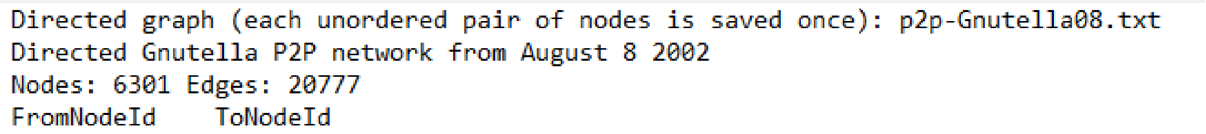
Report

IR Assignment -3

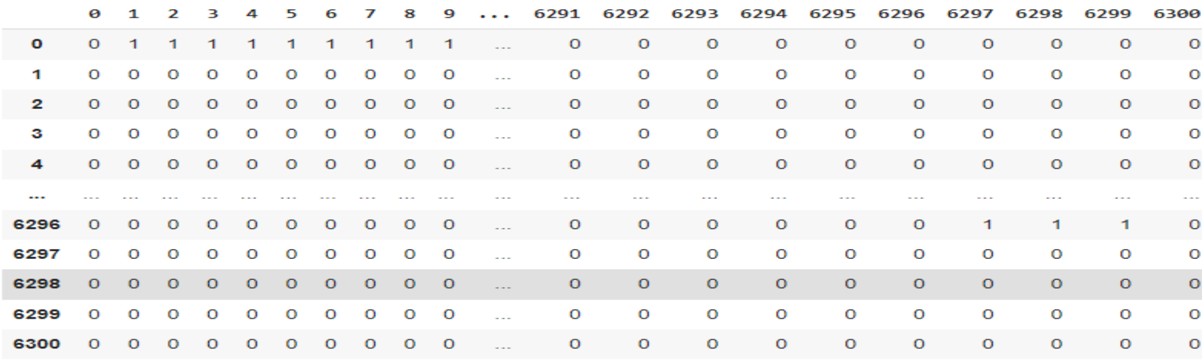
Group No - 80

Q1.For 1st part, we had chosen a database from https://snap.stanford.edu/data/index.html named p2p-Gunetella08.txt.It is a directed graph with 6301 nodes and 20777 edges. The first four lines in p2p-Gunetella08.txt contain comments so we had removed them.



```
Directed graph (each unordered pair of nodes is saved once): p2p-Gnutella08.txt
Directed Gnutella P2P network from August 8 2002
Nodes: 6301 Edges: 20777
FromNodeId    ToNodeId
```

By using the count variable and checking for the condition if the count is greater than four.Than we split the parts where there is space so as to separate starting and ending nodes. After converting into integers,len() is used to calculate the number of nodes and edges.Then we had converted it into the matrix as asked in the question.



	0	1	2	3	4	5	6	7	8	9	...	6291	6292	6293	6294	6295	6296	6297	6298	6299	6300
0	0	1	1	1	1	1	1	1	1	1	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6296	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	1	1	0
6297	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6298	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6299	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6300	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Than we had created edge list.`print("Edge List:"`.

Outdegree and indegree of node has been computed using the formula :`Out={}`

```
for i in list(matrix.keys()):
  Out[i]=len(matrix[i])
```

And thus created the dictionary.To calculate average outdegree we sum all the outdegree and divided it by total number of nodes.To calculate maximum outdegree we had used the basic formula:

```
maxout=-32767
```

`if(maxout<=Deg_Out[i])` and spotted the node which contains maximum outdegree and assigned it. The density of a graph represents the ratio between the edges present in a graph and the maximum number of edges that the graph can contain. So we calculated it using the formula in collab file.

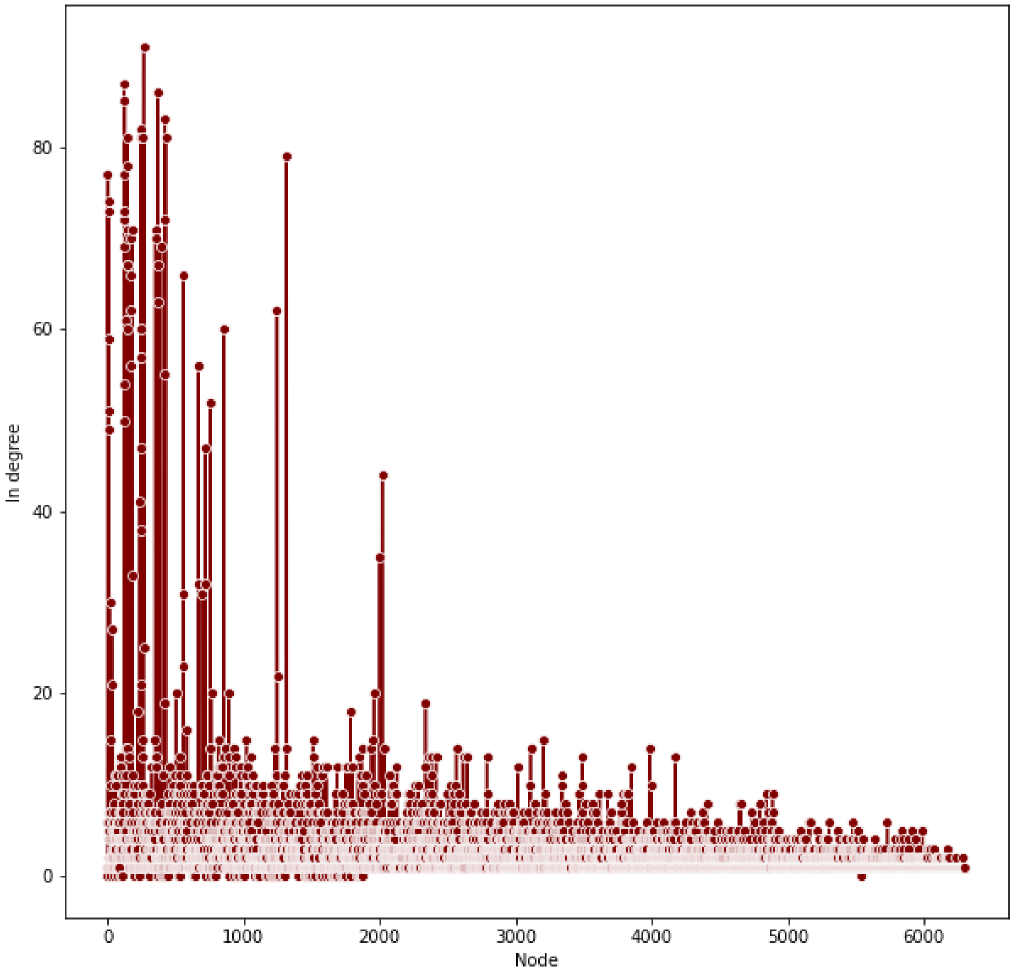
```
a=len(edge)
ttledges=0
ttledges=(len(node)*(len(node)-1))
b=ttledges
density=a/b
```

```
total Nodes present in the graph are: 6301
total edges present in the graph are: 20777
Avg OutDegree is: 3.2974131090303125
max out degree: 48
node with maximum outdegree: 5831
Avg InDegree is: 3.2974131090303125
maximum indegree: 91
node with maximum indegree: 266
The density of the network is: 0.0005233989061952878.
```

