

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN



BÁO CÁO ĐỒ ÁN
MẠNG XÃ HỘI

**ĐỀ TÀI: PHÂN TÍCH DỮ LIỆU NHỮNG NGÀNH NGHỀ
IT ĐƯỢC ĐĂNG TUYỂN TRÊN LINKEDIN**

Lớp: IS353.O11.HTCL

GVHD: Nguyễn Thị Kim Phụng

Sinh viên thực hiện:

Nguyễn Thị Cẩm Vân 20522145

Lưu Thảo Linh 20521532

Tháng 12/2023, TPHCM

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN



BÁO CÁO ĐỒ ÁN
MẠNG XÃ HỘI

**ĐỀ TÀI: PHÂN TÍCH DỮ LIỆU NHỮNG NGÀNH NGHỀ
IT ĐƯỢC ĐĂNG TUYỂN TRÊN LINKEDIN**

Lớp: IS353.O11.HTCL

GVHD: Nguyễn Thị Kim Phụng

Sinh viên thực hiện:

Nguyễn Thị Cẩm Vân 20522145

Lưu Thảo Linh 20521532

Tháng 12/2023, TPHCM

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the entire width of the page. There are no margins, text, or other markings present.

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Với lòng biết ơn sâu sắc nhất, đầu tiên em xin gửi lời cảm ơn chân thành đến tập thể **quý Thầy Cô Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM** và **quý Thầy Cô khoa Hệ thống thông tin** đã giúp cho nhóm có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt nhóm chúng em xin gửi lời cảm ơn chân thành tới cô **Nguyễn Thị Kim Phụng – giảng viên môn Mạng Xã Hội** đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn em trong suốt quá trình làm đề tài. Nhờ đó, em đã tiếp thu được nhiều kiến thức bổ ích trong việc vận dụng cũng như kỹ năng làm đề tài. Nếu không có những lời hướng dẫn, dạy bảo của cô thì em nghĩ đề tài này của em rất khó có thể hoàn thiện được. Một lần nữa, em xin chân thành cảm ơn cô.

Cuối cùng, bản thân em đã làm việc hết công suất để hoàn thành tốt đề tài của mình. Xin chân thành cảm ơn!

MỤC LỤC

LỜI NHẬN XÉT CỦA GIẢNG VIÊN.....	3
LỜI CẢM ƠN.....	4
MỤC LỤC	5
CHƯƠNG 1: TỔNG QUAN	7
1.1 Giới thiệu Dataset.....	7
1.2 Xác định bài toán.....	7
1.3 Mô tả dữ liệu	7
1.3.1 Nguồn dữ liệu	7
1.3.2 Mô tả dữ liệu.....	7
CHƯƠNG II: XỬ LÝ VÀ PHÂN TÍCH DỮ LIỆU.....	9
2.1 Các thư viện được sử dụng	9
2.2 Làm sạch dữ liệu.....	9
2.3 Chuyển đổi dataframe thành đồ thị.....	11
2.3.1 Đồ thị 2 phía	11
2.3.2 Đồ thị 1 phía	13
CHƯƠNG III. CÁC ĐỘ ĐO VÀ KẾT QUẢ.....	17
3.1 Degree Centrality	17
3.1.1 Python	18
3.1.2 Gephi.....	22
3.2 Closeness Centrality	23
3.2.1 Python	23
3.2.2 Gephi.....	28
3.3 Betweenness Centrality	29
3.3.1 Python	29
3.3.2 Gephi.....	33
3.4 Eigen Vector	34
3.4.1 Python	34
3.4.2 Gephi.....	38
3.5 Pagerank.....	39
3.5.1 Python	39
3.5.2 Gephi.....	43

CHƯƠNG IV: THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG	44
4.1 Thuật toán Girvan Newman.....	44
4.2 Thuật toán Louvain	48
4.3 Thuật toán K-Means	54
CHƯƠNG V: TRÍCH XUẤT 10 NODE VÀ THỰC HIỆN TÍNH TAY	63
5.1 Mô hình Gephi	63
5.2 Các độ đo	64
5.2.1 Closeness Centrality	64
5.2.2 Betweenness Centrality	65
5.2.3 Eigen Vector	65
5.2.4 PageRank	66
5.2.5 Clustering Coefficient.....	66
5.2.6 Harmonic	67
TÀI LIỆU THAM KHẢO.....	68

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu Dataset

Bộ dữ liệu LinkedIn cung cấp một cái nhìn tổng quan toàn diện về các thông tin tuyển dụng được đăng trên nền tảng, tập trung vào các vị trí công việc như Nhà phân tích dữ liệu, Kỹ sư máy học, Dịch vụ CNTT và Tư vấn CNTT tại nhiều địa điểm trên thế giới. Dữ liệu này rất hữu ích để thực hiện các phân tích, trực quan hóa, và nghiên cứu về xu hướng việc làm trong các lĩnh vực này.

Bộ dữ liệu bao gồm thông tin chi tiết về các tin tuyển dụng, bao gồm thông tin về công ty, mô tả trách nhiệm của vai trò, cũng như mức lương và giờ làm việc tương ứng. Nó cung cấp cái nhìn sâu sắc về cơ hội việc làm ở các địa điểm đa dạng, giúp người nghiên cứu hiểu rõ về đặc điểm và yêu cầu của các vị trí công việc trong lĩnh vực phân tích dữ liệu, máy học, và CNTT.

Với thông tin về công ty, trách nhiệm công việc, và kỹ năng yêu cầu, bộ dữ liệu này là nguồn tài nguyên quý giá để phân tích sâu sắc về thị trường lao động, giúp cá nhân và doanh nghiệp có cái nhìn rõ ràng về cơ hội và xu hướng việc làm trong các ngành công nghiệp quan trọng này.

1.2 Xác định bài toán

- Input: Tập dữ liệu ban đầu trên nguồn dữ liệu Kaggle đã qua tiền xử lý dữ liệu
- Output: Đưa ra độ đo, đưa ra cộng đồng phục vụ cho việc phân tích mạng xã hội “JobLinkined”

1.3 Mô tả dữ liệu

1.3.1 Nguồn dữ liệu

- Link dữ liệu: [LinkedIn_job_cleandata | Kaggle](#)

1.3.2 Mô tả dữ liệu

- Mỗi dòng trong tập dữ liệu là một ngành nghề được đăng tải trên LinkedIn
- Dữ liệu gồm 5587 dòng và 15 thuộc tính
- Sau khi xử lý khử trùng còn 403 dòng

Tên cột	Kiểu dữ liệu	Ý nghĩa
Job_ID	Int	Mã định danh duy nhất cho mỗi danh sách công việc.
Designation	String	Chức danh hoặc vị trí của công việc.
Company_id	Int	Mã định danh duy nhất cho mỗi công ty.
name	String	Tên của công ty cung cấp công việc.
work_type	String	Cho biết công việc là từ xa hay tại chỗ.
involvement	String	Mức độ tham gia cần thiết cho công việc (ví dụ: Toàn thời gian, Bán thời gian).
employees_count	Int	Số lượng nhân viên trong công ty.
total_applicants	Int	Tổng số người nộp đơn xin việc.
linkedin_followers	Int	Số lượng người theo dõi trên trang LinkedIn của công ty.
job_details	String	Mô tả công việc và trách nhiệm của nó.
details_id	Int	Mã định danh duy nhất cho chi tiết công việc
industry	String	Ngành hoặc lĩnh vực mà công ty thuộc về.
level	String	Mức độ kinh nghiệm hoặc thâm niên liên quan đến công việc.
City	String	Thành phố nơi làm việc.
State	String	Tiểu bang nơi đặt công việc.

CHƯƠNG II: XỬ LÝ VÀ PHÂN TÍCH DỮ LIỆU

2.1 Các thư viện được sử dụng

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import ndlib.models.ModelConfig as mc
import ndlib.models.epidemics as ep

from networkx.algorithms import bipartite
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
import matplotlib
import seaborn as sns
from termcolor import colored
from networkx.algorithms.community import label_propagation_communities
from sklearn import preprocessing
from sklearn.cluster import KMeans
from pandas.core.frame import DataFrame
```

2.2 Làm sạch dữ liệu

- Đọc dữ liệu từ file CSV và đưa vào dataframe

```
# df = pd.read_csv("job_cleanData.csv", usecols = ['City','industry'])
# df
from google.colab import drive

df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/job_cleanData.csv', usecols = ['City','industry'])
df
```

	industry	City
0	IT Services and IT Consulting	Delhi
1	IT Services and IT Consulting	New Delhi
2	IT Services and IT Consulting	Greater Bengaluru Area
3	Not Avilable	Gurugram
4	Not Avilable	Mohali district
...
5582	IT Services and IT Consulting	Kochi
5583	IT Services and IT Consulting	Gurugram
5584	IT Services and IT Consulting	Hyderabad
5585	IT Services and IT Consulting	Bengaluru
5586	IT Services and IT Consulting	Bengaluru

5587 rows x 2 columns

- Làm sạch dữ liệu, xóa dữ liệu trùng lặp

```

# #Tim missing values
# features_na = [features for features in df.columns if df[features].isnull().sum() >0]
# for feature in features_na:
#     print(feature, np.round(df[feature].isnull().mean(),4), '% missing values')
# else:
#     print("No missing values found")

df = df.replace("Not Available", np.nan)
df.dropna(inplace = True)
# df.drop_duplicates()
# df.reset_index(drop=True,inplace=True)
# df
# df = df.dropna()

# Loại bỏ các hàng trùng lặp
df = df.drop_duplicates()

df

```

	industry	City
0	IT Services and IT Consulting	Delhi
1	IT Services and IT Consulting	New Delhi
2	IT Services and IT Consulting	Greater Bengaluru Area
5	Telecommunications	Gurugram
6	IT Services and IT Consulting	Bengaluru
...
5532	IT Services and IT Consulting	Sriperumbudur
5537	Software Development	Kochi
5549	Business Consulting and Services	Greater Kolkata Area
5559	Technology Information and Internet	Navi Mumbai
5565	Hospitals and Health Care	Bangalore Urban

403 rows x 2 columns

- Sau khi xóa các dữ liệu trống và dữ liệu trùng lặp, Kết quả ta nhận được bộ dữ liệu 403 dòng và 2 cột

```

df.info()
df.describe()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 403 entries, 0 to 5565
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  -
0   industry    403 non-null    object
1   City        403 non-null    object
dtypes: object(2)
memory usage: 9.4+ KB

```

	industry	City
count	403	403
unique	102	76
top	IT Services and IT Consulting	Bengaluru
freq	59	56

- Kiểm tra số lượng dữ liệu, xuất ra số thành phố, số ngành công nghiệp, số cạnh

Xuất ra số thành phố và số ngành IT, số cạnh

```
[ ] import networkx as nx
    B = nx.Graph()
    City = df['City']
    Industry = df['industry']

    print('Số thành phố: ', City.nunique())
    print('Số ngành nghề: ', Industry.nunique())
    print('Số cạnh: ', len(df))

Số thành phố: 76
Số ngành nghề: 102
Số cạnh: 403
```

2.3 Chuyển đổi dataframe thành đồ thị

2.3.1 Đồ thị 2 phía

- Tạo node

Tạo Node

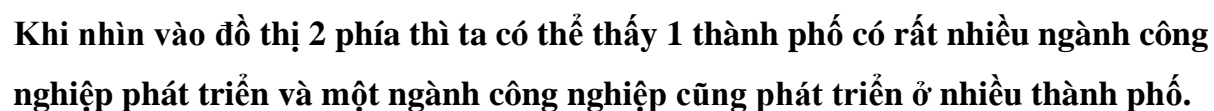
```
for index, row in df.iterrows():
    B.add_edge(row['City'], row['industry'], weight = 1)
B.add_nodes_from(City, bipartite = 0)
B.add_nodes_from(Industry, bipartite = 1)
```

- + **Node:** là tên của ngành nghề và tên các thành phố
- + **Edge:** cho ta biết ngành IT được đăng tuyển ở thành phố nào
- Code hiển thị đồ thị 2 phía

