Assignment 3

Gnanesh Patel MT22030
Hritika Shah MT22101
Nikita Kapoor 2020531

Question 1

Pick a real-world directed network dataset (with number of nodes > 100) from here. [2 points] Represent the network in terms of its 'adjacency matrix' as well as 'edge list'.
[28 points] Briefly describe the dataset chosen and report the following:
1. Number of Nodes
2. Number of Edges
3. Avg In-degree
4. Avg. Out-Degree
5. Node with Max In-degree
6. Node with Max out-degree
7. The density of the network

The dataset chosen for this assignment comes under the Internet peer-to-peer networks category and is called "p2p-Gnutella08". It is of directed type with 6,301 Nodes and 20,777 edges. It is the Gnutella peer to peer network from August 8, 2002.

1. The number of nodes is 6,301
2. The number of edges is 20,777
3. The average In-degree can be calculated by dividing the total number of incoming edges by the total number of nodes. The total number of incoming edges has been found to be 20,777.
   Thus, the average In-degree is = 20,777/6301 = 3.296

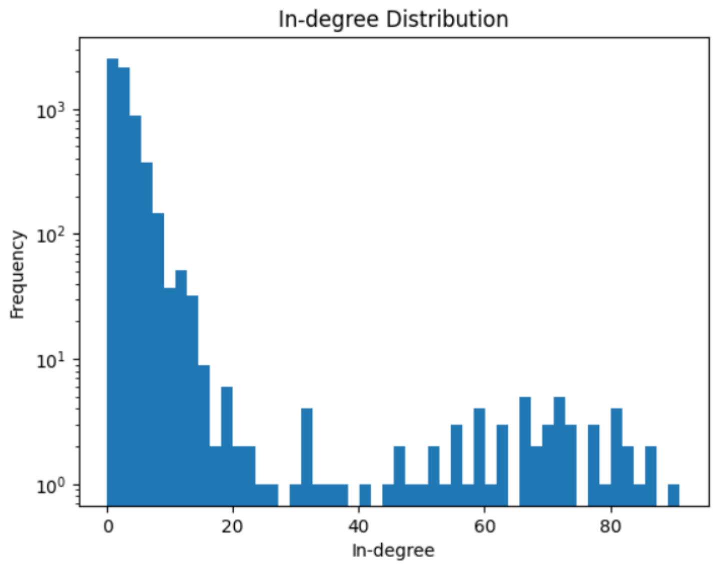
4. The average Out-degree can be calculated by dividing the total number of outgoing edges by the total number of nodes and is shown by the graph below.
   The total number of outgoing edges has been found to be 20,777.
   Thus, the average Out-degree is = 20,777/6301 = 3.296

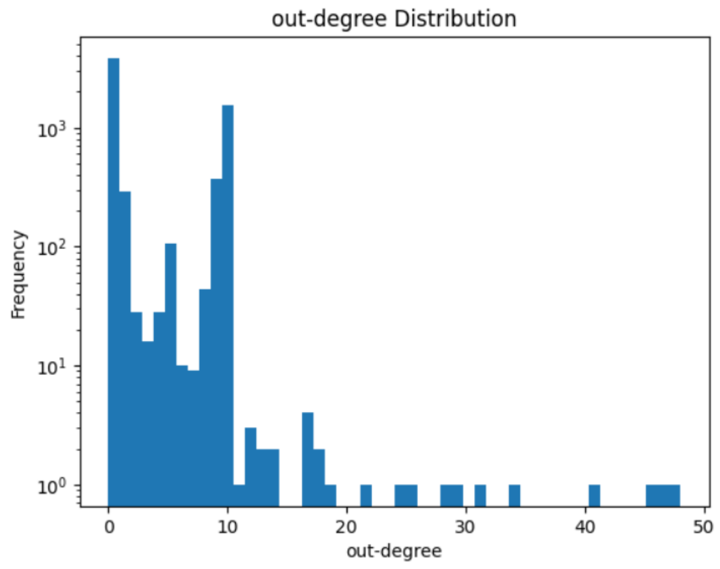
5. The node with the max in-degree has been found to be 266. It was computed by finding the maximum value from the dictionary formed using the dataset. We used the in_degree() function and further used the max() function for the same. Using this, the key was found with the maximum value.
6. The node with the max out-degree has been found to be 5831. It was computed by finding the maximum value from the dictionary formed using the dataset. We used the out_degree() function and further used the max() function for the same. Using this, the key was found with the maximum value.
7. The density of the network can be computed by obtaining the ratio of the number of edges to the maximum number of edges that are possible in a directed graph. It must have the same number of nodes.

Density = 20777/(6301^2-6301)
= 0.000554

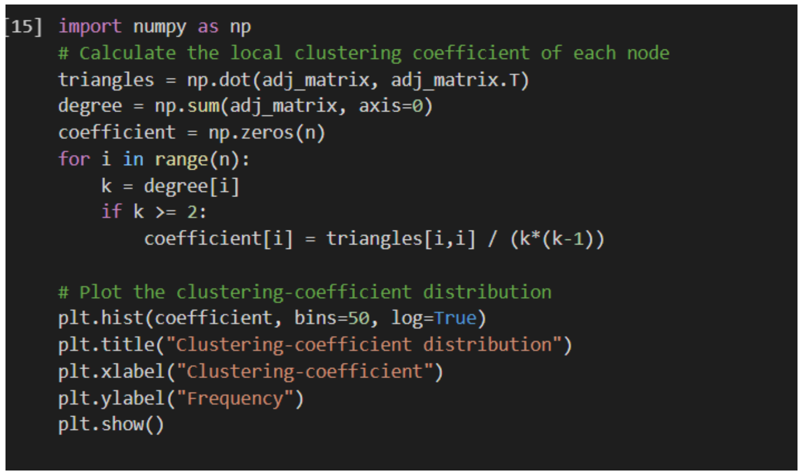
The following tasks have been performed:

1. **Plot degree distribution of the network (in case of a directed graph, plot in-degree and out-degree separately).**

