Analysis

Question 1

Data after Preprocessing:

	0	1
0	[xref, cantaloup, srv, cmu, edu, comp, graphic...	comp.graphics
1	[path, cantaloup, srv, cmu, edu, crabappl, srv...	comp.graphics
2	[path, cantaloup, srv, cmu, edu, da, news, har...	comp.graphics
3	[xref, cantaloup, srv, cmu, edu, comp, edu, se...	comp.graphics
4	[newsgroup, comp, graphic, path, cantaloup, sr...	comp.graphics
...	...	...
4995	[xref, cantaloup, srv, cmu, edu, soc, mot, one...	talk.politics.misc
4996	[xref, cantaloup, srv, cmu, edu, talk, polit, ...	talk.politics.misc
4997	[path, cantaloup, srv, cmu, edu, crabappl, srv...	talk.politics.misc
4998	[newsgroup, talk, polit, misc, path, cantaloup...	talk.politics.misc
4999	[xref, cantaloup, srv, cmu, edu, talk, polit, ...	talk.politics.misc

5000 rows × 2 columns

Naive Bayes Results:

```
********** At ratio 50:50 **********
Confusion Matrix is-
 [[484   0   7   7   0]
 [  7 481   5   1   1]
 [  6   0 461   2   0]
 [  9   0  10 479   4]
 [ 10   6  12  12 496]]
Accuracy is 96.04

********** At ratio 70:30 **********
Confusion Matrix is-
 [[309   0   3   2   0]
 [  5 291   1   1   0]
 [  3   0 268   2   0]
 [  4   0  11 286   2]
 [  2   4   9   3 294]]
Accuracy is 96.53

********** At ratio 80:20 **********
Confusion Matrix is-
 [[208   0   3   3   0]
 [  5 189   1   2   0]
 [  2   0 167   1   0]
 [  1   0   8 202   0]
 [  1   3   6   3 195]]
Accuracy is 96.10
```

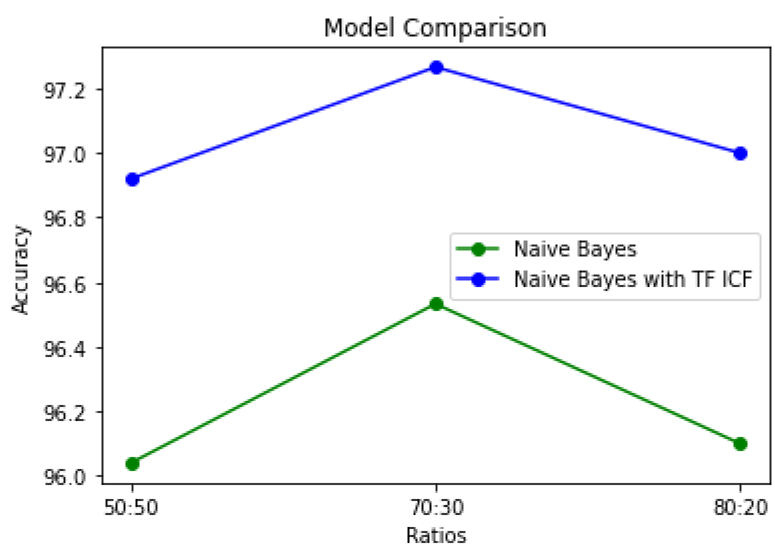
Naive Bayes with TF ICF Results:

```
[13] ********** At ratio 50:50 **********
     Confusion Matrix is-
      [[511   2  21  14   0]
      [  2 482   3   3   2]
      [  0   0 464   3   2]
      [  3   1   6 478   9]
      [  0   2   1   3 488]]
     Accuracy is 96.92

     ********** At ratio 70:30 **********
     Confusion Matrix is-
      [[321   2   8   7   1]
      [  1 291   0   2   0]
      [  0   0 278   5   1]
      [  1   0   4 280   4]
      [  0   2   2   0 290]]
     Accuracy is 97.33

     ********** At ratio 80:20 **********
     Confusion Matrix is-
      [[215   2   5   5   0]
      [  2 188   1   2   0]
      [  0   0 173   1   0]
      [  0   0   4 203   4]
      [  0   2   2   0 191]]
     Accuracy is 97.00
```

Comparison between both the models:



Question 2

Dataset Statistics :

Nodes	7115
Edges	103689
Nodes in largest WCC	7066 (0.993)
Edges in largest WCC	103663 (1.000)
Nodes in largest SCC	1300 (0.183)
Edges in largest SCC	39456 (0.381)
Average clustering coefficient	0.1409
Number of triangles	608389
Fraction of closed triangles	0.04564
Diameter (longest shortest path)	7
90-percentile effective diameter	3.8

Data Analysis and Exploration :

From the adjacency list, we get the following results -

```
Number of nodes :  7115
Number of edges :  103689
```

```
Average In-Degree :  14.573295853829936
Average Out-Degree :  14.573295853829936
```