```
[9] for index, row in df.iterrows():
    B.add_edge(row['City'], row['industry'], weight = 1)
    B.add_nodes_from(City, bipartite = 0)
    B.add_nodes_from(Industry, bipartite = 1)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
pos = nx.spring_layout(B)
fig, ax = plt.subplots(1,1,figsize=(15,30), dpi = 150)
nx.draw_networkx(B, pos = nx.drawing.layout.bipartite_layout(B, Industry), font_size = 8, width = 0.4)
```

- Đồ thị 2 phía



2.3.2 Đồ thị 1 phía

🚦 Node: Là các thành phố

🚦 Edge: Hai thành phố có cùng một ngành công nghiệp sẽ hợp thành 1 cạnh

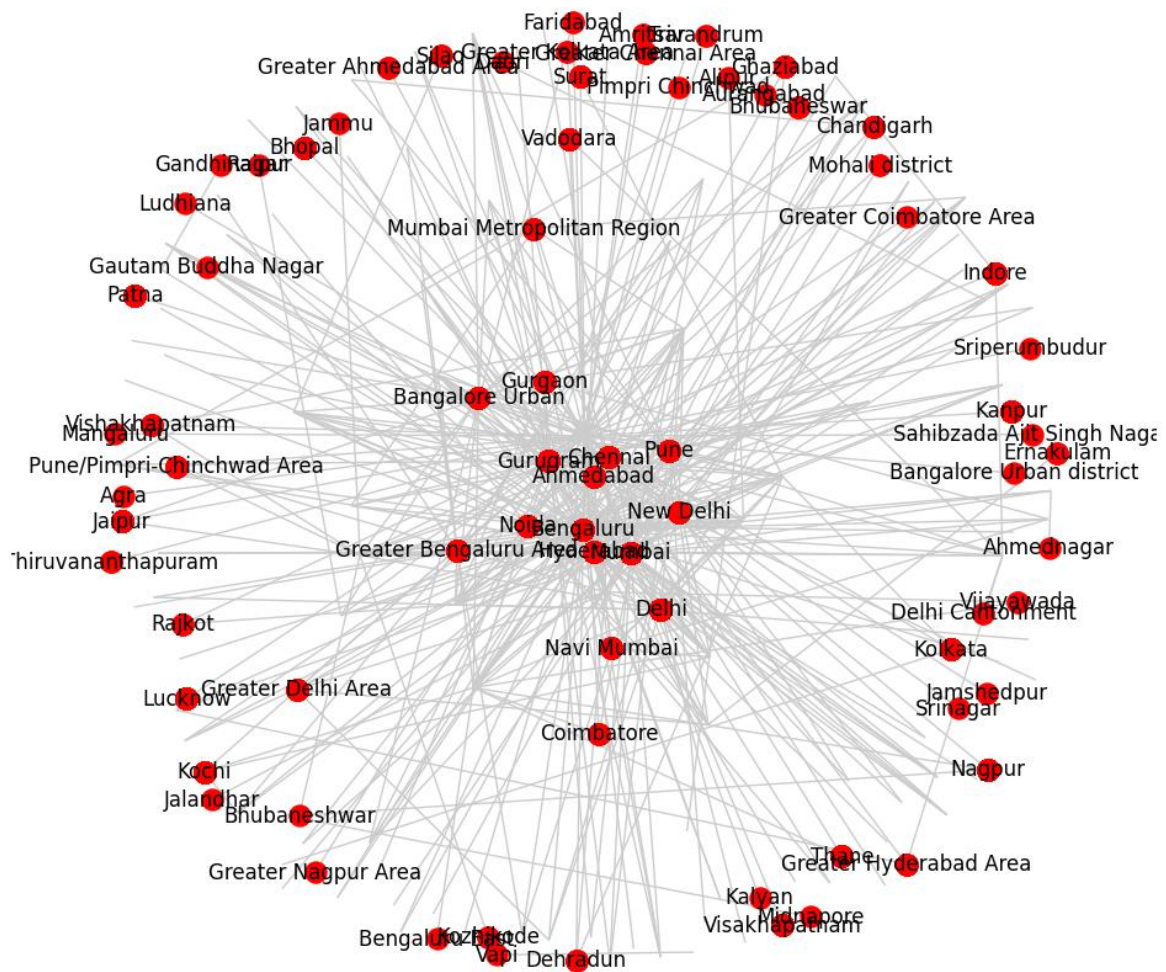
- Code hiển thị đồ thị 1 phía:

ĐỒ THỊ 1 PHÍA

- node: Thành phố
- edge: hai thành phố có cùng ngành công nghiệp sẽ hợp thành 1 cạnh

```
▶ G = bipartite.weighted_projected_graph(B, City.unique())
plt.figure(figsize=(12,12))
layout = nx.spring_layout(B, k = 1)
nx.draw_networkx_nodes(B, layout, nodelist=City, node_size=150, node_color='red')
nx.draw_networkx_edges(B, layout, edge_color='#cccccc')
node_labels = dict(zip(City, City))
nx.draw_networkx_labels(B, layout, labels = node_labels)
plt.axis('off')
plt.title("Graph City")
plt.show()
```

- Đồ thị 1 phía:



Xuất dữ liệu đồ thị một phía ra file csv để thực hiện trên Gephi

Lưu các cạnh của đồ thị vào edges

Tạo dataframe là các cạnh u, v, weight tương đương với source, target, weight

Xuất ra file Gephi để đưa vào phần mềm Gephi

```
[13] edges = G.edges.data()
```

```
data = pd.DataFrame(edges, columns = ['u', 'v', 'weight'])  
data.to_csv('Gelpi.csv')  
data
```



	u	v	weight
0	Delhi	Mumbai	{'weight': 9}
1	Delhi	Pimpri Chinchwad	{'weight': 1}
2	Delhi	Dehradun	{'weight': 3}
3	Delhi	Jammu	{'weight': 1}
4	Delhi	Sahibzada Ajit Singh Nagar	{'weight': 1}
...
1849	Mangaluru	Sriperumbudur	{'weight': 1}
1850	Mangaluru	Gautam Buddha Nagar	{'weight': 1}
1851	Greater Kolkata Area	Sriperumbudur	{'weight': 1}
1852	Greater Kolkata Area	Gautam Buddha Nagar	{'weight': 1}
1853	Gautam Buddha Nagar	Sriperumbudur	{'weight': 1}

1854 rows × 3 columns

Source	Target	weight
Delhi	Surat	{'weight': 1}
Delhi	Mangaluru	{'weight': 1}
Delhi	Kozhikode	{'weight': 1}
Delhi	Thiruvananthapuram	{'weight': 1}
Delhi	Bengaluru	{'weight': 11}
Delhi	Kochi	{'weight': 3}
Delhi	Kanpur	{'weight': 1}
Delhi	Chandigarh	{'weight': 2}
Delhi	Pimpri Chinchwad	{'weight': 1}
Delhi	Agra	{'weight': 1}
Delhi	Navi Mumbai	{'weight': 5}
Delhi	Srinagar	{'weight': 1}
Delhi	Mumbai	{'weight': 9}
Delhi	Ahmedabad	{'weight': 3}
Delhi	Jalandhar	{'weight': 1}
Delhi	Pune/Pimpri-Chinchwad Area	{'weight': 3}
Delhi	Greater Chennai Area	{'weight': 1}
Delhi	Greater Delhi Area	{'weight': 4}
Delhi	Noida	{'weight': 10}

>
Gelphi
+

- Xuất ra số đỉnh và số cạnh của đồ thị và kiểm tra đồ thị liên thông

Xuất ra số đỉnh và số cạnh của đồ thị và kiểm tra đồ thị liên thông

```
G = nx.Graph()

edges = data[['u','v']]
G = nx.from_pandas_edgelist(edges, 'u', 'v')

print('số đỉnh của đồ thị là ',len(G.nodes()))
print('số cạnh của đồ thị là ',len(G.edges()))
```

```
➞ số đỉnh của đồ thị là 75
số cạnh của đồ thị là 1854
```

```
[16] #kiểm tra đồ thị liên thông
print('Kiểm tra đồ thị liên thông: ')
nx.is_connected(G)
```

```
Kiểm tra đồ thị liên thông:
True
```

CHƯƠNG III. CÁC ĐỘ ĐO VÀ KẾT QUẢ

Sau khi import file csv dữ liệu ở trên vào Data Laboratory trên Gephi ta thu được các độ đo cần thiết để so sánh kết quả giữa Gephi và Python

3.1 Degree Centrality

Hệ số này sẽ giúp chúng ta đo lường được số lượng của các mối quan hệ trực tiếp của một actor nào đó với các thành viên khác trong mạng lưới. Giá trị của hệ số này chạy từ 0.00 đến 1.00 và khi giá trị càng gần tới 1.00 thì tính trung tâm trực tiếp của actor càng lớn, tức là càng nằm ở vị trí trung tâm của mạng lưới. Công thức tính như sau:

$$C_d = \frac{k}{n-1}$$

Trong đó,

k = Tổng số các mối quan hệ trực tiếp của actor

n = Tổng số actor trong mạng lưới


3.1.1 Python

- **Code Degree Centrality**

✓ Degree Centrality

```
def print_table(data, columns):  
    df = pd.DataFrame(data, columns = columns)  
    return df  
degree_dict = {node: 0 for node in G.nodes()}  
  
for node in G.nodes():  
    degree_dict[node] = len(list(G.neighbors(node)))  
  
degree_df = print_table(degree_dict.items(), ['Node', 'Degree'])  
degree_df
```

- **Kết quả**

	Node	Degree	
0	Delhi	66	
1	Mumbai	67	
2	Pimpri Chinchwad	58	
3	Dehradun	60	
4	Jammu	16	
...	
70	Gandhinagar	1	
71	Delhi Cantonment	8	
72	Aurangabad	1	
73	Dadri	1	
74	Midnapore	1	

75 rows × 2 columns

- **Top 10 Node có Degree Centrality cao nhất**

```
#Top 10 node có degree cao nhất

sorted_degree_dict = sorted(degree_dict.items(), key=lambda x: x[1], reverse= True)
top_10_max_degree = list(sorted_degree_dict)[:10]

print('Top 10 node có bậc cao nhất : ')
print_table(top_10_max_degree, ['Node', 'Degree'])
```

Top 10 node có bậc cao nhất :

	Node	Degree
0	Bengaluru	70
1	Pune	69
2	Hyderabad	68
3	Mumbai	67
4	Chennai	67
5	Gurugram	67
6	Noida	67
7	Delhi	66
8	New Delhi	66
9	Bangalore Urban	66

- **Top 10 Node có Degree Centrality thấp nhất**

```
#Top 10 có degree thấp nhất

sorted_degree_dict = sorted(degree_dict.items(), key=lambda x: x[1], reverse= False)