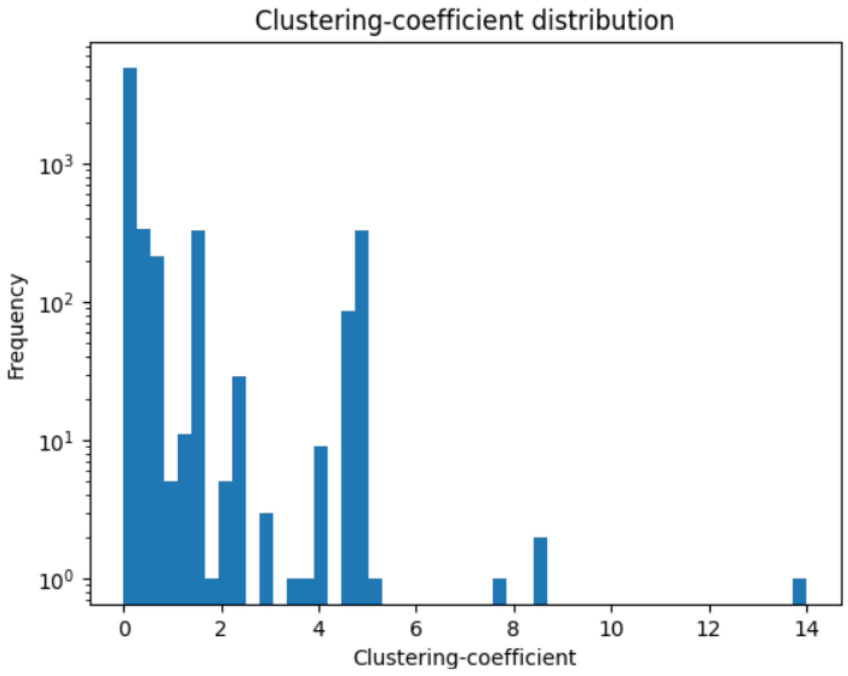


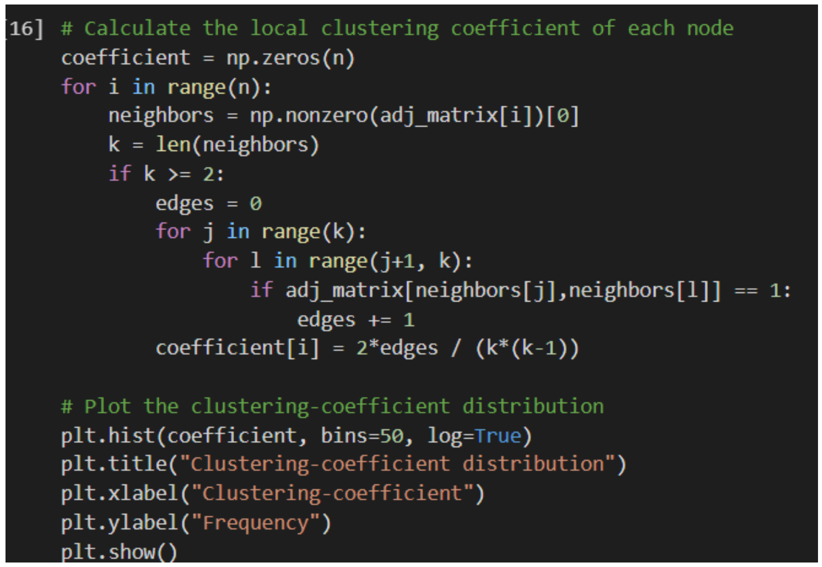


2. **Calculate the local clustering coefficient of each node and plot the clustering-coefficient distribution (lcc vs frequency of lcc) of the network.**

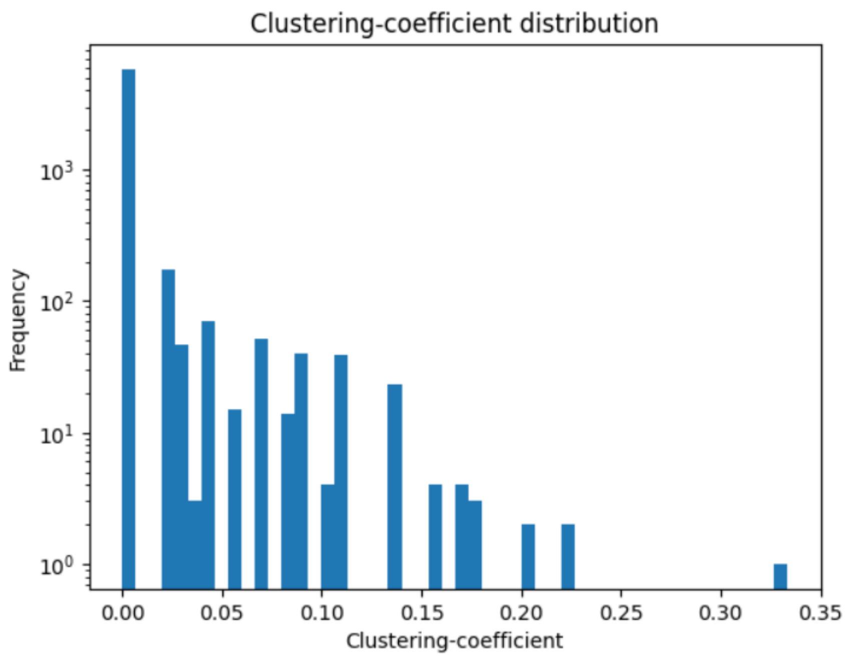


```
[15] import numpy as np
     # Calculate the local clustering coefficient of each node
     triangles = np.dot(adj_matrix, adj_matrix.T)
     degree = np.sum(adj_matrix, axis=0)
     coefficient = np.zeros(n)
     for i in range(n):
         k = degree[i]
         if k >= 2:
             coefficient[i] = triangles[i,i] / (k*(k-1))

     # Plot the clustering-coefficient distribution
     plt.hist(coefficient, bins=50, log=True)
     plt.title("Clustering-coefficient distribution")
     plt.xlabel("Clustering-coefficient")
     plt.ylabel("Frequency")
     plt.show()
```