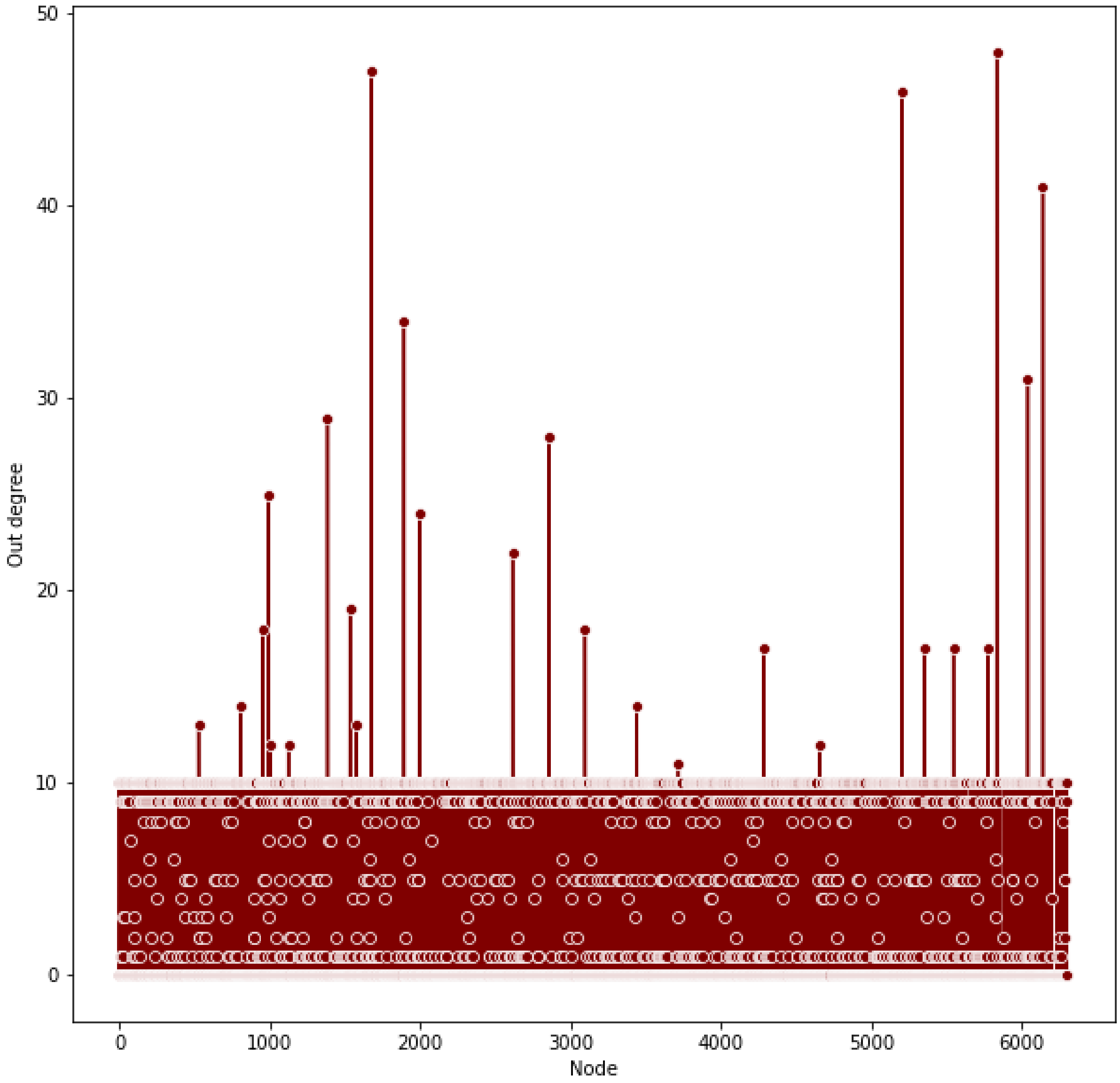
To plot the graph line chart has been used for both indegree and outdegree.

In graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together.

Plot for Indegree of graph

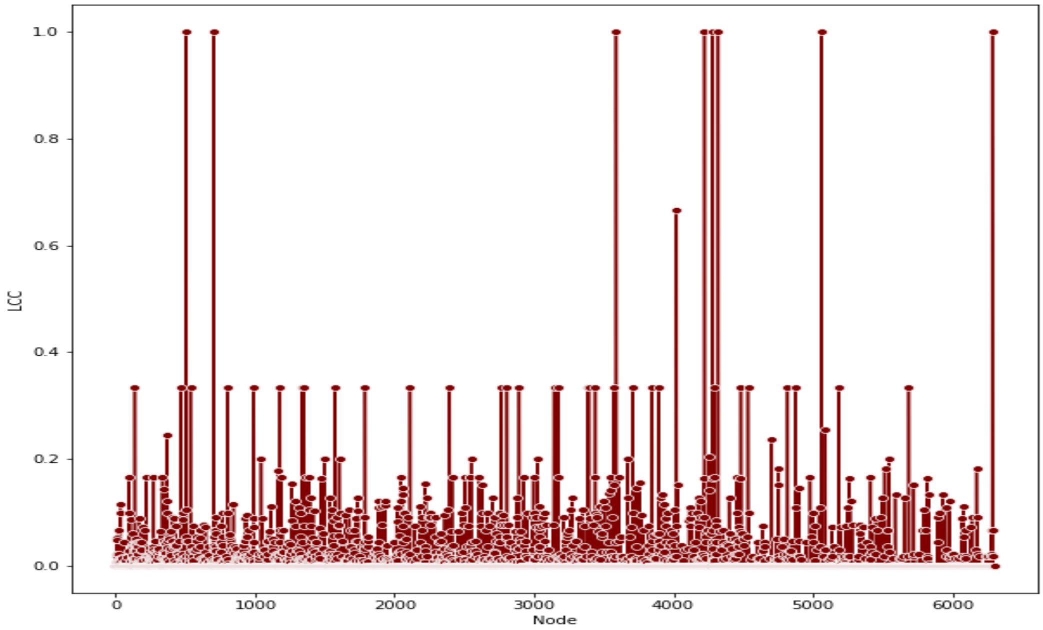


Plot for outdegree of graph

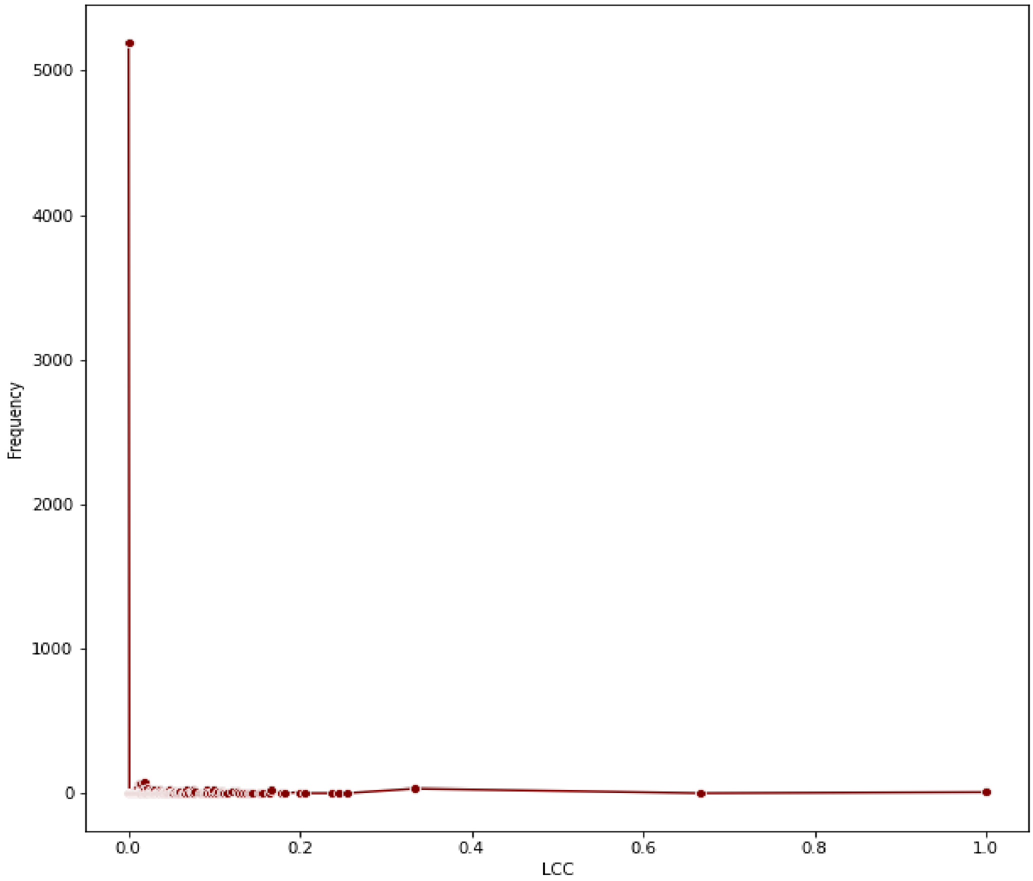


Than we had calculated Local clustering coefficient.It is calculated using the formula no of edges present/total no of edges possible from the nodes.Checked for outgoing edges and for that we had checked node,if outgoing node is there,update counter and zero.After applying formula local clustering coefficient is calculated and graphs are drawn.

Plot for clustering-coefficient distribution of the network.