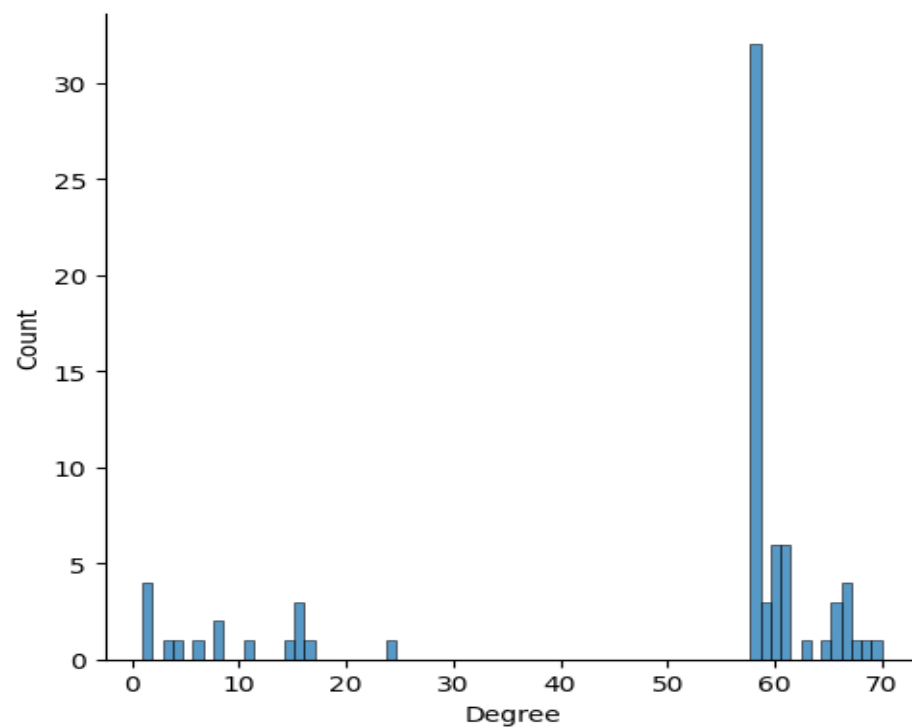
top_10_min_degree = list(sorted_degree_dict)[:10]

print('Top 10 node có bậc thấp nhất : ')
print_table(top_10_min_degree, ['Node', 'Degree'])
```

Top 10 node có bậc thấp nhất :

	Node	Degree
0	Gandhinagar	1
1	Aurangabad	1
2	Dadri	1
3	Midnapore	1
4	Ahmednagar	3
5	Raipur	4
6	Ernakulam	6
7	Silao	8
8	Delhi Cantonment	8
9	Alipur	11

- Biểu đồ phân bố Degree Centrality

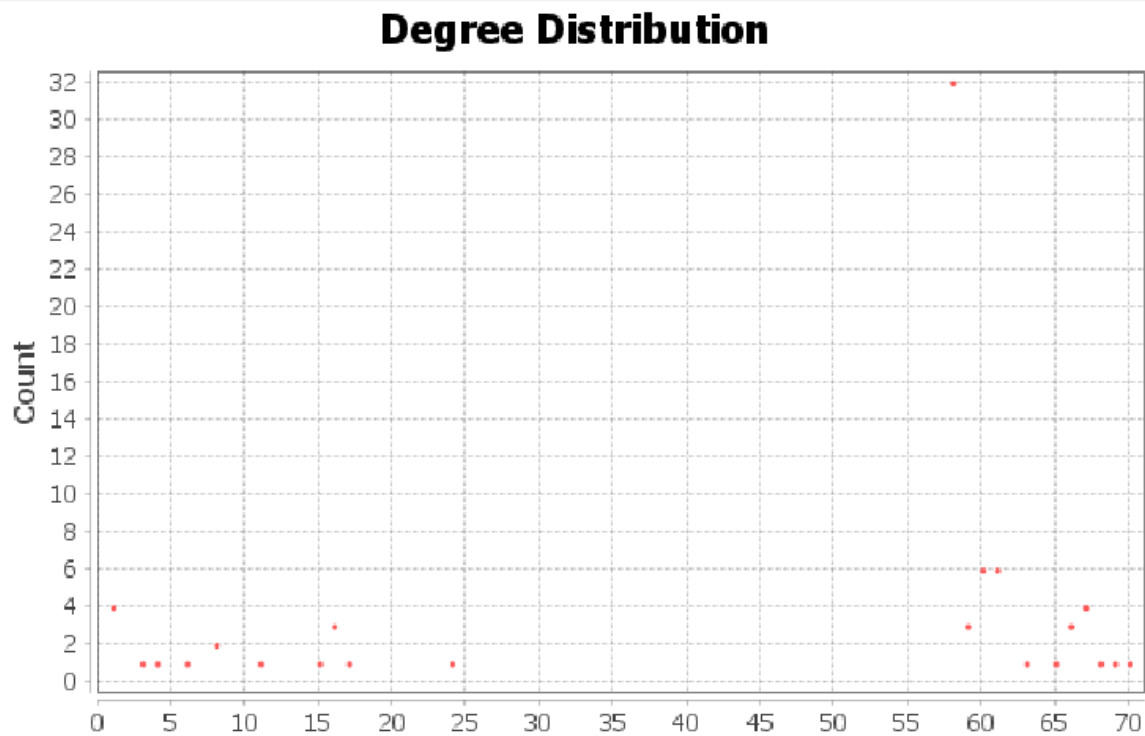


3.1.2 Gephi

Degree Report

Results:

Average Degree: 49.440



Mật độ phân bố Degree Centrality

- Top 10 Node có Degree Centrality cao nhất

Id	Label	Interval	Degree ▾
Bengaluru			70
Pune			69
Hyderabad			68
Mumbai			67
Noida			67
Chennai			67
Gurugram			67
Bangalore Ur...			66
New Delhi			66
Delhi			66

- **Top 10 Node có Degree Centrality thấp nhất**

Id	Label	Interval	Degree ^
Midnapore			1
Gandhinagar			1
Aurangabad			1
Dadri			1
Ahmednagar			3
Raipur			4
Ernakulam			6
Delhi Canton...			8
Silao			8
Alipur			11

3.2 Closeness Centrality

Điểm yếu của hệ số trung tâm trực tiếp là nó chỉ tính các mối quan hệ trực tiếp của actor mà thôi nên chưa chắc actor có hệ số trung tâm trực tiếp cao là người "gần gũi" với mọi thành viên khác trong mạng. Tính gần gũi hay lân cận cũng là một trong những tiêu chí quan trọng thể hiện vị thế của actor trong mạng, bởi một actor càng gần gũi với các thành viên trong mạng lưới bao nhiêu thì actor đó càng dễ có nhiều thông tin, càng có nhiều uy thế và do đó càng dễ gây ảnh hưởng lên toàn bộ mạng lưới. Để đo lường hệ số này, chúng ta sẽ tính tổng số "bước" (step) của "đoạn đường" ngắn nhất (geodesic path) mà actor phải "đi" để đến được với tất cả các thành viên khác trong mạng lưới.

Hệ số này cũng có giá trị đi từ 0.00 đến 1.00, càng gần đến 1.00 thì actor càng gần với mọi thành viên khác trong mạng lưới, tức đoạn đường phải đi để đến với mọi actor khác càng ngắn và ngược lại. Công thức tính hệ số này như sau:

$$C_c = \frac{n-1}{\sum d(x,y)}$$

Trong đó:

n = Tổng số actor trong mạng lưới

$\sum d(x,y)$ = Tổng số "bước" (step) của đoạn đường ngắn nhất mà actor phải đi để đến với mọi actor trong mạng

3.2.1 Python

- **Code Closeness Centrality**

```

closeness centrality = nx.closeness centrality(G)

sorted_closeness centrality = sorted(closeness centrality.items(), key = lambda x:x[1], reverse = True)
closeness_df = print_table(closeness centrality.items(), ['Node', 'Closeness'])

closeness_df

```

- Kết quả

	Node	Closeness
0	Delhi	0.902439
1	Mumbai	0.913580
2	Pimpri Chinchwad	0.822222
3	Dehradun	0.840909
4	Jammu	0.556391
...
70	Gandhinagar	0.490066
71	Delhi Cantonment	0.524823
72	Aurangabad	0.486842
73	Dadri	0.486842
74	Midnapore	0.462500

75 rows × 2 columns

- Top 10 Node có Closeness Centrality cao nhất



#top 10 node có closeness centrality cao nhất

```
top_10_max_closeness = sorted_closeness_centrality[:10]
```

```
print('Top 10 node có closeness centrality cao nhất: ')
```

```
print_table(top_10_max_closeness, ['Node', 'Closeness'])
```



Top 10 node có closeness centrality cao nhất:

	Node	Closeness
0	Bengaluru	0.948718
1	Pune	0.936709
2	Hyderabad	0.925000
3	Mumbai	0.913580
4	Chennai	0.913580
5	Gurugram	0.913580
6	Noida	0.913580
7	Delhi	0.902439
8	New Delhi	0.902439
9	Bangalore Urban	0.902439



- Top 10 Node có Closeness Centrality thấp nhất