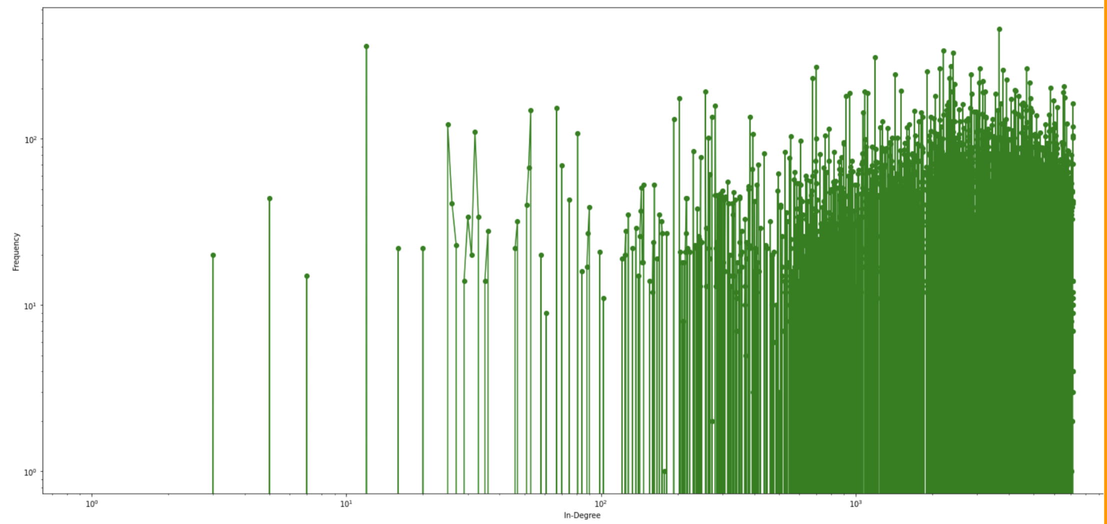
```
Node with maximum  457  In-Degree :  Node  4037
Node with maximum  893  Out-Degree :  Node  2565
```

```
Density of the network is :  0.0020485375110809584
```

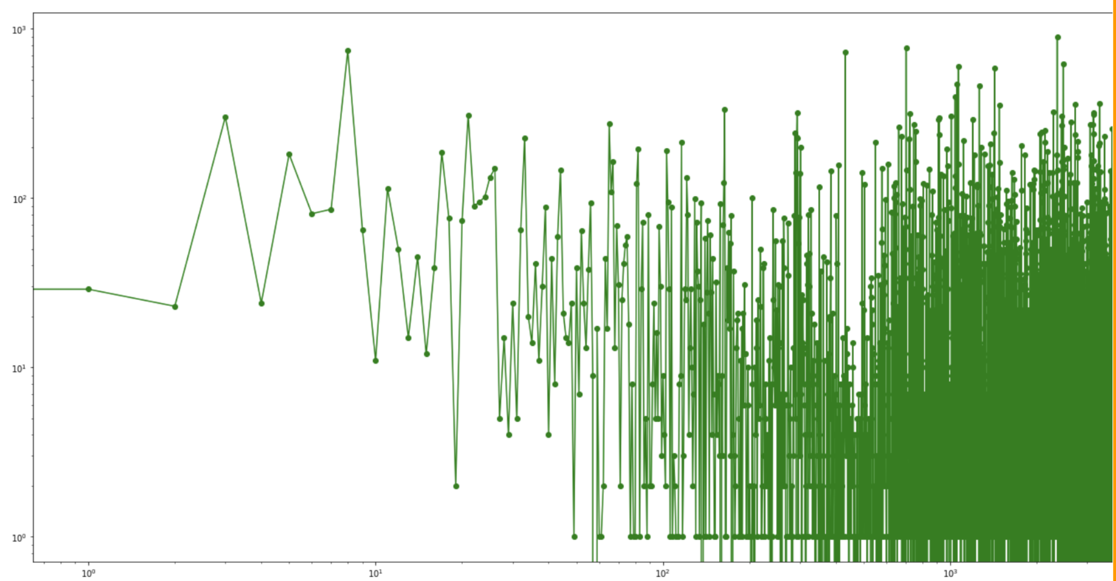
Results :