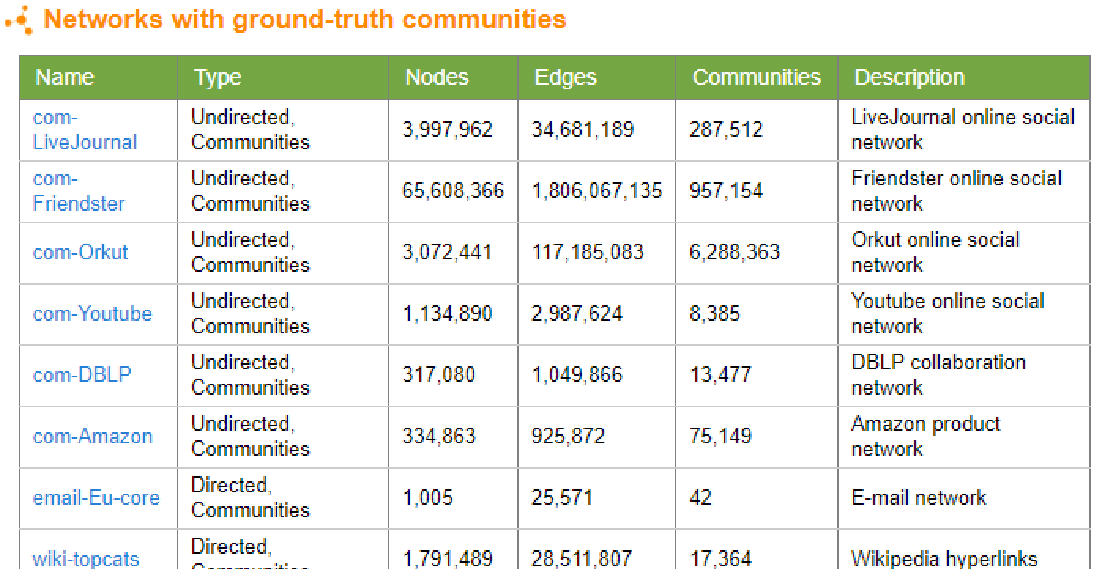
Plot for Frequency vs LCC



Q2.

For question 2 we have used the following data set

email-EU-core



Networks with ground-truth communities

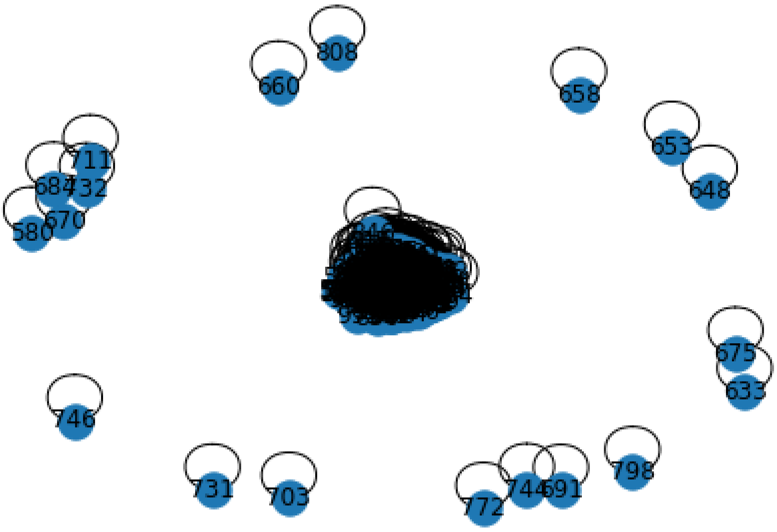
Name	Type	Nodes	Edges	Communities	Description
com-LiveJournal	Undirected, Communities	3,997,962	34,681,189	287,512	LiveJournal online social network
com-Friendster	Undirected, Communities	65,608,366	1,806,067,135	957,154	Friendster online social network
com-Orkut	Undirected, Communities	3,072,441	117,185,083	6,288,363	Orkut online social network
com-Youtube	Undirected, Communities	1,134,890	2,987,624	8,385	Youtube online social network
com-DBLP	Undirected, Communities	317,080	1,049,866	13,477	DBLP collaboration network
com-Amazon	Undirected, Communities	334,863	925,872	75,149	Amazon product network
email-Eu-core	Directed, Communities	1,005	25,571	42	E-mail network
wiki-topcats	Directed,	1,791,489	28,511,807	17,364	Wikipedia hyperlinks

It contains 1005 nodes,25,571 edges, and 42 communities.

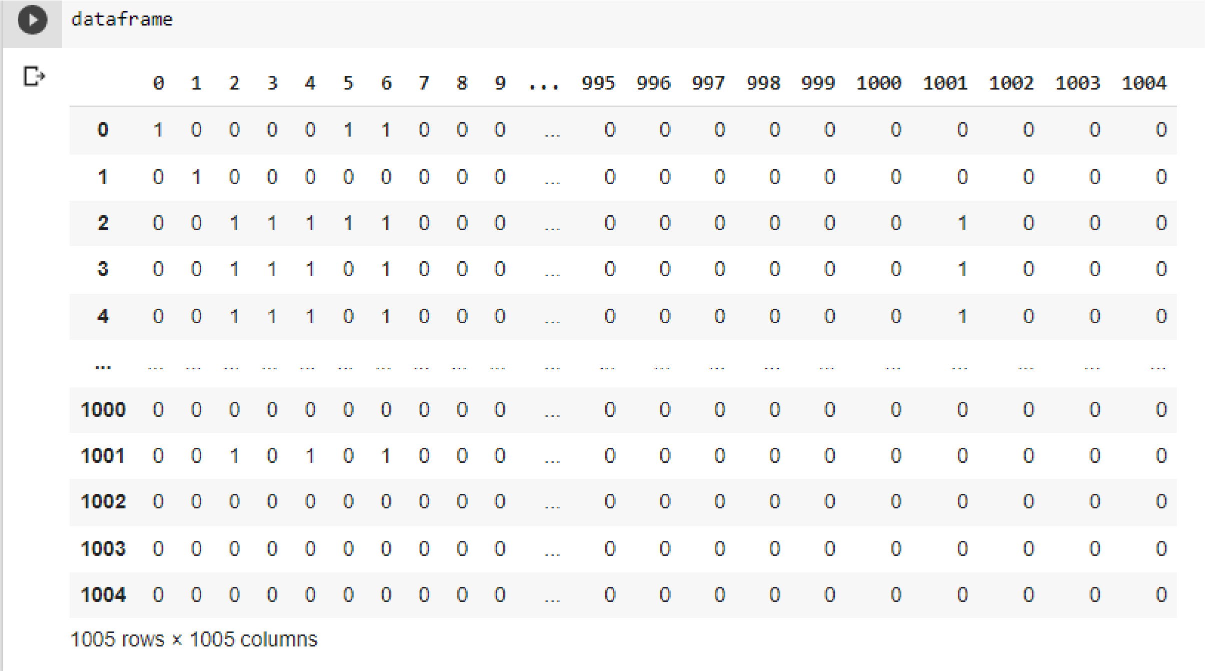
This question was based on **PageRank, Hubs, and Authority** there are two-part in this question

Part 1 Implement algorithm from scratch to page rank score for each node

So first of all we visualise the dataset using `networkx and  pyplot` graph shown below



Then calculate indegree and outdegree for each node by writing the indegree and outdegree function and store in a data frame
Where the index is the source node and column in the destination node if there edge between source to destination then that particular cell contain 1 otherwise cell contain 0