```
#top 10 node có closeness centrality thấp nhất
```

```
top_10_min_closeness = sorted_closeness_centrality[-10:]
```

```
print('Top 10 node có closeness centrality thấp nhất: ')
```

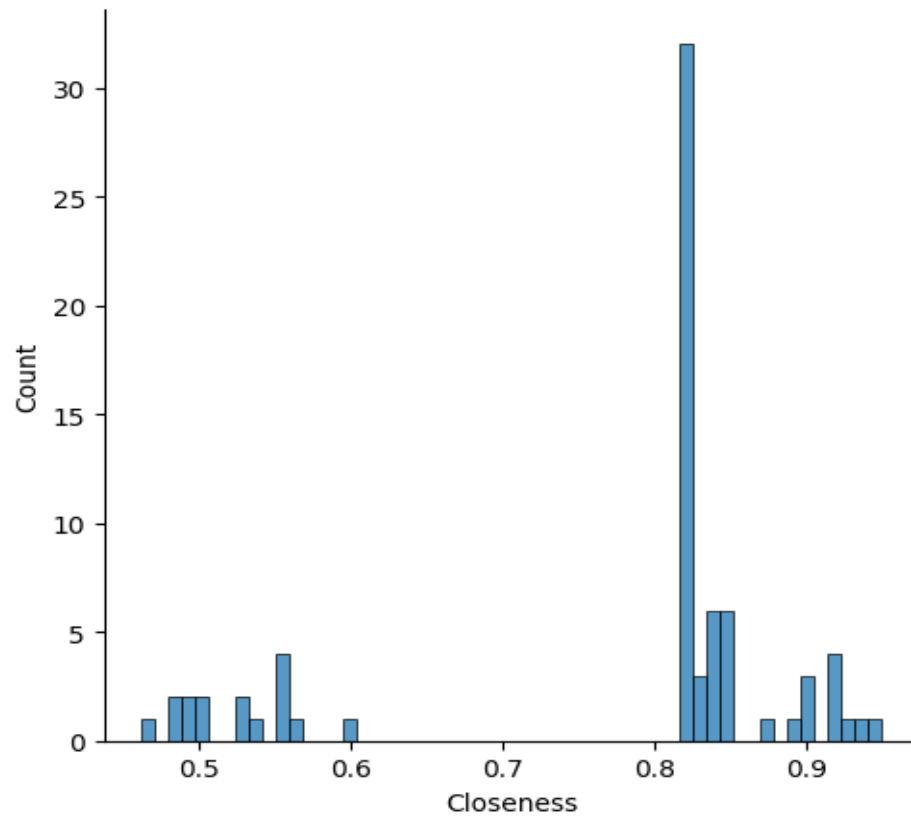
```
print_table(top_10_min_closeness, ['Node', 'Closeness'])
```



Top 10 node có closeness centrality thấp nhất:

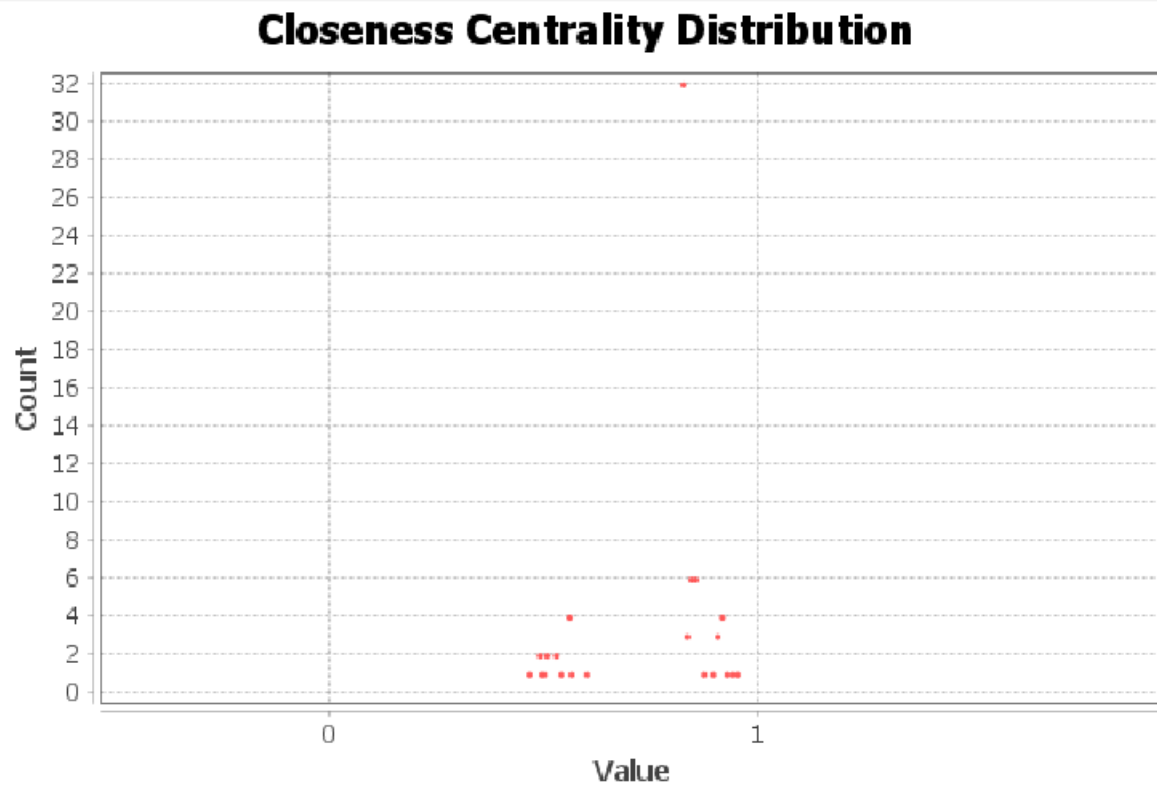
	Node	Closeness
0	Alipur	0.536232
1	Silao	0.524823
2	Delhi Cantonment	0.524823
3	Ernakulam	0.503401
4	Raipur	0.503401
5	Ahmednagar	0.496644
6	Gandhinagar	0.490066
7	Aurangabad	0.486842
8	Dadri	0.486842
9	Midnapore	0.462500

- **Biểu đồ phân bố Closeness Centrality**



→ Độ đo closeness là độ gần khoảng cách của các node với các node khác trong mạng, độ đo càng cao thì thành phố đó xuất hiện ngành công nghiệp có chung cạnh nhiều nhất với các thành phố khác.

3.2.2 Gephi



- Top 10 Node có Closeness Centrality cao nhất

Id	Label	Interval	Closeness Centrali... ▾
Bengaluru			0.948718
Pune			0.936709
Hyderabad			0.925
Mumbai			0.91358
Noida			0.91358
Chennai			0.91358
Gurugram			0.91358
Bangalore Ur...			0.902439
New Delhi			0.902439
Delhi			0.902439

- Top 10 Node có Closeness Centrality thấp nhất

Id	Label	Interval	Closeness Centrali... ^
Midnapore			0.4625
Aurangabad			0.486842
Dadri			0.486842
Gandhinagar			0.490066
Ahmednagar			0.496644
Raipur			0.503401
Ernakulam			0.503401
Delhi Canton...			0.524823
Silao			0.524823
Alipur			0.536232

3.3 Betweenness Centrality

Theo quan điểm của Freeman, một actor nào đó trong mạng lưới có thể ít gắn kết với các thành viên khác trong mạng lưới (tức hệ số trung tâm trực tiếp thấp), cũng không "gần gũi" lắm với mọi thành viên trong mạng lưới (tức hệ số trung tâm lân cận thấp), nhưng lại là "cầu nối" (bridge), là "nhà trung gian" cần thiết trong mọi cuộc trao đổi trong mạng lưới.

Hệ số này cũng đi từ 0.00 đến 1.00. Khi một actor nào đó có hệ số trung tâm trung gian càng gần đến 1.00 thì số lượng quan hệ giữa các actor khác phải "thông qua" actor này càng nhiều và do đó ảnh hưởng của actor cũng càng lớn.

Cách tính hệ số trung tâm trung gian như sau:

$$C_B = \frac{n(j,z;x)}{(n-1)(n-2)/2}$$

Trong đó

$n(j,z;x)$ = Tổng số lần làm "trung gian" của actor i

n = Tổng số actor trong mạng

3.3.1 Python

- Code Betweenness Centrality

```

Betweenness Centrality



betweenness centrality = nx.betweenness centrality(G,normalized=False)

sorted_betweenness = sorted(betweenness centrality.items(), key=lambda x:x[1], reverse=True)

betweenness_df = print_table(betweenness centrality.items(), ['Node', 'Betweenness'])
betweenness_df

```

- **Kết quả**

	Node	Betweenness	
0	Delhi	34.557093	
1	Mumbai	54.357093	
2	Pimpri Chinchwad	0.000000	
3	Dehradun	4.406083	
4	Jammu	0.000000	
...	
70	Gandhinagar	0.000000	
71	Delhi Cantonment	0.000000	
72	Aurangabad	0.000000	
73	Dadri	0.000000	
74	Midnapore	0.000000	

75 rows × 2 columns

- **Top 10 Node có Betweenness Centrality cao nhất**

```

▶ #Top 10 Betweenness Centrality cao nhất
top_10_max_betw = sorted_betweenness[:10]

print('Top 10 node co Betweenness Centrality lon nhat: ')
print_table(top_10_max_betw, ['Node', 'Betweenness'])

```

➞ Top 10 node co Betweenness Centrality lon nhat:

	Node	Betweenness
0	Pune	184.136855
1	Bengaluru	154.970188
2	Nagpur	77.858009
3	Mumbai	54.357093
4	Hyderabad	51.570188
5	Gurugram	51.557093
6	New Delhi	49.609654
7	Bangalore Urban	44.097511
8	Noida	42.420188
9	Greater Bengaluru Area	42.210606

- **Top 10 Node có Betweenness Centrality thấp nhất**

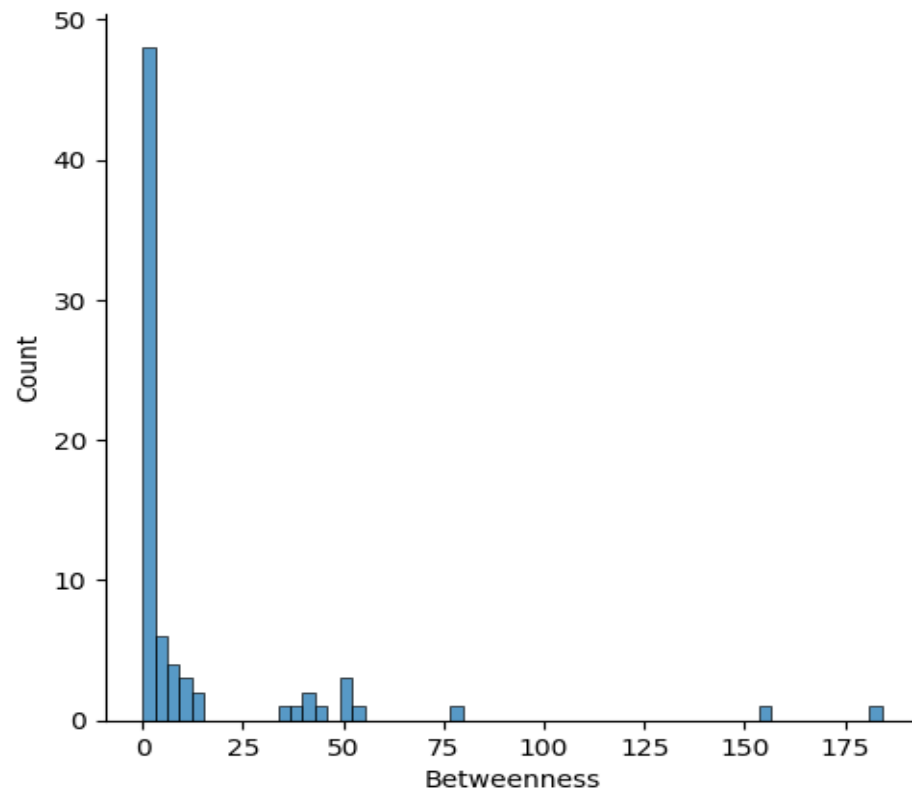
```
#Top 10 Betweenness Centrality thap nhat
top_10_min_betw = sorted_betweenness[-10:]

print('Top 10 node co Betweenness Centrality thap nhat: ')
print_table(top_10_min_betw, ['Node', 'Betweenness'])
```

Top 10 node co Betweenness Centrality thap nhat:

	Node	Betweenness
0	Sriperumbudur	0.0
1	Bangalore Urban district	0.0
2	Raipur	0.0
3	Silao	0.0
4	Ahmednagar	0.0
5	Gandhinagar	0.0
6	Delhi Cantonment	0.0
7	Aurangabad	0.0
8	Dadri	0.0
9	Midnapore	0.0

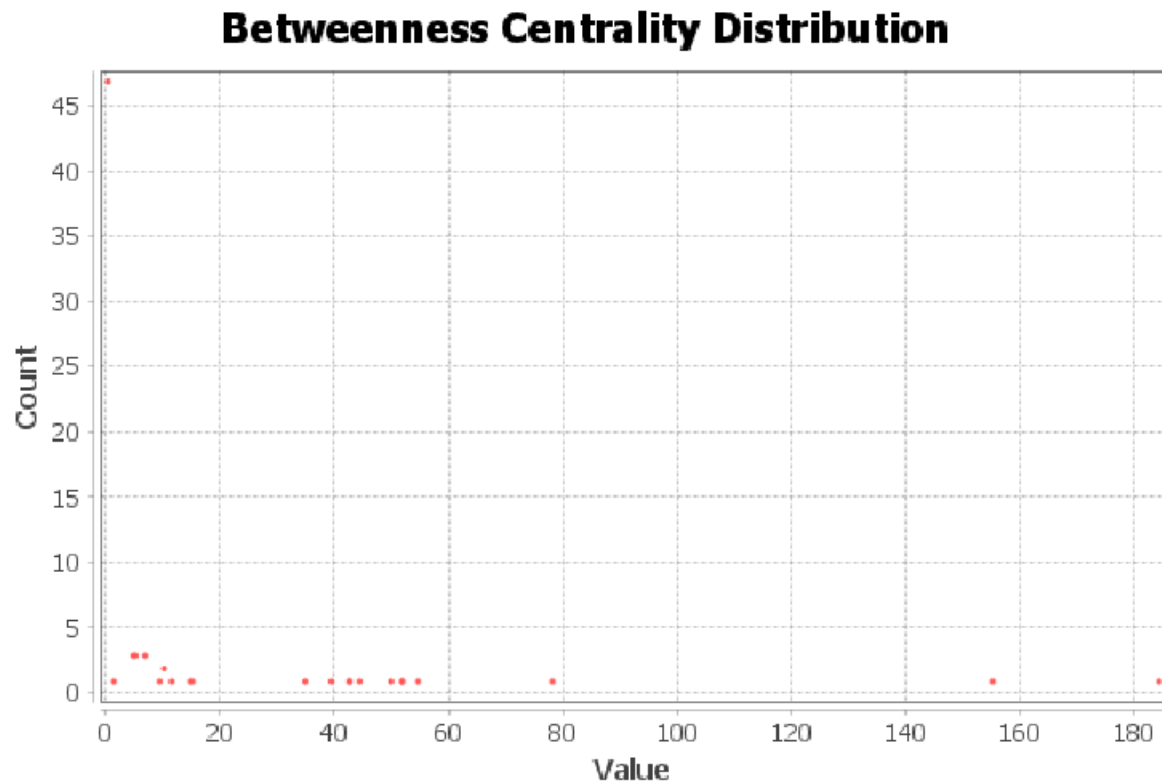
- Biểu đồ phân bố Betweenness Centrality



→ Độ đo betweenness là vai trò trung gian của các mối quan hệ giữa các thành phố trong

mạng, độ đo càng cao thì sự giao thoa của các thành phố càng lớn

3.3.2 Gephi



- Top 10 Node có Betweenness Centrality cao nhất

Id	Label	Interval	Betweenness Centrality
Pune			184.136855
Bengaluru			154.970188
Nagpur			77.858009
Mumbai			54.357093
Hyderabad			51.570188
Gurugram			51.557093
New Delhi			49.609654
Bangalore Ur...			44.097511
Noida			42.420188
Greater Beng...			42.210606

- Top 10 Node có Betweenness Centrality thấp nhất

Id	Label	Interval	Betweenness Centrali... ^
Midnapore			0.0
Aurangabad			0.0
Dadri			0.0
Gandhinagar			0.0
Ahmednagar			0.0
Raipur			0.0
Ernakulam			0.0
Delhi Canton...			0.0
Silao			0.0
Alipur			0.0

3.4 Eigen Vector

Eigenvector là các vector (khác 0) không thay đổi hướng khi bất kỳ phép biến đổi tuyến tính nào được áp dụng. Nó chỉ thay đổi bởi một hệ số vô hướng. Tóm lại, chúng ta có thể nói, nếu A là một phép biến đổi tuyến tính từ một không gian vector V và \mathbf{x} là một vector trong V , không phải là một vector 0, thì λ là một ký hiệu riêng của A nếu $A(\mathbf{x})$ là một bội số vô hướng của \mathbf{x} .

Một Eigenspace của vector \mathbf{x} bao gồm một tập hợp tất cả các eigenvector có giá trị riêng tương đương gọi chung với vector zero. Mặc dù vậy, vector 0 không phải là một ký hiệu riêng.

Giả sử A là ma trận " $n \times n$ " và λ là giá trị riêng của ma trận A , khi đó \mathbf{x} , một vector khác 0, được gọi là eigenvector nếu nó thỏa mãn biểu thức dưới đây;

$$A \mathbf{x} = \lambda \mathbf{x}$$

\mathbf{x} là một ký tự riêng của A tương ứng với giá trị riêng, λ .

3.4.1 Python

- Code Eigen Vector

✓ Eigen Vector