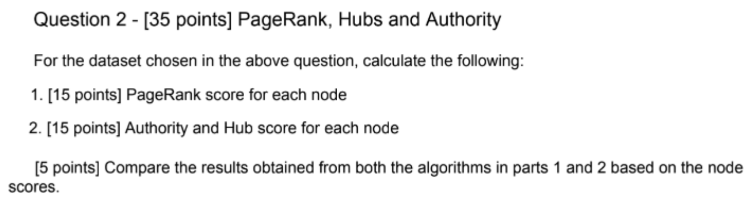




```
[16] # Calculate the local clustering coefficient of each node
     coefficient = np.zeros(n)
     for i in range(n):
         neighbors = np.nonzero(adj_matrix[i])[0]
         k = len(neighbors)
         if k >= 2:
             edges = 0
             for j in range(k):
                 for l in range(j+1, k):
                     if adj_matrix[neighbors[j],neighbors[l]] == 1:
                         edges += 1
             coefficient[i] = 2*edges / (k*(k-1))

     # Plot the clustering-coefficient distribution
     plt.hist(coefficient, bins=50, log=True)
     plt.title("Clustering-coefficient distribution")
     plt.xlabel("Clustering-coefficient")
     plt.ylabel("Frequency")
     plt.show()
```



Question 2

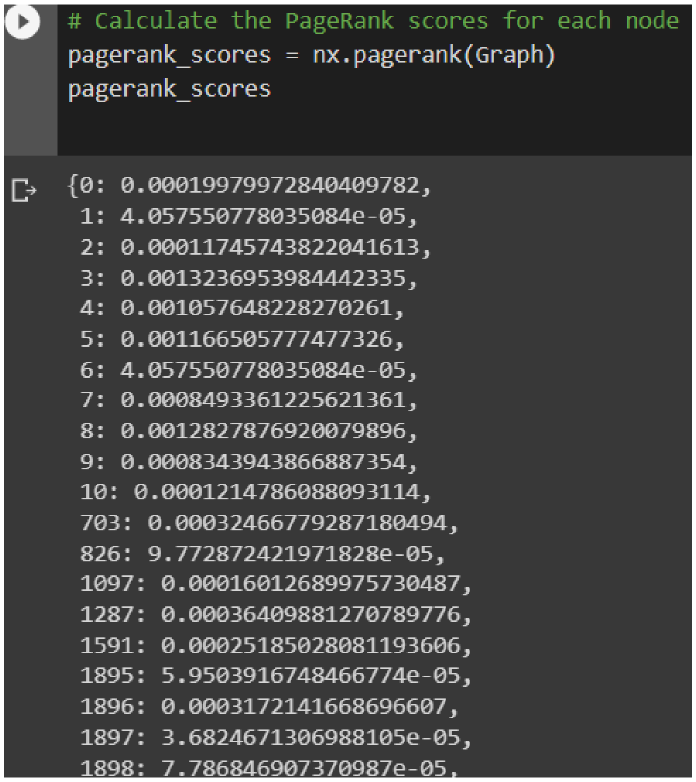

Question 2 - [35 points] PageRank, Hubs and Authority

For the dataset chosen in the above question, calculate the following:

1. [15 points] PageRank score for each node
2. [15 points] Authority and Hub score for each node

[5 points] Compare the results obtained from both the algorithms in parts 1 and 2 based on the node scores.

1. PageRank is an algorithm that assigns a numerical weight to each element of a set of links or web pages and measures its importance within the set. In the dataset we have used, there are 6301 nodes and 20777 edges. The pagerank score was calculated for all the nodes. The below screenshot displays a screenshot of a part of the nodes that were printed by the system.



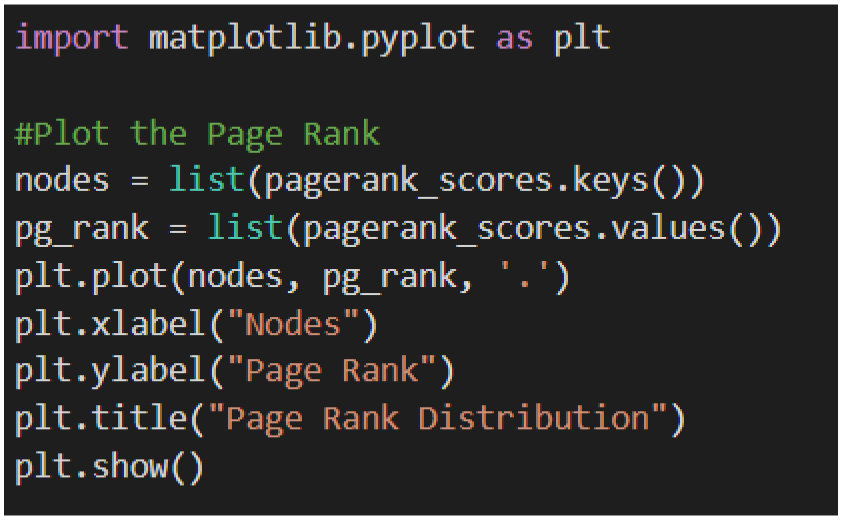
```
# Calculate the PageRank scores for each node
pagerank_scores = nx.pagerank(Graph)
pagerank_scores
```

```
{0: 0.00019979972840409782,
 1: 4.057550778035084e-05,
 2: 0.00011745743822041613,
 3: 0.0013236953984442335,
 4: 0.001057648228270261,
 5: 0.001166505777477326,
 6: 4.057550778035084e-05,
 7: 0.0008493361225621361,
 8: 0.0012827876920079896,
 9: 0.0008343943866887354,
 10: 0.0001214786088093114,
 703: 0.00032466779287180494,
 826: 9.772872421971828e-05,
 1097: 0.00016012689975730487,
 1287: 0.00036409881270789776,
 1591: 0.00025185028081193606,
 1895: 5.9503916748466774e-05,
 1896: 0.0003172141668696607,
 1897: 3.6824671306988105e-05,
 1898: 7.786846907370987e-05,
```

https://docs.google.com/document/d/1RyqAlQWEOKXTRyQAqLC-0K3XgXIGkNDzSXsXKM6sgfs/edit?usp=sharing