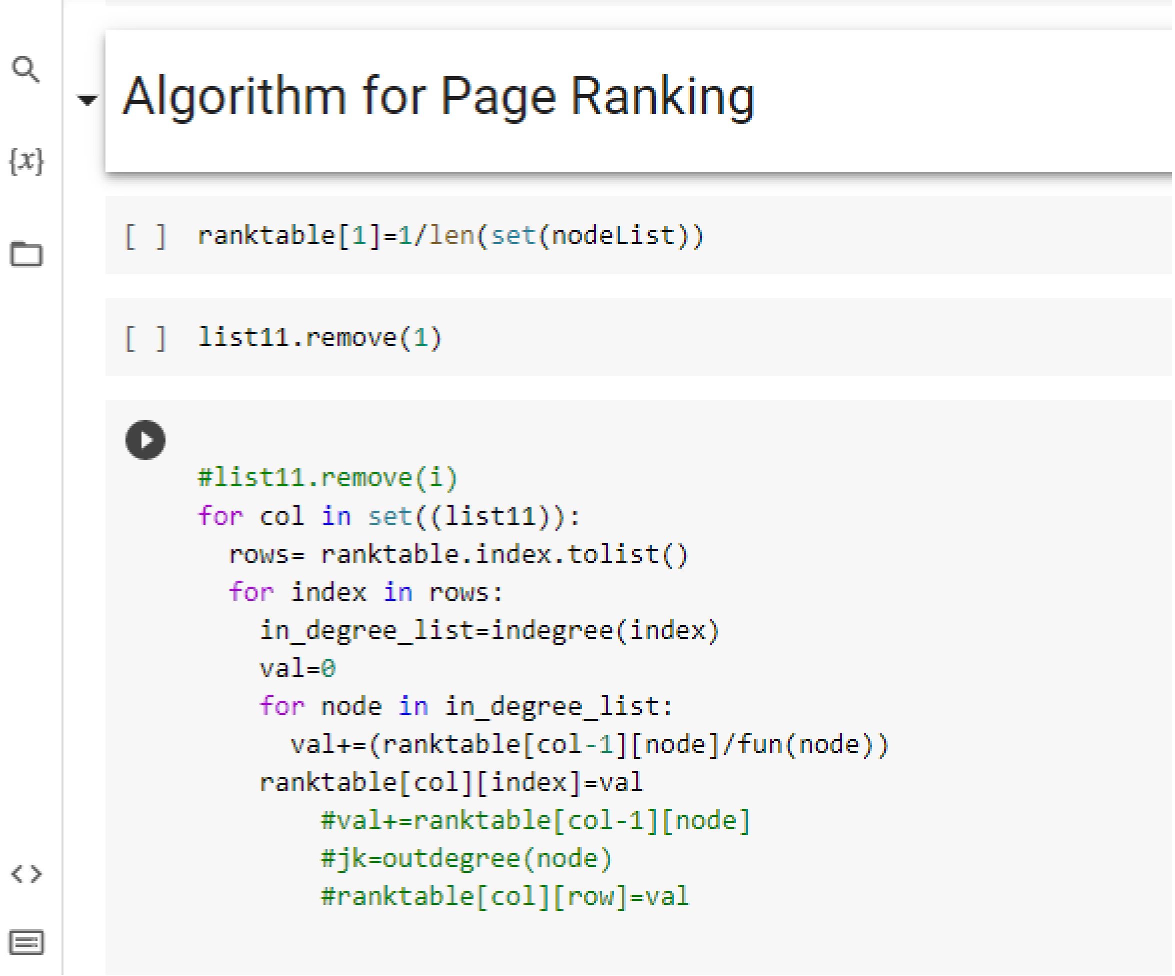


```
dataframe
```

	0	1	2	3	4	5	6	7	8	9	...	995	996	997	998	999	1000	1001	1002	1003	1004
0	1	0	0	0	0	1	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	1	1	1	1	1	0	0	0	...	0	0	0	0	0	0	1	0	0	0
3	0	0	1	1	1	0	1	0	0	0	...	0	0	0	0	0	0	1	0	0	0
4	0	0	1	1	1	0	1	0	0	0	...	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1000	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1001	0	0	1	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1002	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1003	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1004	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1005 rows × 1005 columns

Page rank algorithm



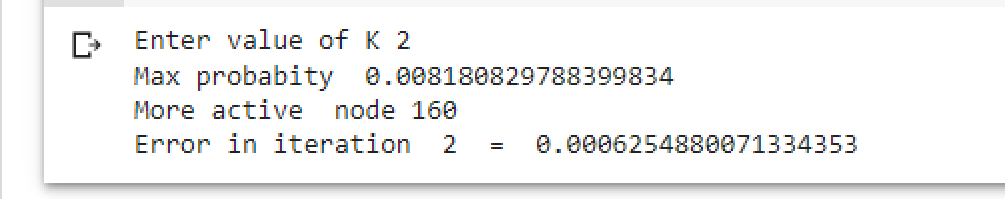
Algorithm for Page Ranking

```
ranktable[1]=1/len(set(nodeList))
```

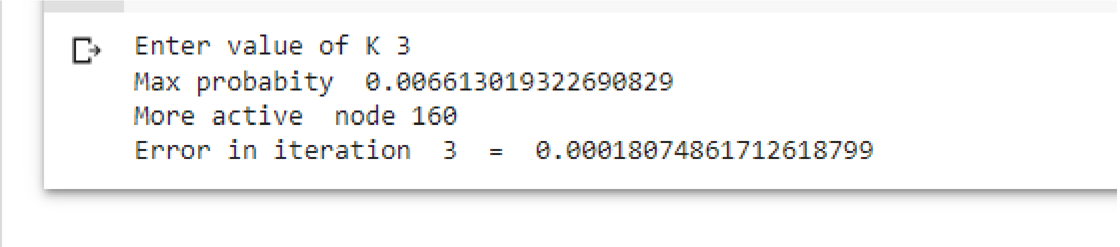
```
list11.remove(1)
```

```
#list11.remove(i)
for col in set((list11)):
  rows= ranktable.index.tolist()
  for index in rows:
    in_degree_list=indegree(index)
    val=0
    for node in in_degree_list:
      val+=(ranktable[col-1][node]/fun(node))
    ranktable[col][index]=val
        #val+=ranktable[col-1][node]
        #jk=outdegree(node)
        #ranktable[col][row]=val
```

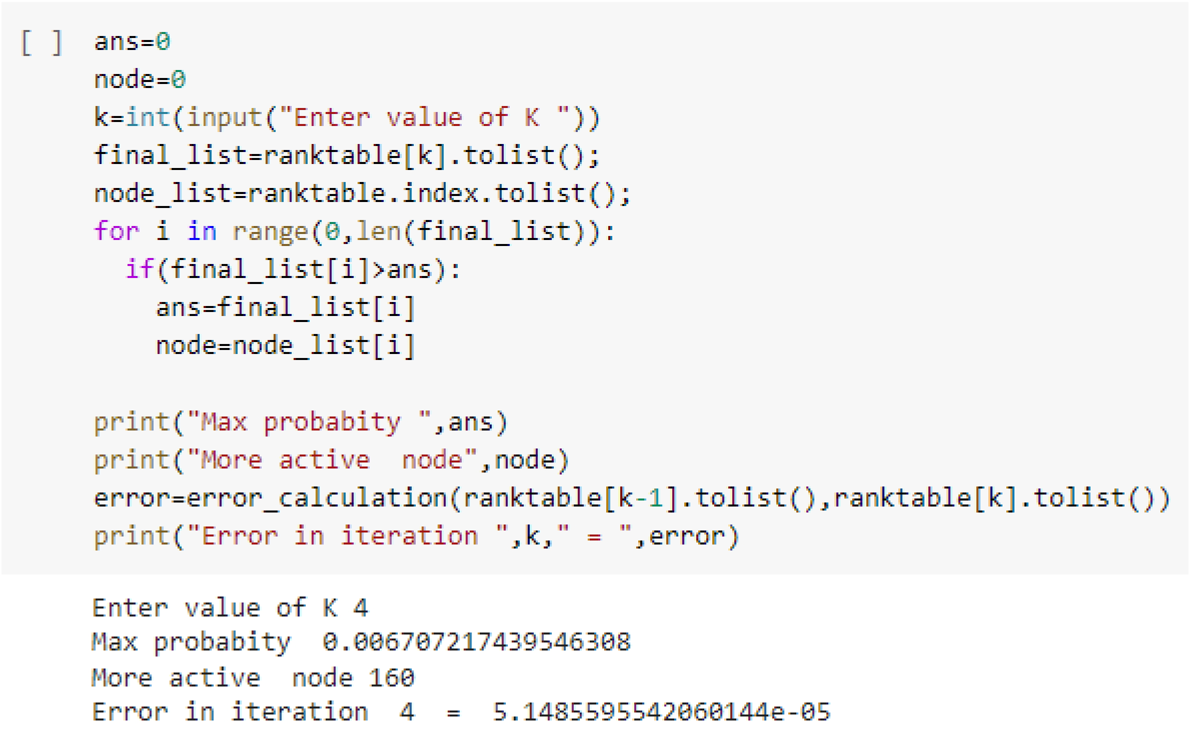
Result for 1st part



```
Enter value of K 2
Max probabity  0.008180829788399834
More active  node 160
Error in iteration  2  =  0.0006254880071334353
```