```

eig_cen = nx.eigenvector_centrality(G, max_iter=100)
sorted_eig_cen = sorted(eig_cen.items(), key=lambda x:x[1], reverse=True)

eigen_df = print_table(eig_cen.items(), ['Node', 'EigenVector'])
eigen_df

```

- Kết quả



	Node	EigenVector
0	Delhi	0.132640
1	Bengaluru	0.133322
2	Thiruvananthapuram	0.128333
3	Kalyan	0.128333
4	Jamshedpur	0.128333
...
70	Gandhinagar	0.002277
71	Delhi Cantonment	0.017956
72	Dadri	0.002273
73	Aurangabad	0.002273
74	Midnapore	0.002216

75 rows × 2 columns

- **Top 10 Node có Eigen Vector cao nhất**

```
#Top 10 eigenvector Centrality cao nhat
top_10_max_eig = sorted_eig_cen[:10]

print('Top 10 node co EigenVector Centrality lon nhat: ')
print_table(top_10_max_eig, ['Node', 'EigenVector'])
```

Top 10 node co EigenVector Centrality lon nhat:

	Node	EigenVector
0	Bengaluru	0.133322
1	Hyderabad	0.133245
2	Pune	0.133094
3	Chennai	0.133017
4	Noida	0.132944
5	Mumbai	0.132828
6	Gurugram	0.132792
7	Delhi	0.132640
8	Bangalore Urban	0.132249
9	New Delhi	0.132133

- Top 10 Node có Eigen Vector thấp nhất

```
#Top 10 eigenvector Centrality thap nhat
top_10_min_eig = sorted_eig_cen[-10:]

print('Top 10 node co EigenVector Centrality thap nhat: ')
print_table(top_10_min_eig, ['Node', 'EigenVector'])
```

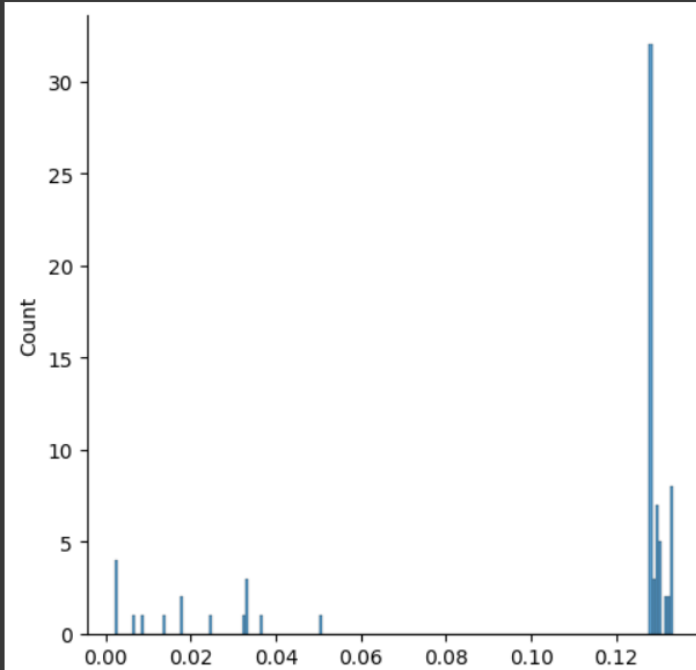
Top 10 node co EigenVector Centrality thap nhat:

	Node	EigenVector
0	Alipur	0.024904
1	Silao	0.018105
2	Delhi Cantonment	0.017956
3	Ernakulam	0.013556
4	Raipur	0.009060
5	Ahmednagar	0.006798
6	Gandhinagar	0.002277
7	Dadri	0.002273
8	Aurangabad	0.002273
9	Midnapore	0.002216

- Biểu đồ phân bố Eigen Vector

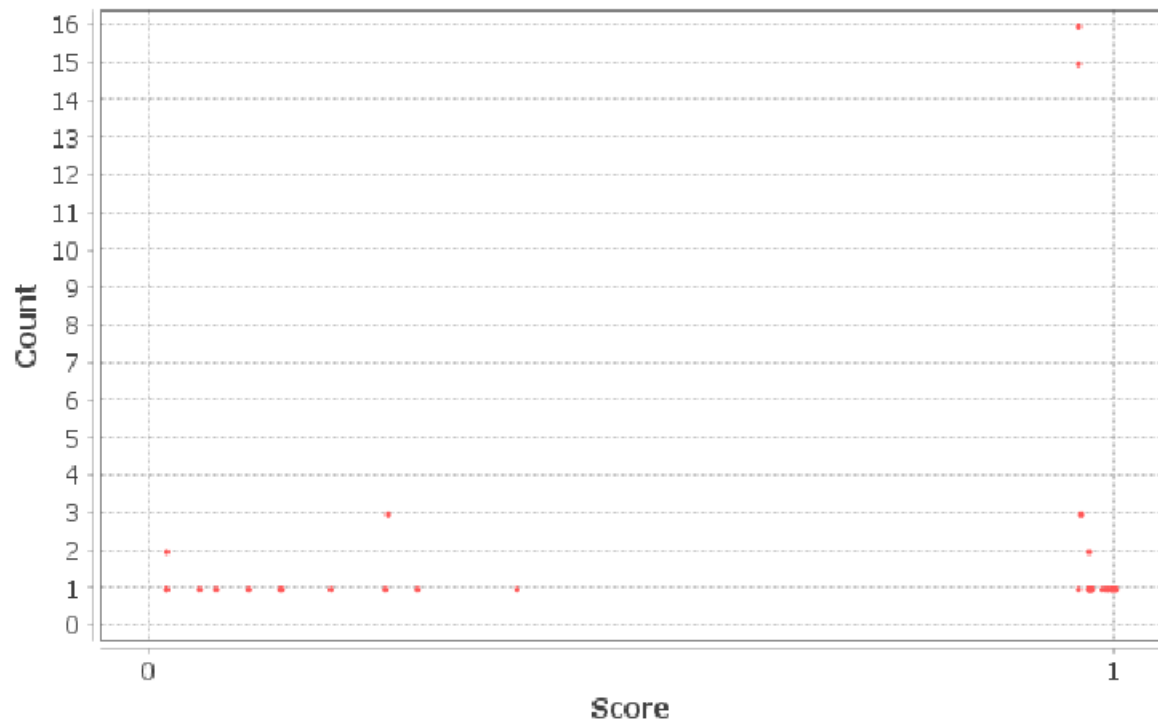
```
#biểu đồ phân bố eigenvector  
print(colored("Biểu đồ thể hiện phân bố eigen vector của các node trong graph", 'green', attrs=['bold']))  
sns.displot(x='EigenVector', data=eigen_df)
```

➡ Biểu đồ thể hiện phân bố eigen vector của các node trong graph
<seaborn.axisgrid.FacetGrid at 0x7cd9a279e950>



3.4.2 Gephi

Eigenvector Centrality Distribution



- Top 10 Node có Eigen Vector cao nhất

Id	Label	Interval	Eigenvector Centrali... ▼
Bengaluru			1.0
Hyderabad			0.999105
Pune			0.998136
Chennai			0.997249
Noida			0.9967
Mumbai			0.995831
Gurugram			0.995566
Delhi			0.994271
Bangalore Ur...			0.991365
New Delhi			0.990491

- Top 10 Node có Eigen Vector thấp nhất

Id	Label	Interval	Eigenvector Centrali... ^
Midnapore			0.016598
Aurangabad			0.017117
Dadri			0.017117
Gandhinagar			0.017159
Ahmednagar			0.051133
Raipur			0.068142
Ernakulam			0.101853
Delhi Canton...			0.134833
Silao			0.136118
Alipur			0.187219

3.5 Pagerank

PageRank là thứ hạng trang. PageRank được phát triển tại đại học Stanford bởi Lary Page (cũng bởi vậy mà có tên PageRank) và sau đó bởi Sergey Brin như một phần dự án công cụ tìm kiếm mới. Theo Google một cách tóm lược thì PageRank chỉ được đánh giá từ hệ thống liên kết đường dẫn. Trang của bạn càng nhận nhiều liên kết (phải là dofollow) trỏ đến thì mức độ quan trọng trang của bạn càng tăng.

Công thức PageRank có dạng như sau:

$$PR(p_i) = \frac{1-d}{n} + d \sum_{p_j \in M(i)} \frac{PR(p_j)}{L(j)}$$

Trong đó:

d: hằng số Google quy định. Thông thường, d = 0.85.

PR(pj): PageRank của các đỉnh đi vào đỉnh i.

L(j): số link out của các đỉnh đi vào đỉnh i.

3.5.1 Python

- Code PageRank

▼ Pagerank

```

▶ pagerank = nx.pagerank(G, tol=1e-6, alpha=0.85)
sorted_pagerank = sorted(pagerank.items(), key = lambda x:x[1], reverse=True)
pagerank_df = print_table(pagerank.items(),['Node', 'PageRank'])
pagerank_df

```

- **Kết quả**

	Node	PageRank
0	Delhi	0.017540
1	Bengaluru	0.021173
2	Thiruvananthapuram	0.014899
3	Kalyan	0.014899
4	Jamshedpur	0.014899
...
70	Gandhinagar	0.002257
71	Delhi Cantonment	0.003874
72	Dadri	0.002267
73	Aurangabad	0.002267
74	Midnapore	0.002242

75 rows × 2 columns

- **Top 10 Node có PageRank cao nhất**


```
#Top 10 pagerank cao nhất
top_10_max_page = sorted_pagerank[:10]

print('Top 10 node có Pagerank cao nhất: ')
print_table(top_10_max_page, ['Node', 'Pagerank'])
```

→ Top 10 node có Pagerank cao nhất:

	Node	Pagerank
0	Pune	0.021702
1	Bengaluru	0.021173
2	Hyderabad	0.018362
3	Mumbai	0.018239
4	Gurugram	0.018162
5	Noida	0.017952
6	New Delhi	0.017908
7	Chennai	0.017887
8	Bangalore Urban	0.017802
9	Delhi	0.017540

- Top 10 Node có PageRank thấp nhất