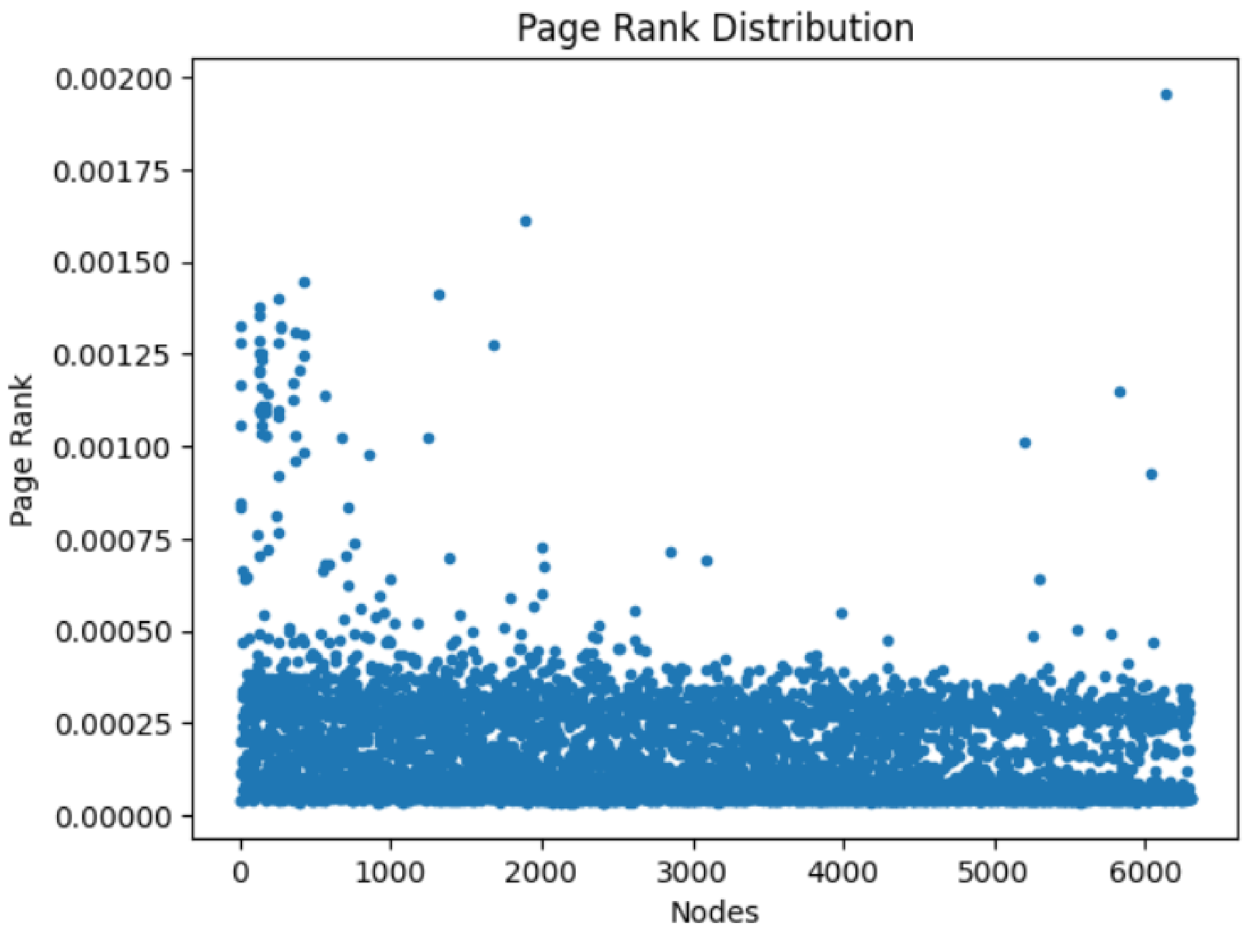
The nx.pagerank() function used in the above, computes the PageRank scores of the nodes in the graph given as input. The graph in this case is a directed graph. The function returns a dictionary mapping each node in the graph to its corresponding PageRank score.
The values always lie between 0 and 1. A higher value closer to 1 shows that it holds more importance in its set.

Matplotlib has been used to plot the PageRank distribution. The code is as given below:

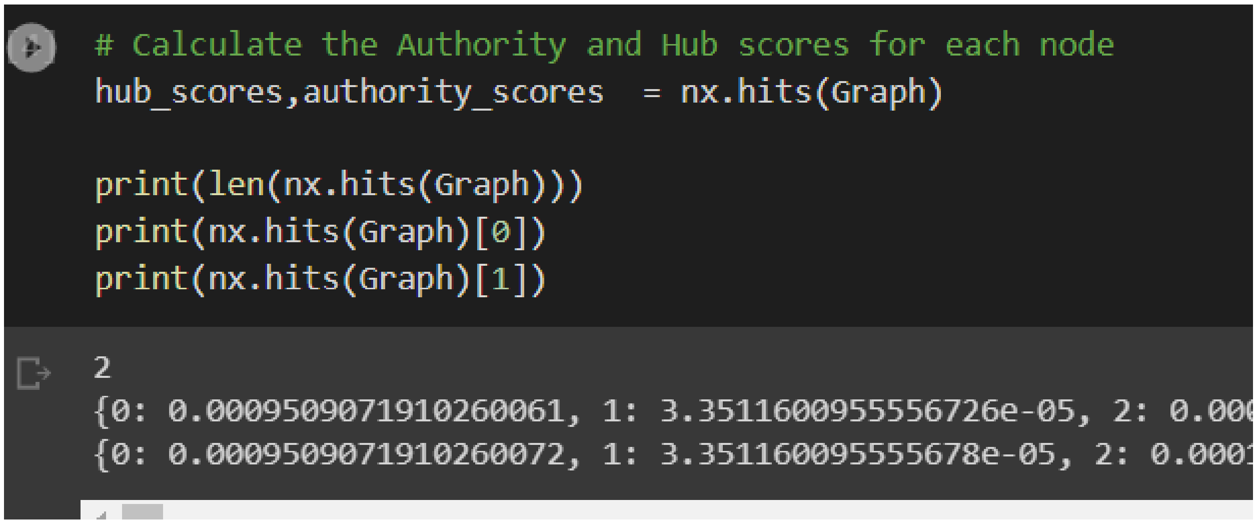


```
import matplotlib.pyplot as plt

#Plot the Page Rank
nodes = list(pagerank_scores.keys())
pg_rank = list(pagerank_scores.values())
plt.plot(nodes, pg_rank, '.')
plt.xlabel("Nodes")
plt.ylabel("Page Rank")
plt.title("Page Rank Distribution")
plt.show()
```