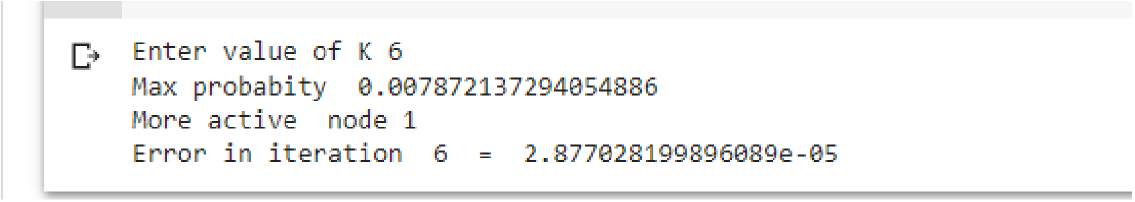
```
Enter value of K 3
Max probabity  0.006613019322690829
More active  node 160
Error in iteration  3  =  0.00018074861712618799
```



```
[ ]  ans=0
     node=0
     k=int(input("Enter value of K "))
     final_list=ranktable[k].tolist();
     node_list=ranktable.index.tolist();
     for i in range(0,len(final_list)):
       if(final_list[i]>ans):
         ans=final_list[i]
         node=node_list[i]

     print("Max probabity ",ans)
     print("More active  node",node)
     error=error_calculation(ranktable[k-1].tolist(),ranktable[k].tolist())
     print("Error in iteration ",k," = ",error)
```

```
Enter value of K 4
Max probabity  0.006707217439546308
More active  node 160
Error in iteration  4  =  5.148559554206014e-05
```



```
Enter value of K 6
Max probabity  0.007872137294054886
More active  node 1
Error in iteration  6  =  2.877028199896089e-05
```

Part -2 Authority and Hub score for each node

Here we have used the same table of indegree and outdegree that in 1st part.

Assumption: here initial weight of the hub is not given so i take it as vector of 1's

Then take Transpose of a matrix (that contains indegree and outdegree of graph nodes)
Then used the following formula to calculate the Authority weight vector

# Authority weight vector V=A^T*U

And to calculate Hub weight vector

Hub vector U =A*V

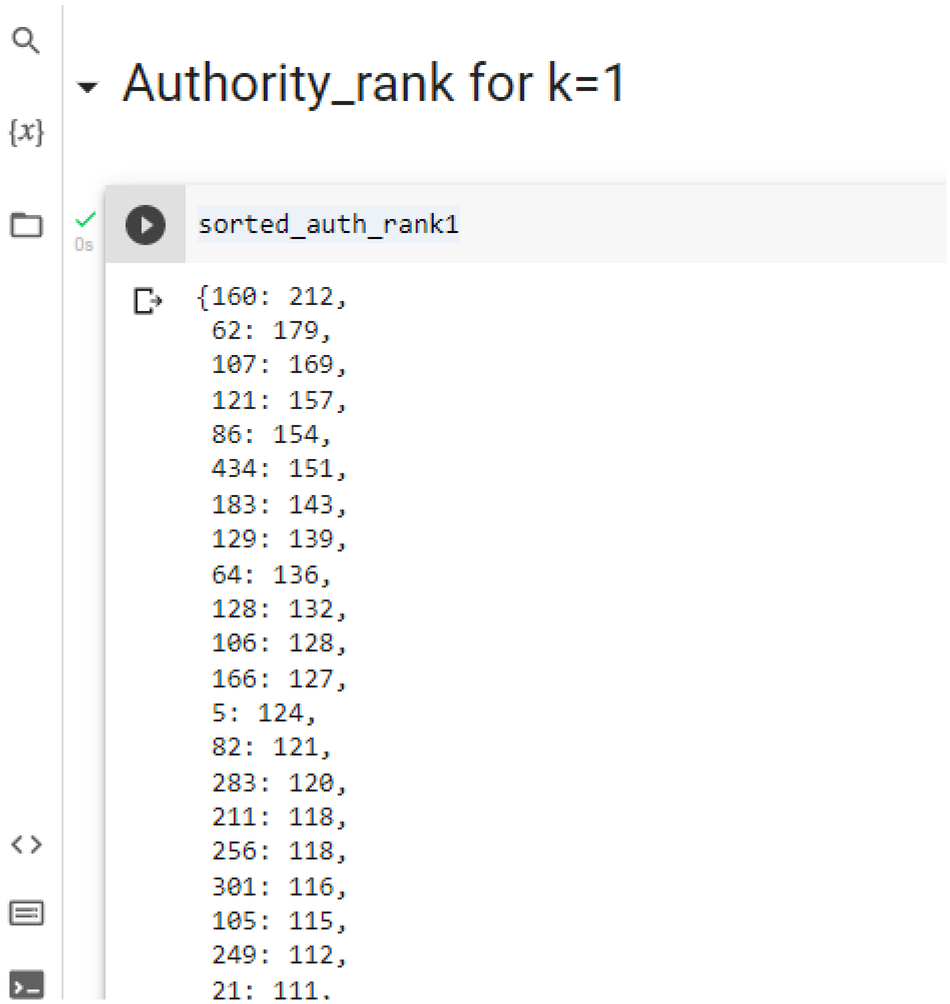
A:- Original graph matrix maintains in-degree and out-degree of graph
V:-Authority vector

U:- Hub vector

Logic:

Here 1st we take the transpose of the matrix and multiply it with the vector(Hub vector) and get Authority
Weight vector then multiple matrix and Authority weight vector to calculate Hub vector
Then create a dictionary in which nodes as a key and value as a score the sort the dictionary in reverse order and the node in 1st position has the best score.
For k=2 and so on to calculate the score we first sum the vector of the previous value of k and then divide all individual Val by its root value. and again maintain

