.000056	0.000000	0.000000
7111	7038	1	1	2	1.0	65.920495	0.000305	0.000000	0.002677
7112	2659	0	2	2	1.0	0.000000	0.000056	0.000000	0.000000
7115	6444	0	1	1	1.0	0.000000	0.000056	0.000000	0.000000

483 rows × 9 columns

483 Nodes have the highest Closeness centrality**(1.0)** measures how efficiently a node can reach others in the network. Indicating that they sit on the shortest paths between many other nodes.

- A value of 1.0 suggests these streamers are at an optimal position to interact with the rest of the network.
- This result indicates that the network might be tightly connected or structured in a way where most nodes are equidistant to others.

```python
max_clustering = df['clustering'].max()
print(max_clustering)
df[df['clustering'] ==  max_clustering]
```

```
0.5
```

	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
233	453	1	1	2	0.248470	0.0	0.000056	0.5	0.000757
370	2658	1	1	2	0.300532	0.0	0.000056	0.5	0.000851
437	31	0	2	2	0.380000	0.0	0.000056	0.5	0.000000
525	1276	2	1	3	0.212095	0.0	0.000079	0.5	0.001607
572	3178	2	0	2	0.000000	0.0	0.000063	0.5	0.020112
...	...	...	...	...	...	...	...	...	...
7066	5484	0	2	2	0.666667	0.0	0.000056	0.5	0.000000
7067	6646	2	0	2	0.000000	0.0	0.000116	0.5	0.002285
7090	4666	2	0	2	0.000000	0.0	0.000190	0.5	0.001966
7091	5905	1	1	2	1.000000	0.0	0.000103	0.5	0.001102
7103	6566	1	1	2	0.666667	0.0	0.000159	0.5	0.008759

212 rows × 9 columns

212 Nodes with the highest clustering coefficient **(0.5)** are embedded in tightly-knit groups, where their neighbors are likely interconnected. These streamers may be part of small, cohesive communities or sub-networks of streamers who frequently follow one another. However, their lack of high centrality in other measures suggests they may not be influential in the network's larger structure.

```python
max_eigencentrality = df['eigencentrality'].max()
print(max_eigencentrality)
df[df['eigencentrality'] ==  max_eigencentrality]
```