The plot for the same is given below:

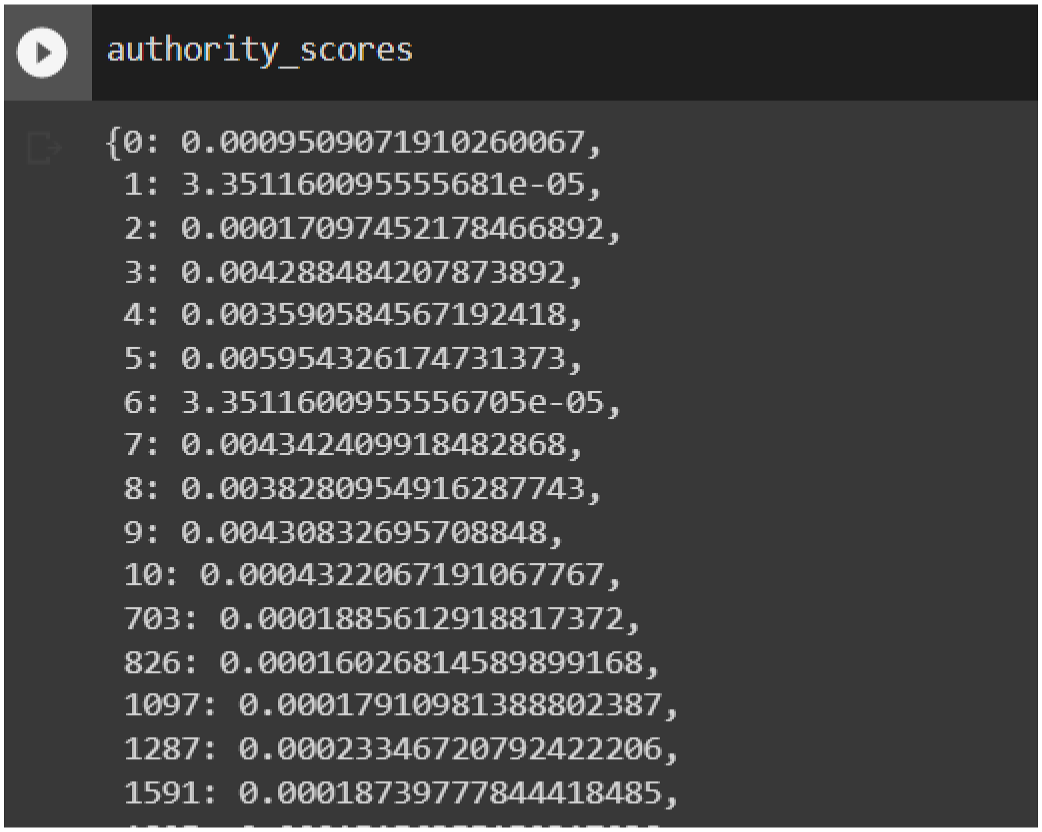


2. Authority and Hub scores need to be calculated for each of the nodes. These are calculated by HITS, which ranks web pages based on their importance. The authority score of a web page is said to be a measure of its importance or relevance as a destination page. It may or may not be linked through other pages. A high authority score indicates that the same would be linked by several other pages. The Hub score denotes the importance of a page that links to other pages. A high score indicates that it would link to several other pages.



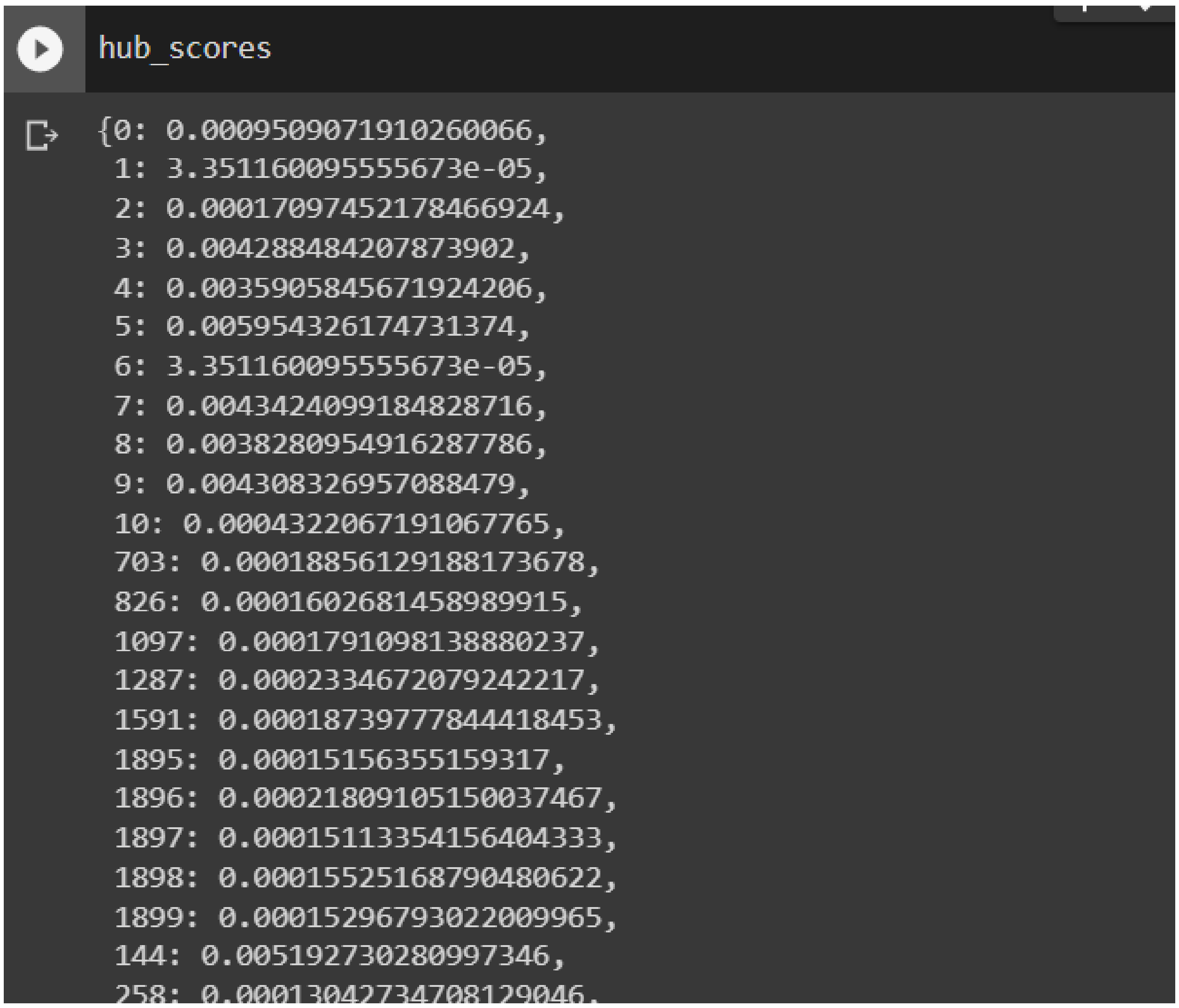
```
# Calculate the Authority and Hub scores for each node
hub_scores,authority_scores  = nx.hits(Graph)

print(len(nx.hits(Graph)))
print(nx.hits(Graph)[0])
print(nx.hits(Graph)[1])
```

```
2
{0: 0.0009509071910260061, 1: 3.3511600955556726e-05, 2: 0.000
{0: 0.0009509071910260072, 1: 3.351160095555678e-05, 2: 0.0001
```



```
authority_scores
```