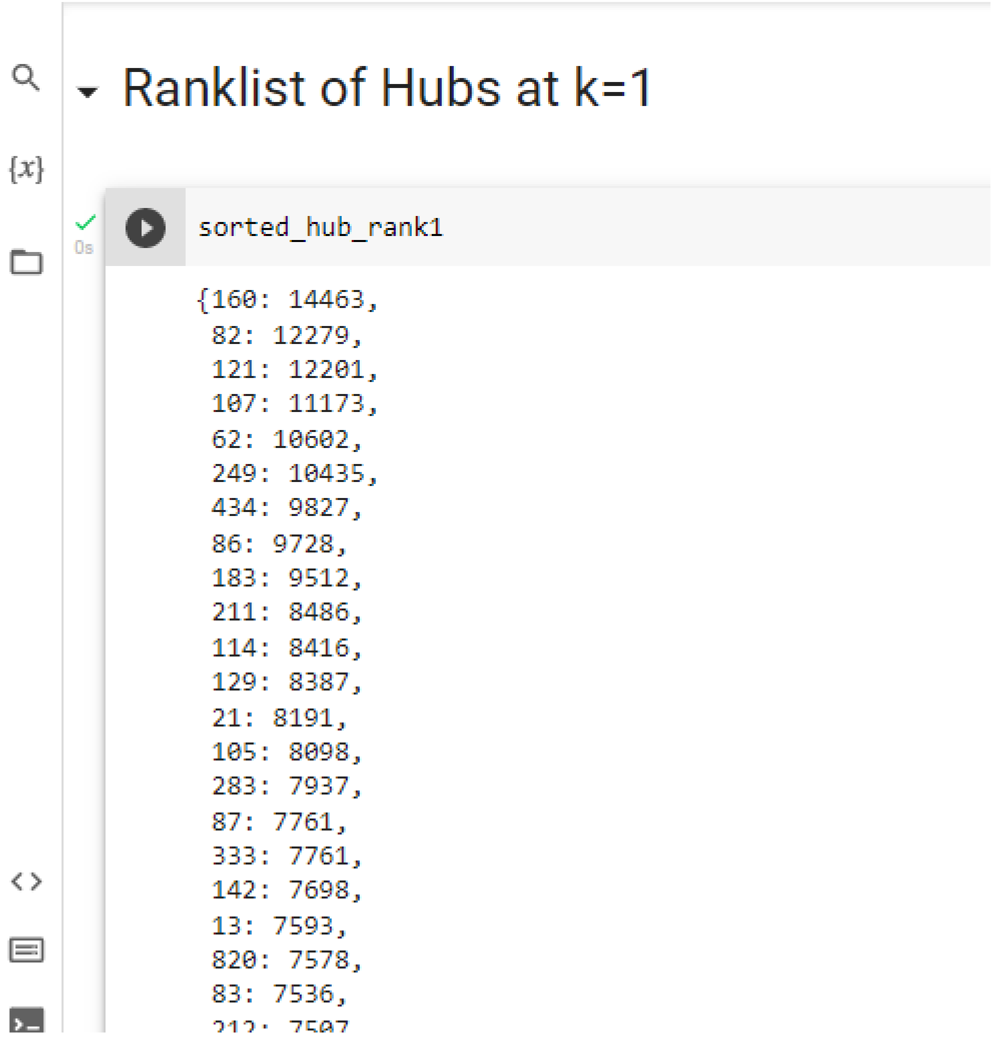
Result



Authority_rank for k=1

```
sorted_auth_rank1
```

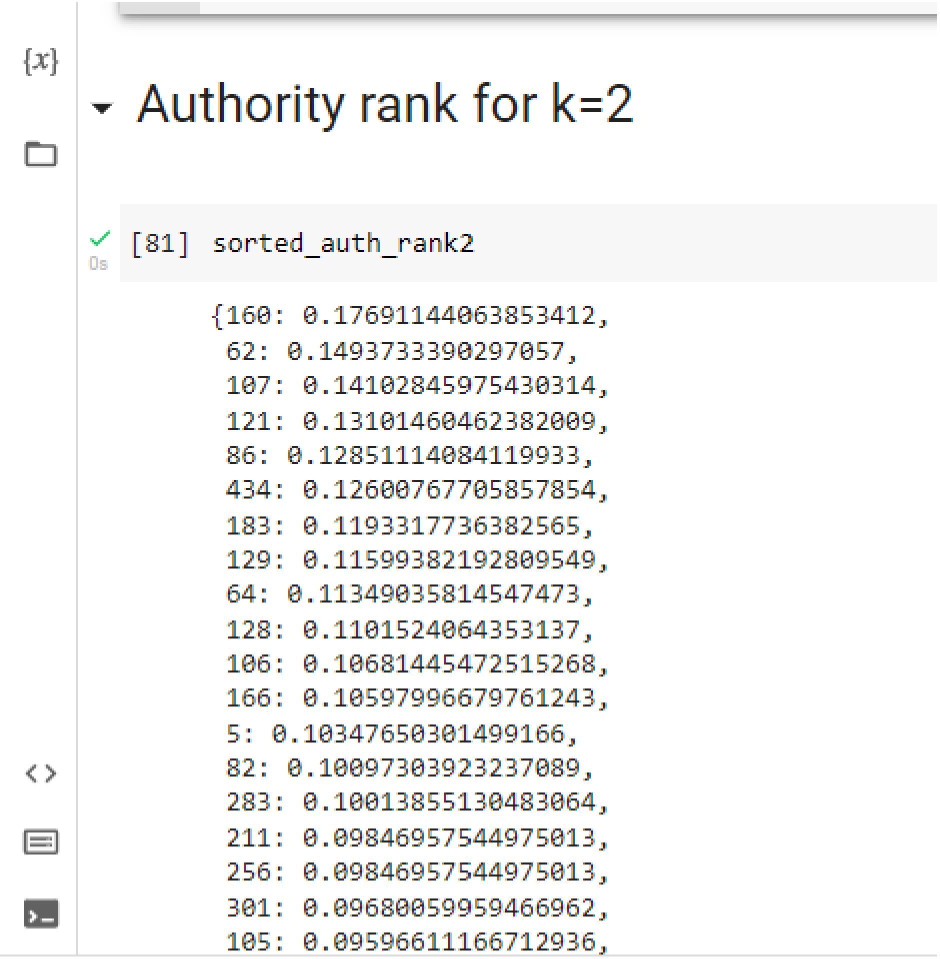
```
{160: 212,
 62: 179,
 107: 169,
 121: 157,
 86: 154,
 434: 151,
 183: 143,
 129: 139,
 64: 136,
 128: 132,
 106: 128,
 166: 127,
 5: 124,
 82: 121,
 283: 120,
 211: 118,
 256: 118,
 301: 116,
 105: 115,
 249: 112,
 21: 111.
```



Ranklist of Hubs at k=1

```
sorted_hub_rank1
```

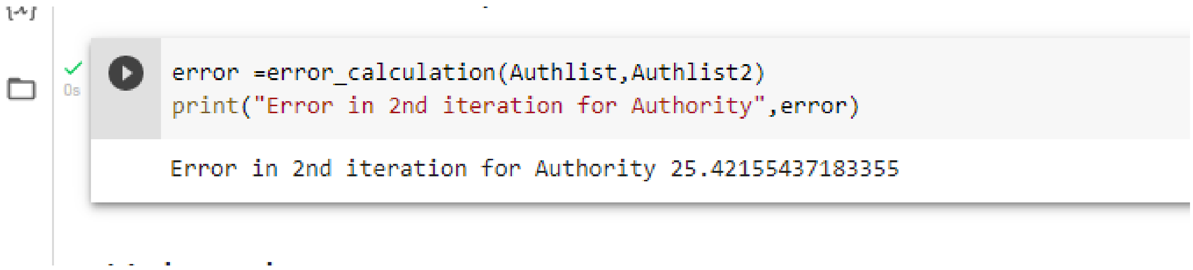
```
{160: 14463,
 82: 12279,
 121: 12201,
 107: 11173,
 62: 10602,
 249: 10435,
 434: 9827,
 86: 9728,
 183: 9512,
 211: 8486,
 114: 8416,
 129: 8387,
 21: 8191,
 105: 8098,
 283: 7937,
 87: 7761,
 333: 7761,
 142: 7698,
 13: 7593,
 820: 7578,
 83: 7536,
 212: 7507
```



Authority rank for k=2

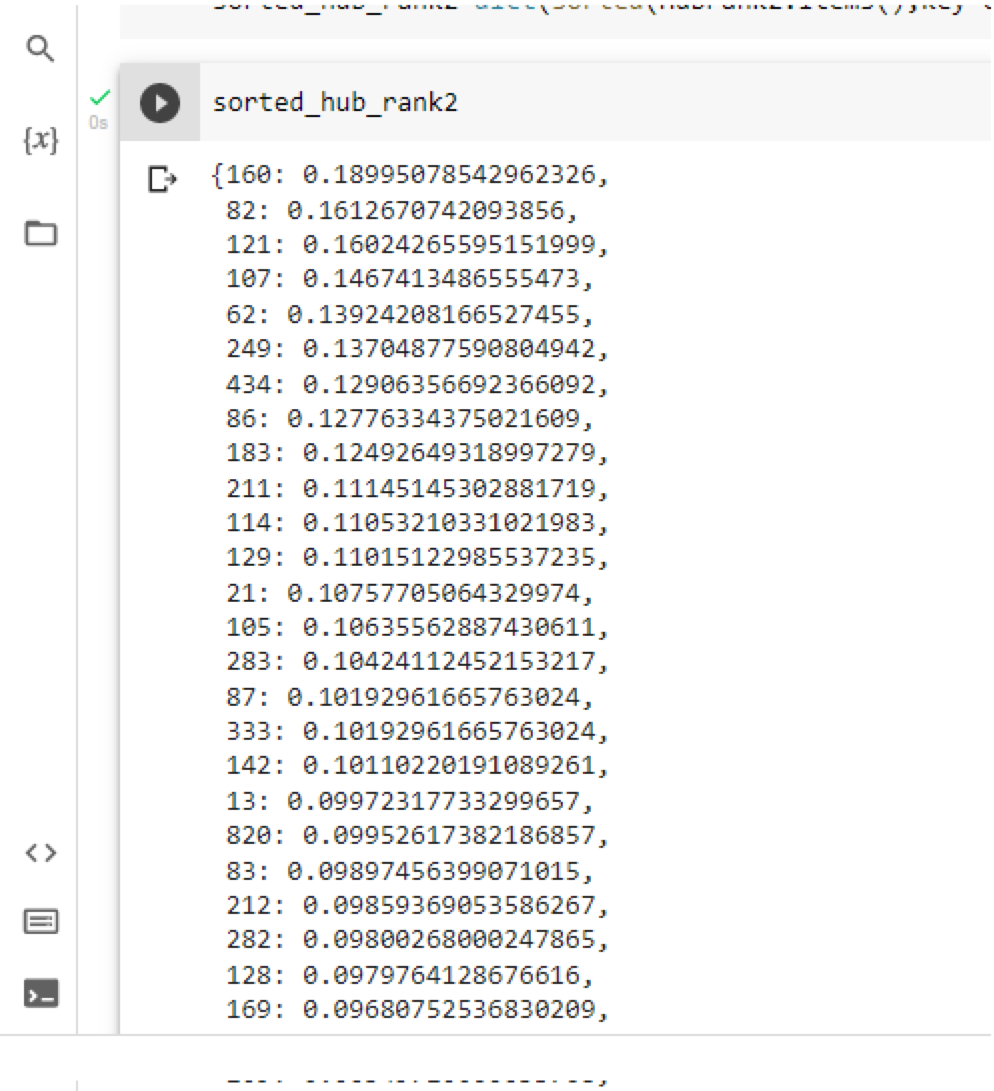
```
[81] sorted_auth_rank2
```

```
{160: 0.17691144063853412,
 62: 0.1493733390297057,
 107: 0.14102845975430314,
 121: 0.13101460462382009,
 86: 0.12851114084119933,
 434: 0.12600767705857854,
 183: 0.1193317736382565,
 129: 0.11599382192809549,
 64: 0.11349035814547473,
 128: 0.1101524064353137,
 106: 0.10681445472515268,
 166: 0.10597996679761243,
 5: 0.10347650301499166,
 82: 0.10097303923237089,
 283: 0.10013855130483064,
 211: 0.09846957544975013,
 256: 0.09846957544975013,
 301: 0.09680059959466962,
 105: 0.09596611166712936,
```



```
error =error_calculation(Authlist,Authlist2)
print("Error in 2nd iteration for Authority",error)
```