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

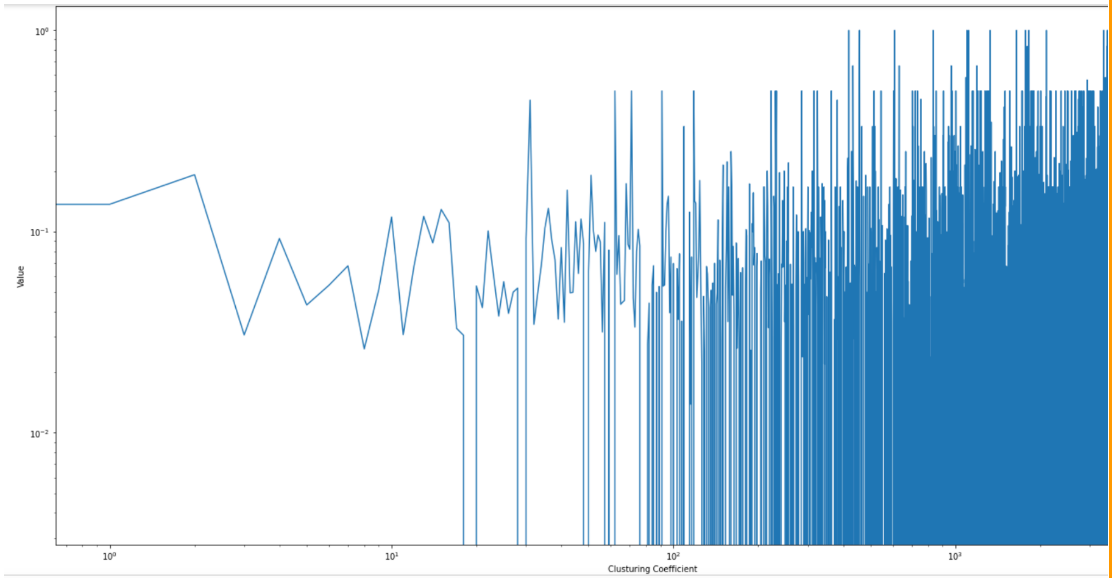
In-Degree of node x1 refers to the number of edges incident to x1.



Out-Degree of node x1 refers to the number of edges that emerged from x1 to other nodes.



2. Calculate the local clustering coefficient of each node and plot the clustering coefficient distribution of the network.



3. Find any 1 centrality measure for each node. Store the values in a separate text file.

dCenterality.txt

```
3 : 0.005342
4 : 0.015603
5 : 0.013635
6 : 0.391763
7 : 0.007169
8 : 0.200028
9 : 0.049339
10 : 0.069441
11 : 2.024318
12 : 0.029941
13 : 0.001827
14 : 0.055665
15 : 0.023053
16 : 0.003514
17 : 0.024459
18 : 0.002390
19 : 0.023053
20 : 0.161653
```

Part B (Dataset: wiki-Vote):

PageRank Scores for each node:

PageRank Scores

```
[6] #default damping factor of 0.85
    prscores = nx.pagerank(G, max_iter = 100)

    #Top to least PageRank Scores for each node
    dict(sorted(prscores.items(), key=lambda item: item[1], reverse = True))
    #prscores
```

```
{'4037': 0.0046127158911675485,
 '15': 0.0036812207295292792,
 '6634': 0.003524813657640259,
 '2625': 0.0032863743692309023,
 '2398': 0.0026053331717250192,
 '2470': 0.0025301053283849546,
 '2237': 0.002504703800483994,
 '4191': 0.0022662633042363454,
 '7553': 0.002170185049195958,
 '5254': 0.0021500675059293235,
 '1186': 0.0020438936876029153,
 '2328': 0.0020416288860889186,
 '1297': 0.0019518608216122285,
 '4335': 0.0019353014475784877,
 '7620': 0.0019301193957548752,
 '5412': 0.00191670807752399,
 '7632': 0.0019037739909136618,
 '4875': 0.0018675748225119092,
 '3352': 0.0017851250122027215,
 '2654': 0.0017693207143482425,
 '6832': 0.0017646895191923734,
 '762': 0.0017478626294191988,
```

Authority Scores for each node:

Authority scores for each node:

```
#Top to least Authorities score for each node
dict(sorted(authorities.items(), key=lambda item: item[1], reverse = True))
```

```
{'2398': 0.002580147178008918,
 '4037': 0.002573241124142803,
 '3352': 0.002328415091537902,
 '1549': 0.0023037314804751075,
 '762': 0.00225587485637424,
 '3089': 0.0022534066884266454,
 '1297': 0.00225014463679536,
 '2565': 0.002223564103945871,
 '15': 0.002201543492543811,
 '2625': 0.0021978968035237852,
 '2328': 0.0021723715454129585,
 '2066': 0.0021070409397065445,
 '4191': 0.0020811941305208825,...
```

Hub Scores for each node:

```
[ ] #Top to least Hub Score for each node
    dict(sorted(hubs.items(), key=lambda item: item[1], reverse = True))
```

```
{'2565': 0.00794049270807403,
 '766': 0.007574335297444512,
 '2688': 0.006440248991012525,
 '457': 0.00641687049019565,
 '1166': 0.006010567902433343,
 '1549': 0.0057207540583986485,
 '11': 0.004921182064008282,
 '1151': 0.004572040701802756,
 '1374': 0.004467888792672376,
 '1133': 0.003918881732047633,...
```

Checking if scores taken for all the nodes:

```
#checking if scores are taken for all the nodes (each node)
print(len(prscores))
print(len(hubs))
print(len(authorities))
```

```
7115
7115
7115
```