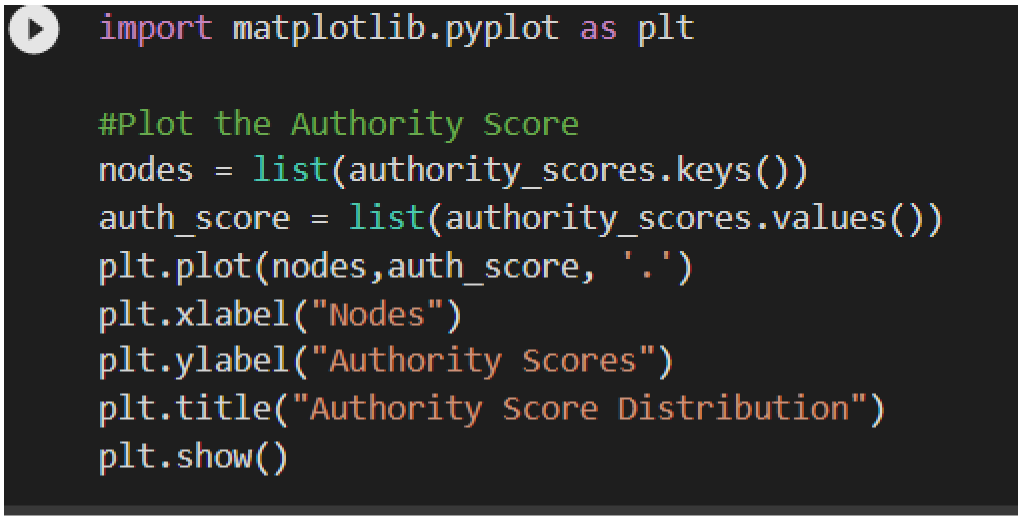
```
{0: 0.0009509071910260067,
 1: 3.351160095555681e-05,
 2: 0.00017097452178466892,
 3: 0.004288484207873892,
 4: 0.003590584567192418,
 5: 0.005954326174731373,
 6: 3.3511600955556705e-05,
 7: 0.004342409918482868,
 8: 0.0038280954916287743,
 9: 0.004308326957088848,
 10: 0.00043220671910677 67,
 703: 0.00018856129188173 72,
 826: 0.00016026814589899168,
 1097: 0.00017910981388802387,
 1287: 0.00023346720792422206,
 1591: 0.00018739777844418485,
```