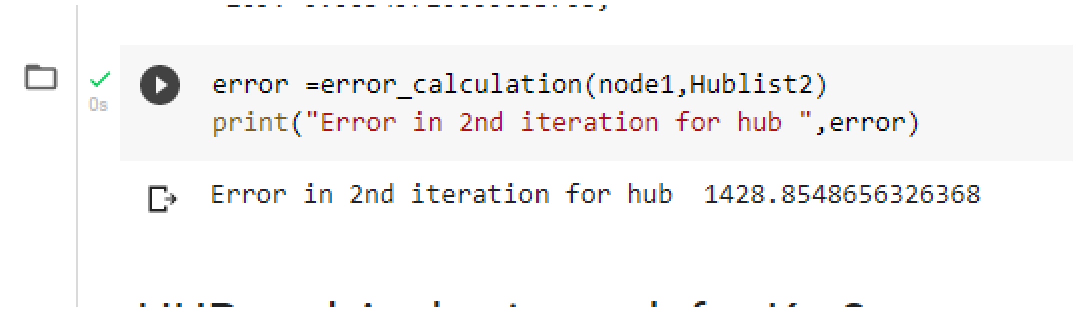
```
Error in 2nd iteration for Authority 25.42155437183355
```



```
sorted_hub_rank2 =dict(sorted(hubrank2.items(),key=
```

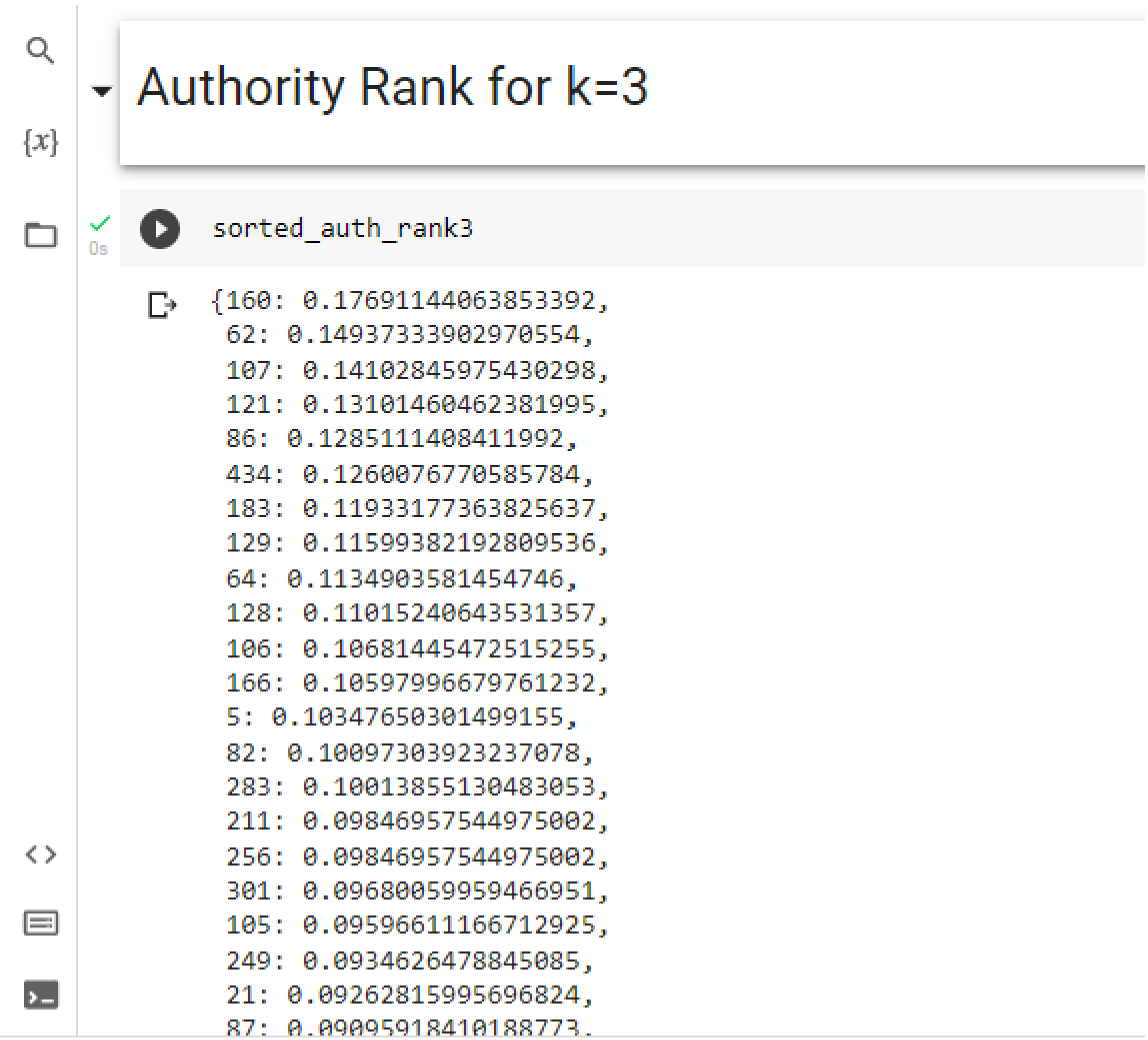
```
sorted_hub_rank2
```

```
{160: 0.18995078542962326,
 82: 0.1612670742093856,
 121: 0.16024265595151999,
 107: 0.1467413486555473,
 62: 0.13924208166527455,
 249: 0.13704877590804942,
 434: 0.12906356692366092,
 86: 0.12776334375021609,
 183: 0.12492649318997279,
 211: 0.11145145302881719,
 114: 0.11053210331021983,
 129: 0.11015122985537235,
 21: 0.10757705064329974,
 105: 0.10635562887430611,
 283: 0.10424112452153217,
 87: 0.10192961665763024,
 333: 0.10192961665763024,
 142: 0.10110220191089261,
 13: 0.09972317733299657,
 820: 0.09952617382186857,
 83: 0.09897456399071015,
 212: 0.09859369053586267,
 282: 0.09800268000247865,
 128: 0.0979764128676616,
 169: 0.09680752536830209,
```



```
error =error_calculation(node1,Hublist2)
print("Error in 2nd iteration for hub ",error)
```