```
#Top 10 pagerank thap nhat
top_10_min_page = sorted_pagerank[-10:]

print('Top 10 node co Pagerank thap nhat: ')
print_table(top_10_min_page, ['Node', 'Pagerank'])
```

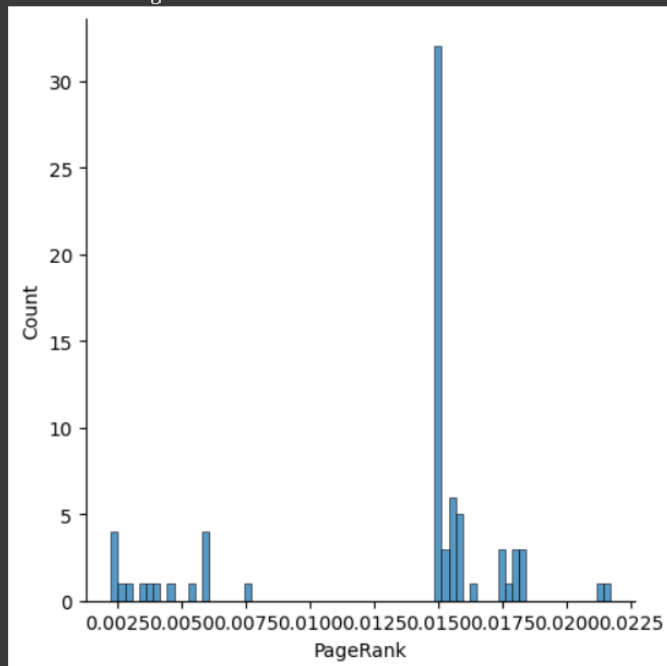
→ Top 10 node co Pagerank thap nhat:

	Node	Pagerank
0	Alipur	0.004575
1	Silao	0.003890
2	Delhi Cantonment	0.003874
3	Ernakulam	0.003366
4	Raipur	0.002947
5	Ahmednagar	0.002718
6	Dadri	0.002267
7	Aurangabad	0.002267
8	Gandhinagar	0.002257
9	Midnapore	0.002242

- Biểu đồ phân bố PageRank

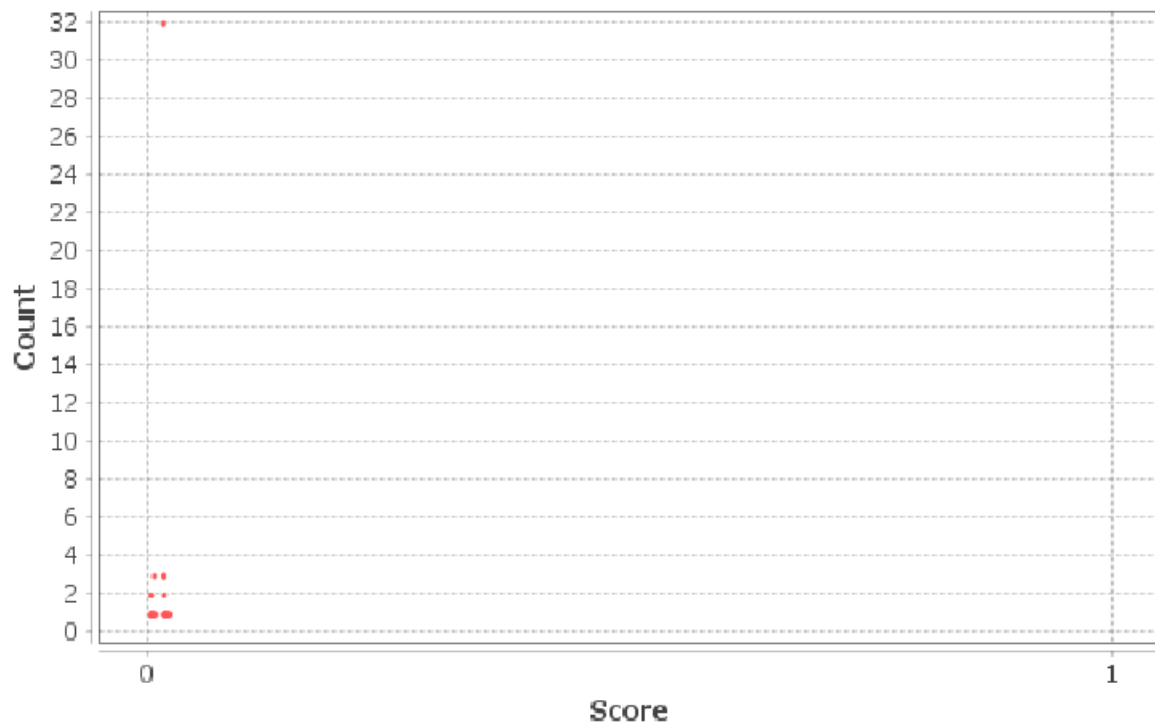
```
[ ] #biểu đồ phân bố pagerank
print(colored("Biểu đồ thể hiện phân bố pagerank của các node trong graph", 'green', attrs=['bold']))
sns.displot(x='PageRank', data=pagerank_df)
```

Biểu đồ thể hiện phân bố pagerank của các node trong graph
 <seaborn.axisgrid.FacetGrid at 0x7cd9a0a705b0>



3.5.2 Gephi

PageRank Distribution



- **Top 10 Node có PageRank cao nhất**

Id	Label	Interval	PageRank ▾
Pune			0.021702
Bengaluru			0.021173
Hyderabad			0.018362
Mumbai			0.018239
Gurugram			0.018162
Noida			0.017952
New Delhi			0.017908
Chennai			0.017887
Bangalore Ur...			0.017802
Delhi			0.01754

- **Top 10 Node có PageRank thấp nhất**

Id	Label	Interval	PageRank ^
Midnapore			0.002242
Gandhinagar			0.002257
Aurangabad			0.002267
Dadri			0.002267
Ahmednagar			0.002718
Raipur			0.002947
Ernakulam			0.003366
Delhi Canton...			0.003874
Silao			0.00389
Alipur			0.004575

CHƯƠNG IV: THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG

4.1 Thuật toán Girvan Newman

Thuật toán lần đầu tiên được đề xuất bởi Freeman. Theo Freeman, các cạnh được coi là cạnh có số lượng con đường ngắn nhất giữa các cặp đỉnh khác nhau chạy qua nó. Cạnh nối có ảnh hưởng rất lớn đến dòng chảy của thông tin giữa các nút khác, đặc biệt là trong trường hợp thông tin lưu truyền trong mạng chủ yếu theo con đường ngắn nhất. Thuật toán điển hình nhất trong các thuật toán chia này là thuật toán Girvan-Newman. Để tìm các cạnh trong mạng nối hai đỉnh thuộc hai cộng đồng khác nhau, khái quát đây là cạnh có độ trung gian cao, và xác định độ đo trung gian này bằng cách tính số đường đi ngắn nhất giữa các cặp đỉnh mà có qua nó. Với một đồ thị m cạnh và n 13 đỉnh thì thời gian tính toán cho giai đoạn này là $O(mn)$. Với đồ thị có trọng số, độ đo trung gian của cạnh có trọng số đơn giản được tính bằng độ đo trung gian của cạnh không có trọng số chia cho trọng số của cạnh đó. Nếu một mạng lưới bao gồm

các cộng đồng hoặc nhóm chúng chỉ được liên kết nối yếu bằng một nhóm cạnh, thì tất cả các đường đi ngắn nhất giữa các cộng đồng khác nhau sẽ phải đi dọc theo một trong số ít các cạnh thuộc nhóm cạnh đó. Vì vậy, các cạnh kết nối các cộng đồng sẽ là cạnh có độ đo trung gian cao. Bằng cách loại bỏ các cạnh, thuật toán Girvan-Newman tách được thành các nhóm riêng biệt.

Thuật toán được thực hiện theo các bước sau:

1. Tính độ đo trung gian cho tất cả các cạnh trong mạng.
2. Hủy bỏ các cạnh có độ trung gian cao nhất.
3. Tính lại độ trung gian cho tất cả các cạnh bị ảnh hưởng theo các cạnh đã loại bỏ.
4. Lặp lại từ bước 2 cho đến khi không còn các cạnh trung gian.

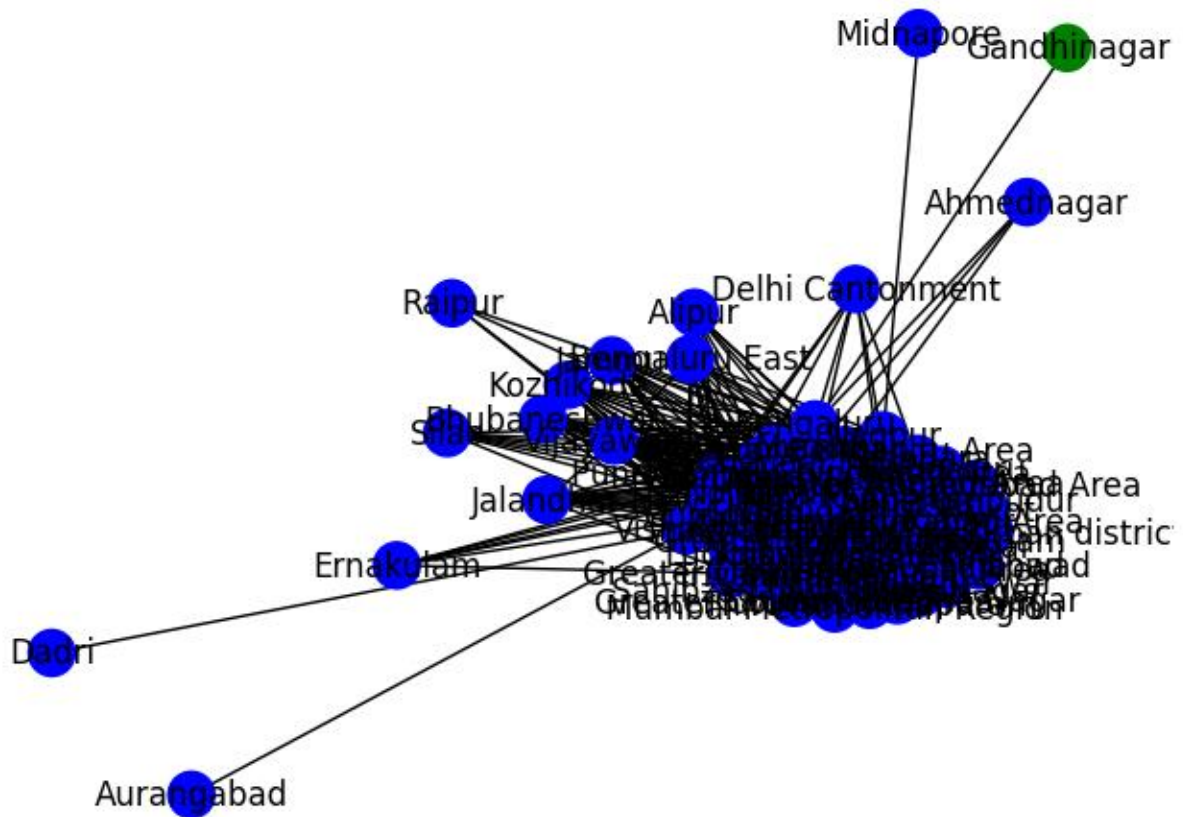
- **Code cài đặt và sử dụng thuật toán:**

```
[ ] dt_girvan = df.copy()
dt_girvan

from networkx.algorithms.community centrality import girvan_newman
G_Girvan = G.copy()
communities = girvan_newman(G_Girvan)
node_groups = []
for com in next(communities):
    node_groups.append(list(com))

color_map = []
for node in G_Girvan:
    if node in node_groups[0]:
        color_map.append('blue')
    elif node in node_groups[1]:
        color_map.append('green')
    else:
        color_map.append('black')
nx.draw(G_Girvan, node_color=color_map, with_labels=True)
plt.show()
```

- **Đồ thị phân cụm sử dụng Girvan Newman**



- In ra số lượng Cluster