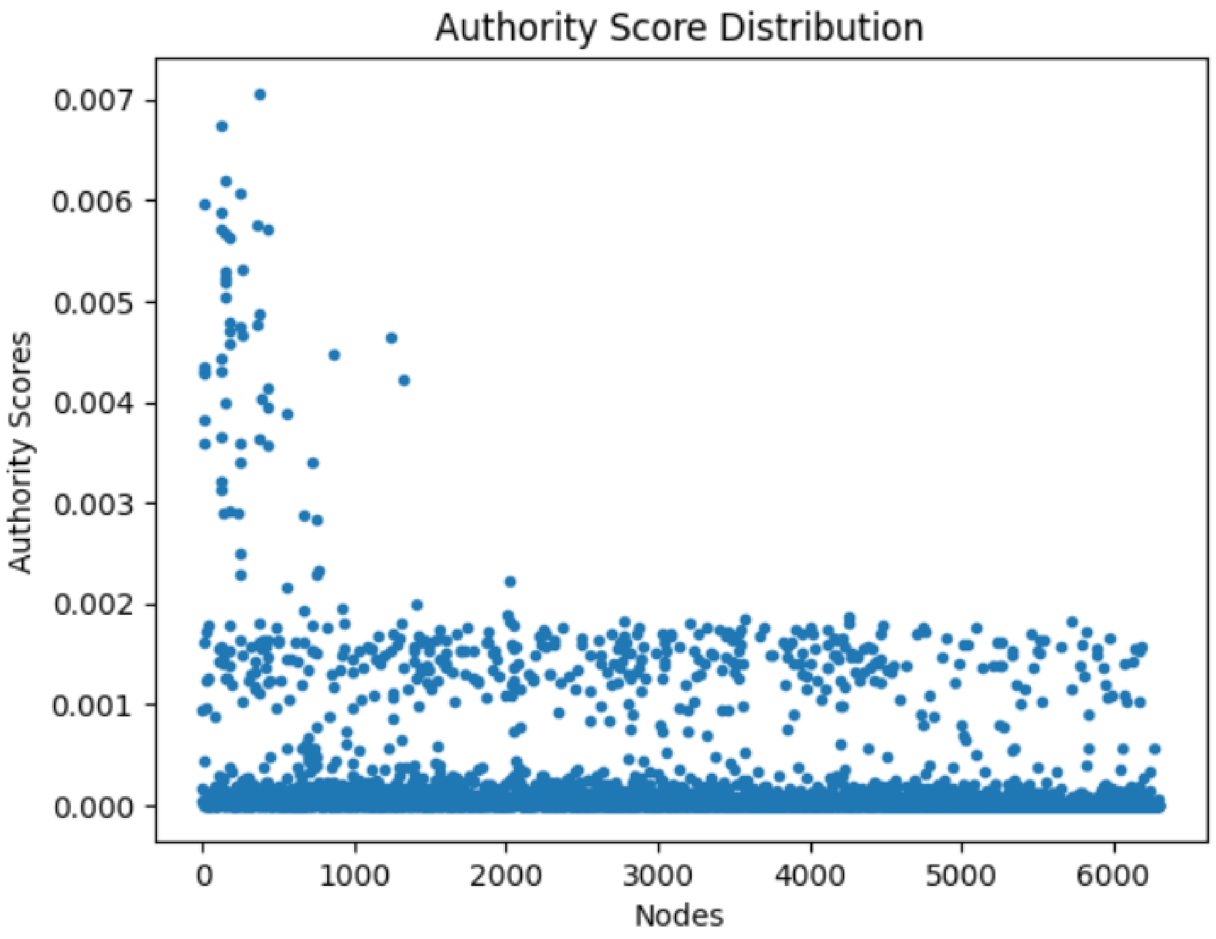
```
hub_scores
```

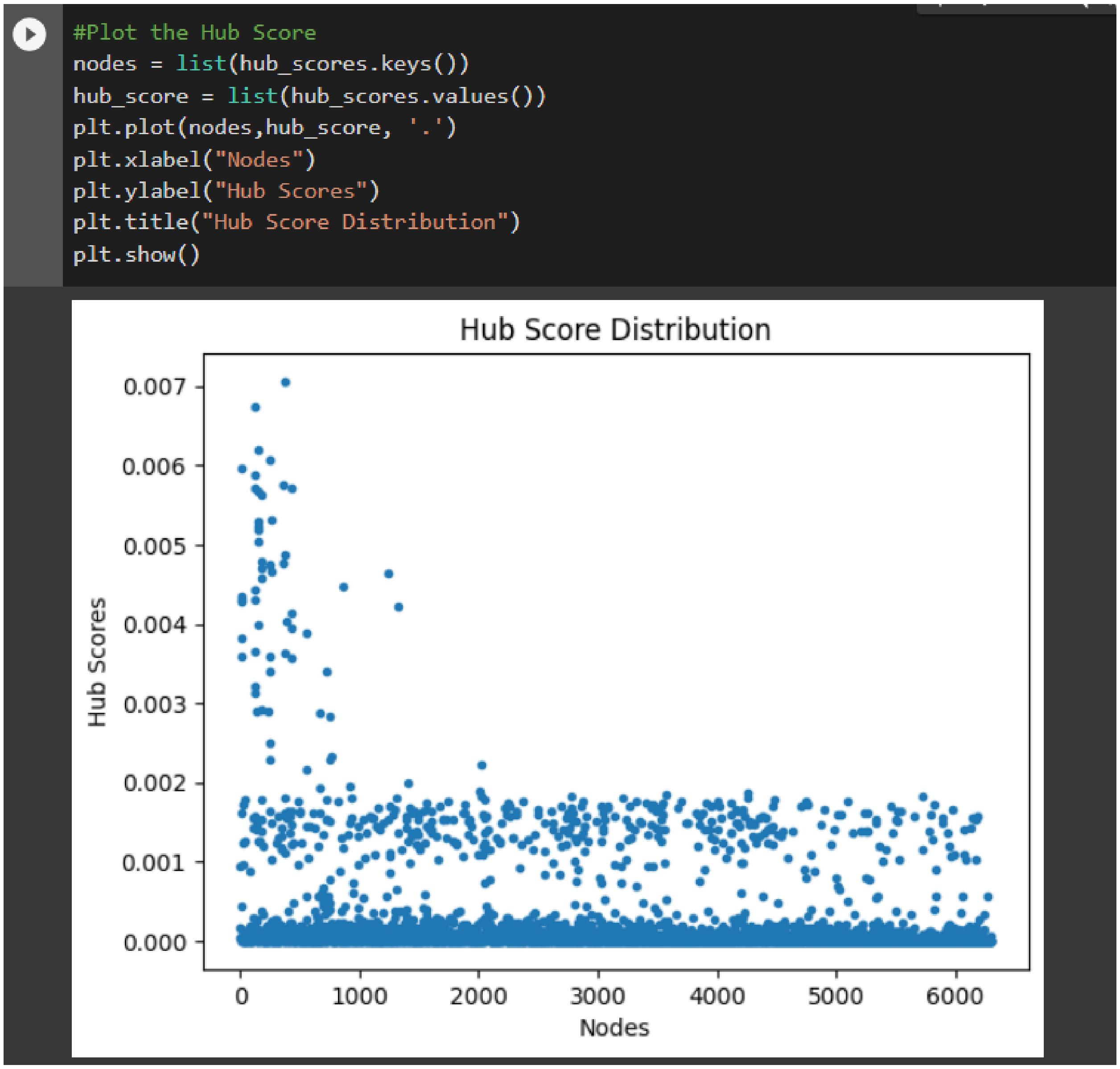
```
{0: 0.0009509071910260066,
 1: 3.351160095555673e-05,
 2: 0.00017097452178466924,
 3: 0.004288484207873902,
 4: 0.0035905845671924206,
 5: 0.005954326174731374,
 6: 3.351160095555673e-05,
 7: 0.0043424099184828716,
 8: 0.0038280954916287786,
 9: 0.004308326957088479,
 10: 0.0004322067191067765,
 703: 0.00018856129188173678,
 826: 0.0001602681458989915,
 1097: 0.0001791098138880237,
 1287: 0.0002334672079242217,
 1591: 0.00018739777844418453,
 1895: 0.00015156355159317,
 1896: 0.00021809105150037467,
 1897: 0.00015113354156404333,
 1898: 0.00015525168790480622,
 1899: 0.00015296793022009965,
 144: 0.005192730280997346,
 258: 0.00013042734708129046,
```