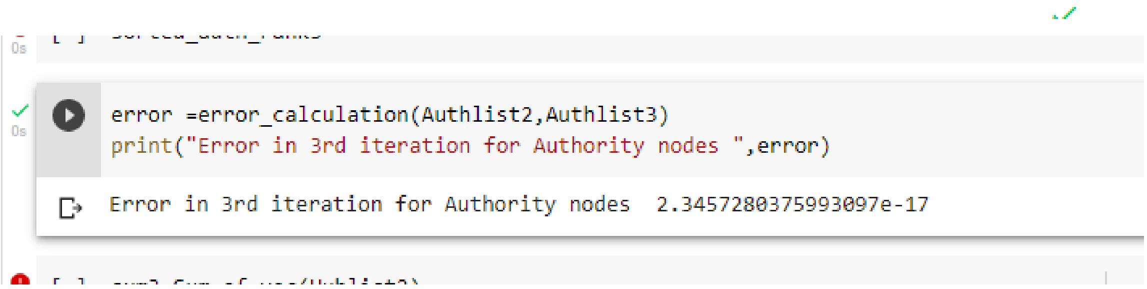
```
Error in 2nd iteration for hub  1428.8548656326368
```



Authority Rank for k=3

```
sorted_auth_rank3
```

```
{160: 0.17691144063853392,
 62: 0.14937333902970554,
 107: 0.14102845975430298,
 121: 0.13101460462381995,
 86: 0.1285111408411992,
 434: 0.1260076770585784,
 183: 0.11933177363825637,
 129: 0.11599382192809536,
 64: 0.1134903581454746,
 128: 0.11015240643531357,
 106: 0.10681445472515255,
 166: 0.10597996679761232,
 5: 0.10347650301499155,
 82: 0.10097303923237078,
 283: 0.10013855130483053,
 211: 0.09846957544975002,
 256: 0.09846957544975002,
 301: 0.09680059959466951,
 105: 0.09596611166712925,
 249: 0.0934626478845085,
 21: 0.09262815995696824,
 87: 0.09095918410188773,
```

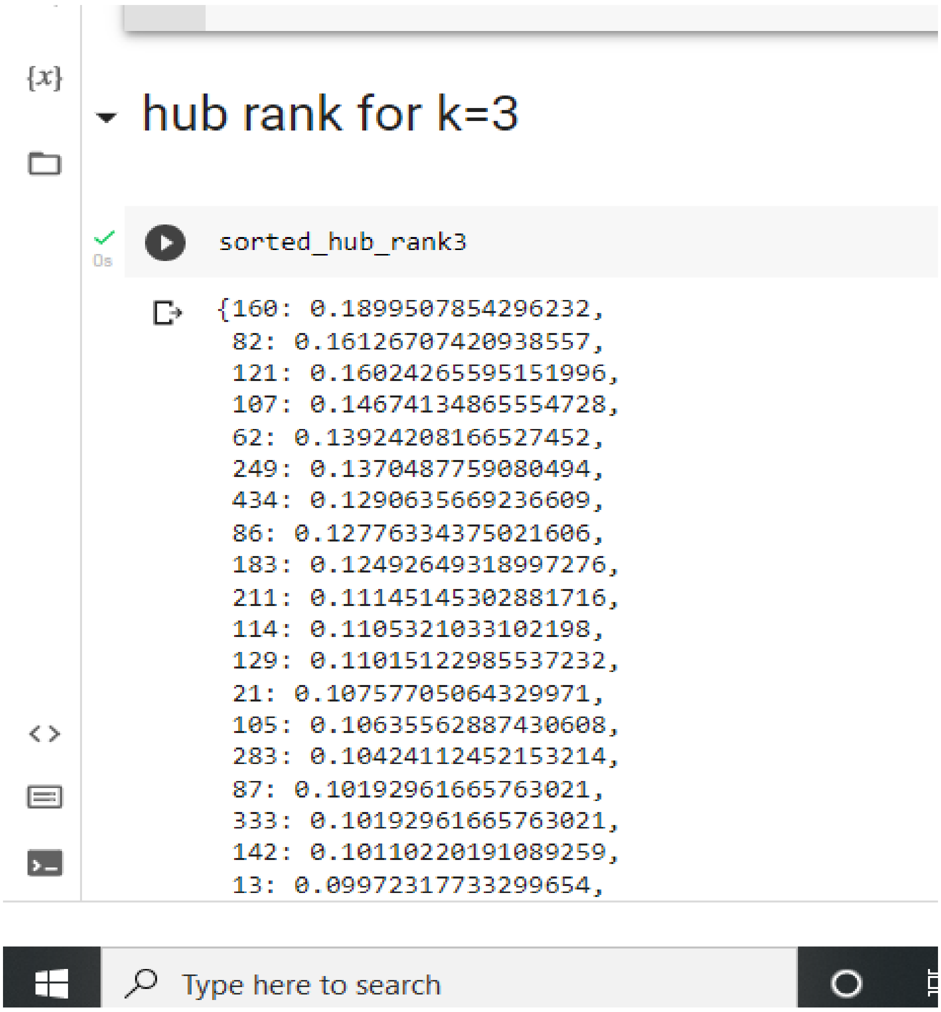


```
[ ] sorted_auth_rank3
```

```
error =error_calculation(Authlist2,Authlist3)
print("Error in 3rd iteration for Authority nodes ",error)
```

```
Error in 3rd iteration for Authority nodes  2.3457280375993097e-17
```

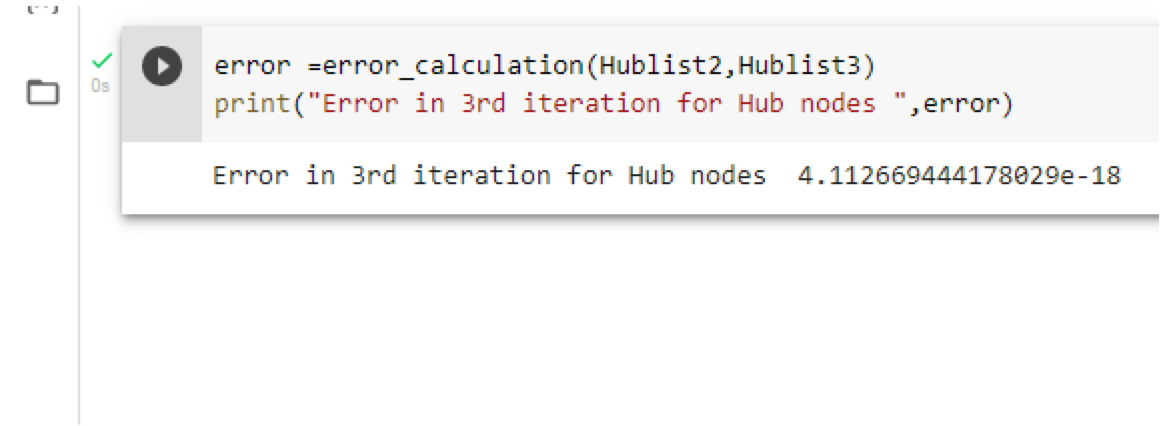
```
[ ] sum3=Sum_of_vec(Hublist3)
```



hub rank for k=3

```
sorted_hub_rank3
```

```
{160: 0.1899507854296232,
 82: 0.16126707420938557,
 121: 0.16024265595151996,
 107: 0.14674134865554728,
 62: 0.13924208166527452,
 249: 0.1370487759080494,
 434: 0.1290635669236609,
 86: 0.12776334375021606,
 183: 0.12492649318997276,
 211: 0.11145145302881716,
 114: 0.1105321033102198,
 129: 0.11015122985537232,
 21: 0.10757705064329971,
 105: 0.10635562887430608,
 283: 0.10424112452153214,
 87: 0.10192961665763021,
 333: 0.10192961665763021,
 142: 0.10110220191089259,
 13: 0.09972317733299654,
```



```
error =error_calculation(Hublist2,Hublist3)
print("Error in 3rd iteration for Hub nodes ",error)
```

```
Error in 3rd iteration for Hub nodes  4.112669444178029e-18
```

Part C Comparision of the results obtained from both the algorithms in parts 1 and 2 based on the node scores.

In part 1 we use the Page rank algorithm to compute node rank so it basically based on the structure of the graph and incoming edges of node(indegree of nodes) if the indegree of a node a more then it has the best score because we sum value of all nodes of incoming degree

In part2 score is calculated based on the Authority weight vector and hub weight vector so 1st node for every iteration is the same but in part1 as well in part2 then other change.