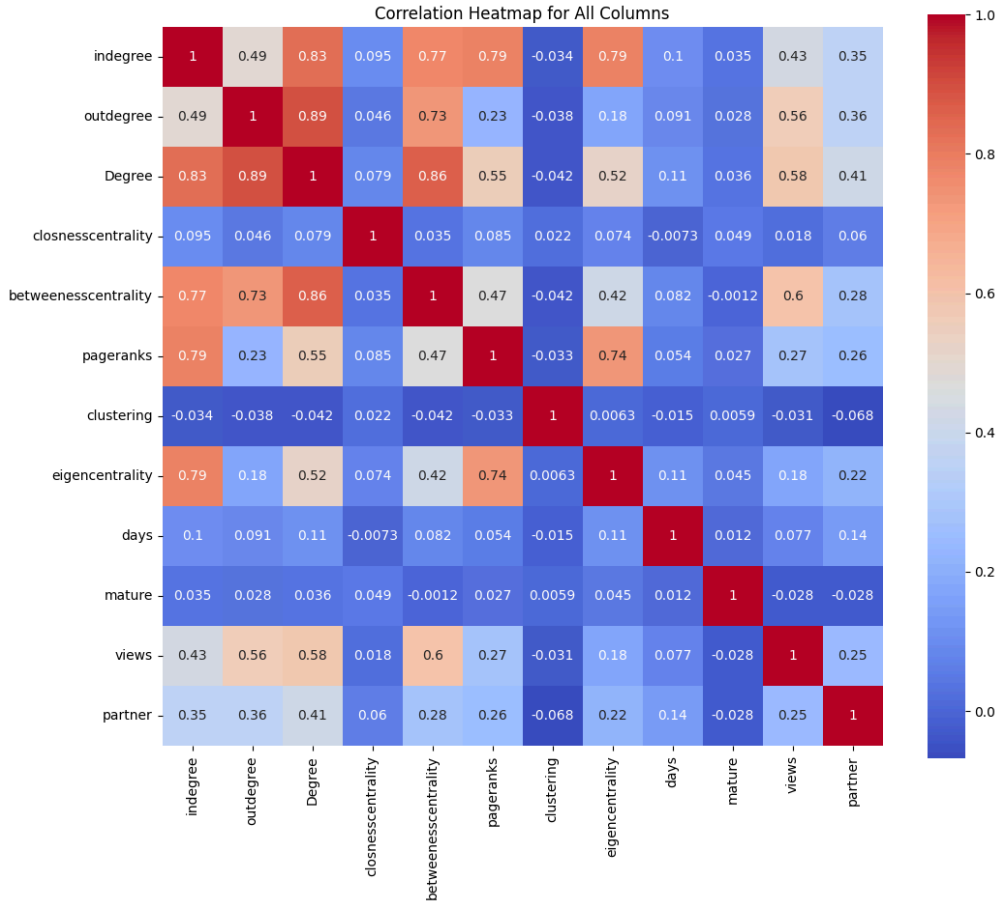
```
1.0
```

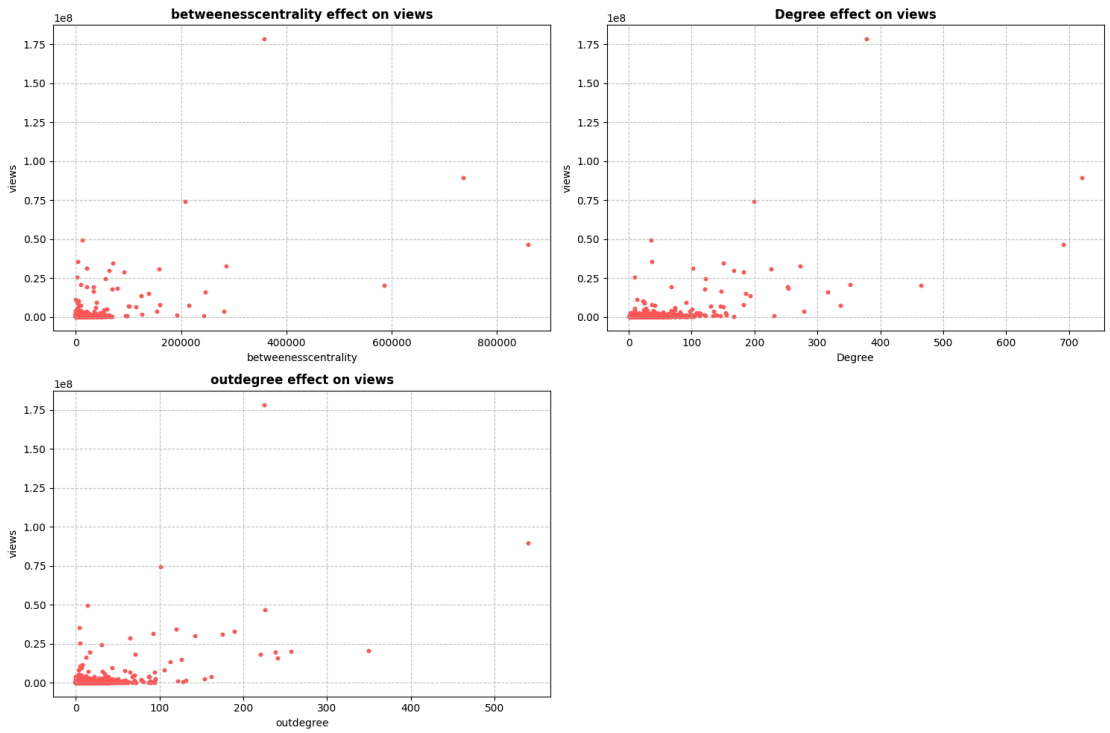
	Id	indegree	outdegree	Degree	closnesscentrality	betweenesscentrality	pageranks	clustering	eigencentrality
88	4949	465	226	691	0.43852	859191.964214	0.007983	0.003192	1.0

Node 4949 is a key influencer in the Twitch streamer network. It doesn't only have a high number of followers (in-degree) but also is connected to other influential nodes, making it central to the network's structure and interactions. Its high eigenvector centrality and betweenness centrality suggest that this node is integral for connecting various clusters or communities within the network.

It represents a highly prominent or well-known streamer who interacts broadly across different groups.

Data Correlation Representation using Heat Maps and Scatter Plots





The majority of data points cluster in the bottom-left corner of the charts because the top 0.5% of streamers account for 75% of total viewers[1]*, leaving the vast majority with significantly lower view counts.*

From the previous graphs, we can conclude that the 3 attributes that influence the **view count** the most are **'Degree'**, '**Out Degree**' and '**Betweenness Centrality**'.

Which tells us that successful streamers:

- Follow and engage with other streamers to help establish connections and lead to collaborations.
- Connect other diverse streamers as a key link in the network.

Difficulties Faced

- **Picking the dataset:** We were unsure what we should look for in a dataset before deciding on this idea.
- **Resource Heavy Applications:** Originally we used **'Cytoscape'** but instead, we used **'Gephi'** as it was a bit more lightweight and easier to work with.

Conclusion

The analysis of the Twitch social network dataset has provided valuable insights into the structure and dynamics of user interactions within the network. Through the application of social network analysis techniques, we calculated key metrics such as **betweenness centrality**, **closeness centrality**, **degree distribution**, **clustering coefficient** and **eigenvector centrality**. These metrics helped uncover patterns of connectivity, influential nodes, and community behavior.

Additionally, we visualized the network structure and explored the binary node classification problem, focusing on **discovering which attributes affect view count the most.** This process highlights the relationship between streamer interactions and network characteristics.

Overall, this project demonstrated the power of social network analysis tools, such as **Gephi**, in extracting meaningful insights from large datasets. These findings can inform further studies on social dynamics, content moderation, and community management in online platforms like **Twitch**.

References

1. **[1]** *Digital Music News*, "Top 0.5% of Twitch streamers generate 75% of views, data shows," Jul. 31, 2023. [Online]. Available: https://www.digitalmusicnews.com/2023/07/31/top-one-percent-twitch-streamers-generate-views/. [Accessed: Dec. 8, 2024].