The distributions for the above scores have been plotted using matplotlib and are as follows with their respective codes:

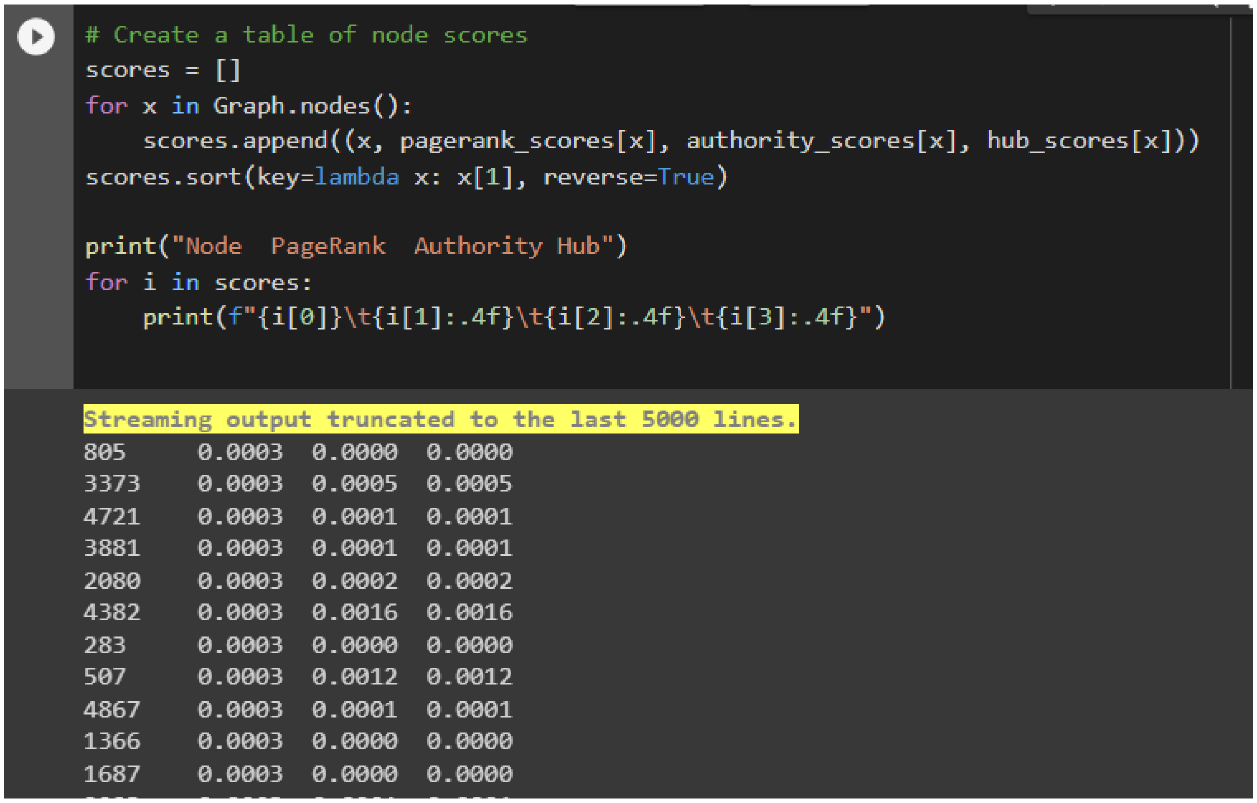


```python
import matplotlib.pyplot as plt

#Plot the Authority Score
nodes = list(authority_scores.keys())
auth_score = list(authority_scores.values())
plt.plot(nodes,auth_score, '.')
plt.xlabel("Nodes")
plt.ylabel("Authority Scores")
plt.title("Authority Score Distribution")
plt.show()
```


Authority Score Distribution
Authority Scores
Nodes
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0
1000
2000
3000
4000
5000
6000



```
#Plot the Hub Score
nodes = list(hub_scores.keys())
hub_score = list(hub_scores.values())
plt.plot(nodes,hub_score, '.')
plt.xlabel("Nodes")
plt.ylabel("Hub Scores")
plt.title("Hub Score Distribution")
plt.show()
```



```
# Create a table of node scores
scores = []
for x in Graph.nodes():
    scores.append((x, pagerank_scores[x], authority_scores[x], hub_scores[x]))
scores.sort(key=lambda x: x[1], reverse=True)

print("Node  PageRank  Authority Hub")
for i in scores:
    print(f"{i[0]}\t{i[1]:.4f}\t{i[2]:.4f}\t{i[3]:.4f}")
```

```
Streaming output truncated to the last 5000 lines.
805     0.0003  0.0000  0.0000
3373    0.0003  0.0005  0.0005
4721    0.0003  0.0001  0.0001
3881    0.0003  0.0001  0.0001
2080    0.0003  0.0002  0.0002
4382    0.0003  0.0016  0.0016
283     0.0003  0.0000  0.0000
507     0.0003  0.0012  0.0012
4867    0.0003  0.0001  0.0001
1366    0.0003  0.0000  0.0000
1687    0.0003  0.0000  0.0000
```

Please note that due to the large amount of data in these sets, the whole cannot be effectively displayed. In order to cross-check the data it needs to be done manually by running the codes that have been submitted for the assignment.