```
# Code remove edge
def edge_to_remove(graph):
    G_dict = nx.edge_betweenness centrality(graph)
    edge = ()

    #extract the edge with highest edge betweenness centrality score
    for key, value in sorted(G_dict.items(), key= lambda item: item[1], reverse=True):
        edge = key
        break

    return edge

#code define hàm Girvan Newman
def girvan_newman(graph):
    #find number of connected components
    sg = nx.connected_components(graph)
    sg_count = nx.number_connected_components(graph)

    while(sg_count == 1):
        graph.remove_edge(edge_to_remove(graph)[0], edge_to_remove(graph)[1])
        sg = nx.connected_components(graph)
        sg_count = nx.number_connected_components(graph)
    return sg

#code define communities
# find communities in the graph
c = girvan_newman(G.copy())

# find the nodes forming the communities
node_groups = []

for i in next(c):
    node_groups.append(list(i))

print("Phan cum")
print(node_groups)
```

Phan cum
[['Pimpri Chinchwad', 'Alipur', 'Gurugram', 'Bengaluru East', 'Dadri', 'Thane', 'Chandigarh', 'Lucknow', 'Jammu', 'Agra', 'Mohali district', 'Greater Delhi Area', 'Greater Hyderabad Area', 'Visakhapatnam', 'Ghaziabad', 'Vishakhapatnam', 'Jalandhar', 'Greater Kolkata Area', 'Bengaluru', 'New Delhi', 'Ludhiana', 'Dehradun', 'Mumbai Metropolitan Region', 'Ernakulam', 'Navi Mumbai', 'Gurgaon', 'Delhi Cantonment', 'Greater Ahmedabad Area', 'Kolkata', 'Bangalore Urban', 'Pune/Pimpri-Chinchwad Area', 'Vijayawada', 'Gautam Buddha Nagar', 'Silao', 'Aurangabad', 'Kozhikode', 'Jamshedpur', 'Faridabad', 'Srinagar', 'Ahmednagar', 'Mangaluru', 'Bhubaneswar', 'Bhubaneshwar', 'Patna', 'Mumbai', 'Midnapore', 'Greater Bengaluru Area', 'Noida', 'Bhopal', 'Greater Chennai Area', 'Thiruvananthapuram', 'Hyderabad', 'Sriperumbudur']]

- Danh sách các cụm: 2 cụm

Cụm 1: Pimpri Chinchwad | Alipur | Gurugram | Bengaluru East | Dadri | Thane | Chandigarh | Lucknow | Jammu | Agra | Mohali district | Greater Delhi Area | Greater Hyderabad Area | Visakhapatnam | Ghaziabad | Vishakhapatnam | Jalandhar | Greater Kolkata Area | Bengaluru | New Delhi | Ludhiana | Dehradun | Mumbai Metropolitan Region | Ernakulam | Navi Mumbai | Gurgaon | Delhi Cantonment | Greater Ahmedabad Area | Kolkata | Bangalore Urban | Pune/Pimpri-Chinchwad Area | Vijayawada | Gautam Buddha Nagar | Silao | Aurangabad | Kozhikode | Jamshedpur | Faridabad | Srinagar | Ahmednagar | Mangaluru | Bhubaneswar | Bhubaneshwar | Patna | Mumbai | Midnapore | Greater Bengaluru Area | Noida | Bhopal | Greater Chennai Area | Thiruvananthapuram | Hyderabad | Sriperumbudur |

Bangalore Urban district | Kanpur | Coimbatore | Chennai | Trivandrum | Amritsar | Greater Coimbatore Area | Nagpur | Surat | Kochi | Kalyan | Indore | Sahibzada Ajit Singh Nagar | Delhi | Ahmedabad | Vadodara | Pune | Rajkot | Raipur | Jaipur | Greater Nagpur Area |

Cụm 2: | Gandhinagar |

4.2 Thuật toán Louvain

Phương pháp Louvain để phát hiện cộng đồng là một thuật toán để phát hiện các cộng đồng trong mạng. Nó tối đa hóa điểm mô-đun cho mỗi cộng đồng, trong đó mô-đun định lượng chất lượng của việc gán các nút cho cộng đồng. Điều này có nghĩa là đánh giá mức độ kết nối của các nút trong cộng đồng với mật độ cao hơn như thế nào so với mức độ kết nối của chúng trong một mạng ngẫu nhiên. Thuật toán Louvain là một thuật toán phân cụm phân cấp, hợp nhất một cách đệ quy các cộng đồng thành một nút duy nhất và thực hiện phân cụm mô-đun trên các đồ thị cô đọng

- Code cài đặt và sử dụng thuật toán:

```
[ ] #Bieu do
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import networkx as nx
import numpy as np
import community

dt = pd.DataFrame(df)

G_louvain = G.copy()

from community import community_louvain
#compute the best partition
partition = community_louvain.best_partition(G_louvain, resolution=0.95)

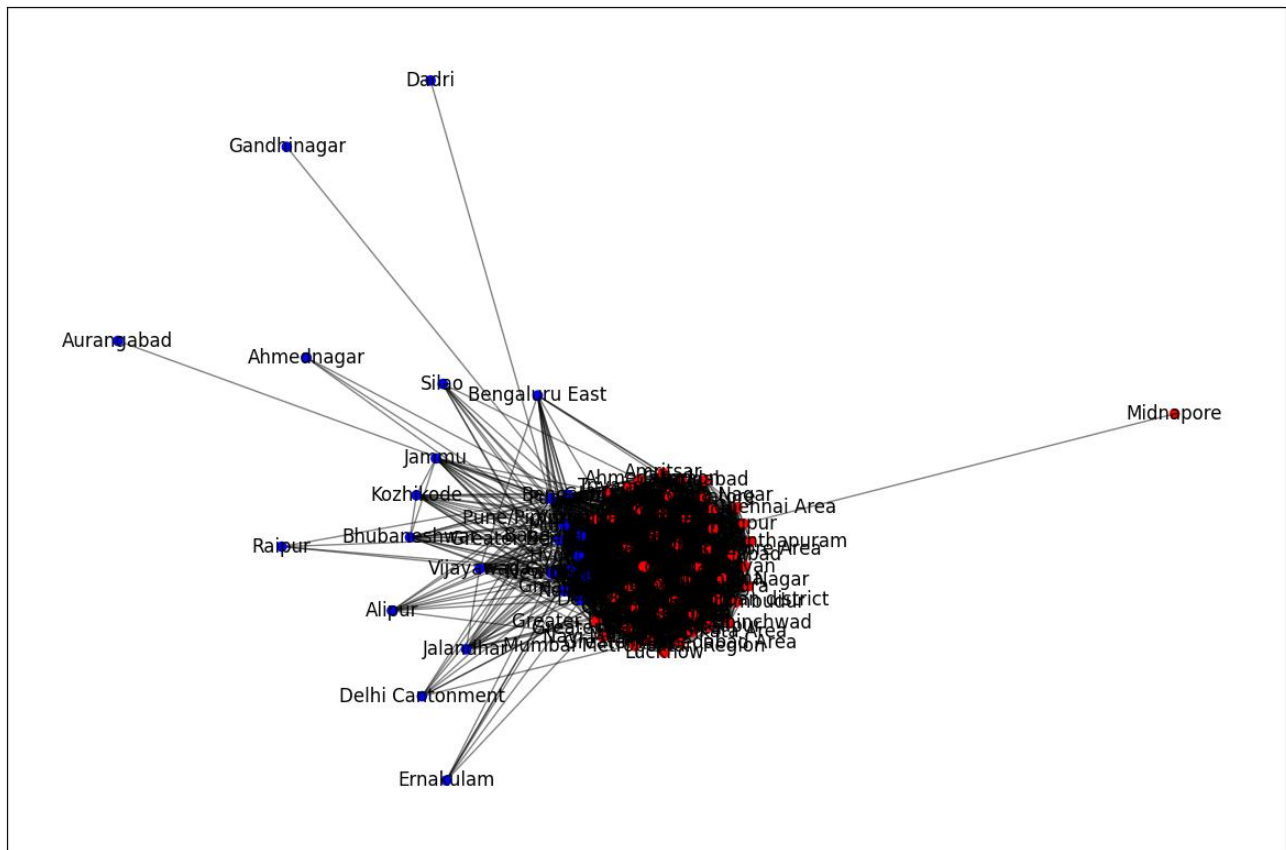
custom_colors = ['red', 'green', 'blue']

pos = nx.spring_layout(G_louvain)

fig = plt.figure(figsize=(15,10), dpi = 100)
cmap = colors.ListedColormap(custom_colors, N=len(custom_colors))
nx.draw_networkx_nodes(G_louvain, pos, partition.keys(), node_size = 35, cmap=cmap, node_color = list(partition.values()))
nx.draw_networkx_edges(G_louvain, pos, alpha = 0.5)
nx.draw_networkx_labels(G_louvain, pos)

plt.show()
```

- Đồ thị phân cụm sử dụng Louvain



- In ra số lượng cụm

```
[ ] clusters = set([cluster for _, cluster in partition.items()])
print('Số lượng cộng đồng Louvain: ', len(clusters))
```

Số lượng cộng đồng Louvain: 2


```
dt = pd.DataFrame(df)

from community import community_louvain
#compute the best partition
partition = community_louvain.best_partition(G_louvain, resolution=0.95)

clusters = set([cluster for _, cluster in partition.items()])
print('Số lượng cộng đồng Louvain: ', len(clusters))

values = list(partition.values())
groups = []
print("Phân cụm")
for i in range(len(np.unique(values))):
    print("Cụm", i)
    values = []
    for name, k in partition.items():
        if k == i:
            print(name, end = ' | ')
            values.append(name)
    groups.append(values) #Thêm vào group
print()
print()
```

- **Danh sách các cụm: 2 cụm**

 Số lượng cộng đồng Louvain: 2
 Phân cụm
 Cụm 0
 Delhi | Alipur | Gurugram | Jalandhar | Bengaluru | Ernakulam | Bangalore Urban | Kozhikode | Bhubaneswar | Mumbai | Hyderabad
 Cụm 1
 Jaipur | Thane | Lucknow | Agra | Mohali district | Greater Delhi Area | Greater Kolkata Area | Ludhiana | Dehradun | Gurgaon

Cụm 1: Delhi | Alipur | Gurugram | Jalandhar | Bengaluru | Ernakulam | Bangalore Urban | Kozhikode | Bhubaneswar | Mumbai | Hyderabad | Chennai | Pune | Vijayawada | Bengaluru East | New Delhi | Greater Bengaluru Area | Noida | Jammu | Silao | Raipur | Ahmednagar | Gandhinagar | Delhi Cantonment | Dadri | Aurangabad |

Cụm 2: Jaipur | Thane | Lucknow | Agra | Mohali district | Greater Delhi Area | Greater Kolkata Area | Ludhiana | Dehradun | Gurgaon | Greater Ahmedabad Area | Pune/Pimpri-Chinchwad Area | Gautam Buddha Nagar | Jamshedpur | Mangaluru | Bhubaneswar | Greater Chennai Area | Amritsar | Greater Nagpur Area | Surat | Kochi | Sahibzada Ajit Singh Nagar | Ahmedabad | Vadodara | Thiruvananthapuram | Pimpri Chinchwad | Chandigarh | Greater Hyderabad Area | Visakhapatnam | Ghaziabad | Vishakhapatnam | Mumbai Metropolitan Region | Navi Mumbai | Kolkata | Faridabad | Srinagar | Patna | Bhopal | Sriperumbudur | Bangalore Urban district | Kanpur | Trivandrum | Greater Coimbatore Area | Nagpur | Kalyan | Indore | Rajkot | Coimbatore | Midnapore |

- **Xuất file Excel**

```

louvain = pd.DataFrame.from_dict(partition, orient='index', columns = ['Cluster'])
louvain.index.name = 'Node'
# louvain.reset_index(inplace=True)
louvain

```

	cluster
Node	
Delhi	0
Alipur	0
Jaipur	1
Gurugram	0
Thane	1
...	...
Gandhinagar	0
Delhi Cantonment	0
Dadri	0
Aurangabad	0
Midnapore	1

75 rows × 1 columns

```

merged_data = pd.merge(dt, louvain, left_on = 'city', right_on = 'Node')
merged_data

```

	industry	city	cluster
0	IT Services and IT Consulting	Delhi	0
1	Information Technology & Services	Delhi	0
2	Government Administration	Delhi	0
3	Internet Marketplace Platforms	Delhi	0
4	Business Consulting and Services	Delhi	0
...
397	Wholesale Building Materials	Midnapore	1
398	E-Learning Providers	Raipur	0
399	Machinery Manufacturing	Gandhinagar	0
400	IT Services and IT Consulting	Gautam Buddha Nagar	1
401	IT Services and IT Consulting	Sriperumbudur	1

402 rows × 3 columns

```
[41] from google.colab import drive
drive.mount('drive')
merged_data.to_csv("job_Louvain.csv", index = False, header = True, encoding = "utf-8")
!cp job_Louvain.csv "/content/drive/MyDrive/Colab Notebooks/"
```

Drive already mounted at drive; to attempt to forcibly remount, call drive.mount("drive", force_remount=True).

	A	B	C	D	E	F	G
1	industry	City	Cluster				
1816	IT Services and IT Consulting	Kolkata	1				
1817	IT Services and IT Consulting	Kolkata	1				
1818	IT Services and IT Consulting	Kolkata	1				
1819	IT Services and IT Consulting	Kolkata	1				
1820	IT Services and IT Consulting	Kolkata	1				
1821	IT Services and IT Consulting	Kolkata	1				
1822	IT Services and IT Consulting	Kolkata	1				
1823	IT Services and IT Consulting	Kolkata	1				
1824	IT Services and IT Consulting	Kolkata	1				
1825	IT Services and IT Consulting	Kolkata	1				
1826	IT Services and IT Consulting	Kolkata	1				
1827	IT Services and IT Consulting	Kolkata	1				
1828	IT Services and IT Consulting	Kolkata	1				
1829	IT Services and IT Consulting	Kolkata	1				
1830	IT Services and IT Consulting	Kolkata	1				
1831	IT Services and IT Consulting	Kolkata	1				
1832	IT Services and IT Consulting	Kolkata	1				
1833	IT Services and IT Consulting	Kolkata	1				
1834	IT Services and IT Consulting	Kolkata	1				
1835	IT Services and IT Consulting	Kolkata	1				
1836	IT Services and IT Consulting	Kolkata	1				
1837	IT Services and IT Consulting	Kolkata	1				
1838	IT Services and IT Consulting	Kolkata	1				
1839	IT Services and IT Consulting	Kolkata	1				
1840	IT Services and IT Consulting	Kolkata	1				
1841	IT Services and IT Consulting	Kolkata	1				
1842	IT Services and IT Consulting	Kolkata	1				
1843	IT Services and IT Consulting	Kolkata	1				
1844	IT Services and IT Consulting	Kolkata	1				
1845	IT Services and IT Consulting	Kolkata	1				

< >

job_Louvain

Cluster0

Cluster1

+

4.3 Thuật toán K-Means

Trong thuật toán k-Means mỗi cụm dữ liệu được đặc trưng bởi một tâm (centroid). tâm là điểm đại diện nhất cho một cụm và có giá trị bằng trung bình của toàn bộ các quan sát nằm trong cụm. Chúng ta sẽ dựa vào khoảng cách từ mỗi quan sát tới các tâm để xác định nhãn cho chúng trùng thuộc về tâm gần nhất. Ban đầu thuật toán sẽ khởi tạo ngẫu nhiên một số lượng xác định trước tâm cụm. Sau đó tiến hành xác định nhãn cho từng điểm dữ liệu và tiếp tục cập nhật lại tâm cụm. Thuật toán sẽ dừng cho tới khi toàn bộ các điểm dữ liệu được phân về đúng cụm hoặc số lượt cập nhật tâm chạm ngưỡng.

Cụ thể các bước của thuật toán k-Means được tóm tắt như sau:

- + Khởi tạo ngẫu nhiên k tâm cụm $\mu_1, \mu_2, \dots, \mu_k$.
- + Lặp lại quá trình cập nhật tâm cụm cho tới khi dừng:
 - a. Xác định nhãn cho từng điểm dữ liệu c_i dựa vào khoảng cách tới từng tâm cụm:

$$c_i = \arg \min_j \|\mathbf{x}_i - \mu_j\|_2^2$$

b. Tính toán lại tâm cho từng cụm theo trung bình của toàn bộ các điểm dữ liệu trong một cụm:

$$\mu_j := \frac{\sum_{i=1}^n \mathbf{1}(c_i = j) \mathbf{x}_i}{\sum_{i=1}^n \mathbf{1}(c_i = j)}$$

Trong công thức 2.a thì $\|\mathbf{x}\|_2^2$ là bình phương của norm chuẩn bậc 2, kí hiệu là L2, norm chuẩn bậc 2 là một độ đo khoảng cách thường được sử dụng trong machine learning.

Trong công thức 2.b chúng ta sử dụng hàm $\mathbf{1}(\cdot)$, hàm này có giá trị trả về là 1 nếu nhãn của điểm dữ liệu c_i được dự báo thuộc về cụm j , trái lại thì trả về giá trị 0. Như vậy tử số của vế phải trong công thức 2.b chính là tổng khoảng cách của toàn bộ các điểm dữ liệu nằm trong cụm j trong khi mẫu số chính là số lượng các điểm dữ liệu thuộc cụm j . μ_j chính là vị trí của tâm cụm j mà ta dự báo tại thời điểm hiện tại. Trong thuật toán trên thì tham số mà chúng ta cần lựa chọn chính là số lượng cụm k . Thời điểm ban đầu ta sẽ khởi tạo k điểm dữ liệu một

cách ngẫu nhiên và sau đó gán các tâm bằng giá trị của k điểm dữ liệu này. Các bước trong vòng lặp ở bước 2 thực chất là:

- a. Gán nhãn cho mỗi điểm dữ liệu bằng với nhãn của tâm cụm gần nhất.
- b. Dịch chuyển dần dần tâm cụm μ_j tới trung bình của những điểm dữ liệu mà được phân về j.

- **Code cài đặt và sử dụng thuật toán:**

Chuyển đổi cột City và Industry thành encode

```
# import preprocessing
# chuyển đổi cột City thành encode
from sklearn import preprocessing

# Sử dụng phương thức LabelEncoder để chuyển đổi dữ liệu chữ thành số

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dt['CityID'] = le.fit_transform(dt['City'])

[48] # Chuyển đổi cột Industry thành encode

le = LabelEncoder()
dt['IndustryID'] = le.fit_transform(dt['industry'])
```

dt

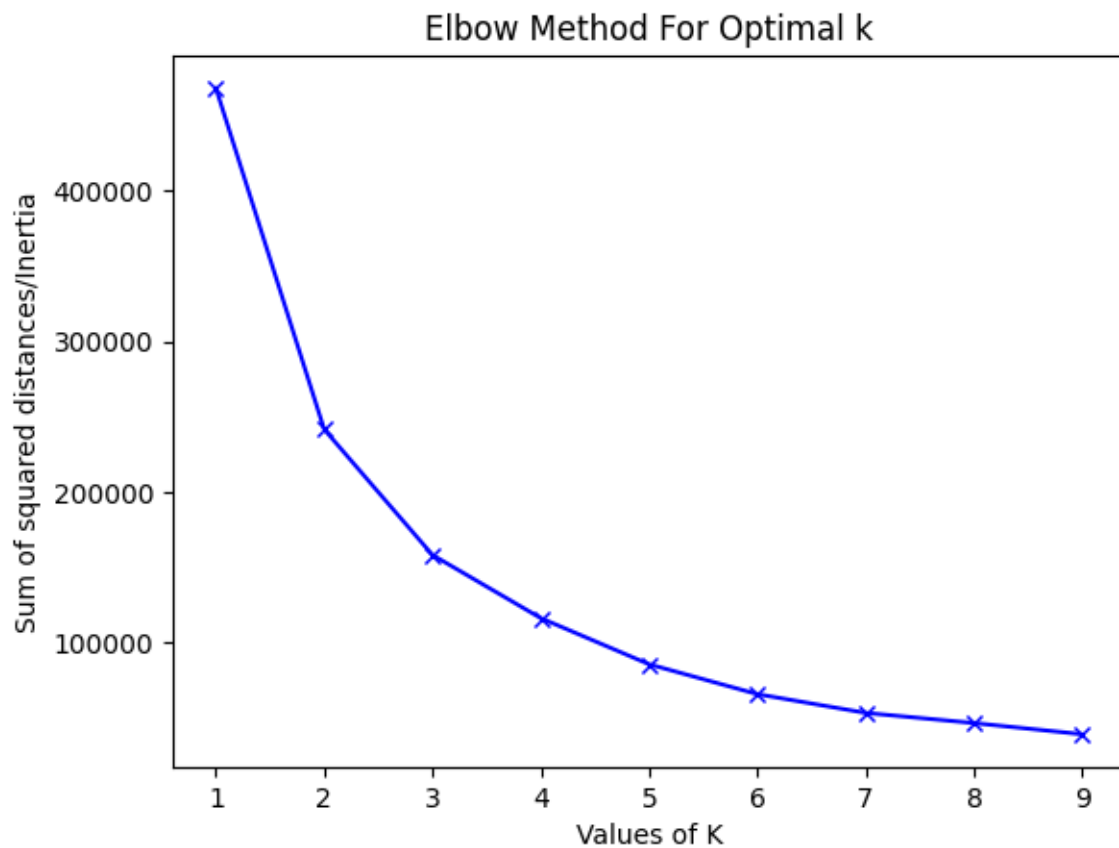
	industry	city	CityID	IndustryID
0	IT Services and IT Consulting	Delhi	18	42
1	IT Services and IT Consulting	New Delhi	55	42
2	IT Services and IT Consulting	Greater Bengaluru Area	26	42
5	Telecommunications	Gurugram	34	92
6	IT Services and IT Consulting	Bengaluru	8	42
...
5532	IT Services and IT Consulting	Sriperumbudur	66	42
5537	Software Development	Kochi	43	87
5549	Business Consulting and Services	Greater Kolkata Area	31	15
5559	Technology Information and Internet	Navi Mumbai	54	91
5565	Hospitals and Health Care	Bangalore Urban	6	39

403 rows × 4 columns

- Sử dụng phương pháp elbow để tìm số cụm cho thuật toán Kmeans

```
[50] #Dùng phương pháp elbow để tìm ra số cụm cho thuật toán KMeans
import numpy as np
from sklearn.cluster import KMeans

sum_of_squared_distances = []
K = range(1, 10)
for num_clusters in K:
    kmeans = KMeans(n_clusters = num_clusters)
    kmeans.fit(dt[['CityID', 'IndustryID']])
    sum_of_squared_distances.append(kmeans.inertia_)
plt.plot(K, sum_of_squared_distances, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```

- Tìm ra số cụm là 3

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

algorithm = KMeans(n_clusters = 3 )
algorithm
```

→ KMeans
KMeans(n_clusters=3)

```

y_predicted = algorithm.fit_predict(dt[['CityID', 'IndustryID']])
y_predicted

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 1 in the future. This will affect the results of the fit method.
warnings.warn(
array([0, 1, 0, 2, 0, 2, 1, 1, 1, 2, 1, 2, 0, 1, 0, 1, 2, 1, 1, 1, 0, 2,
       0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 2, 0, 1, 0, 2, 2, 0, 0, 0,
       1, 2, 1, 0, 0, 1, 1, 1, 0, 2, 2, 1, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2,
       0, 1, 1, 1, 0, 0, 1, 2, 2, 1, 1, 1, 0, 0, 2, 2, 2, 0, 2, 2, 0, 2,
       2, 1, 0, 0, 1, 1, 1, 1, 2, 1, 0, 2, 2, 2, 0, 1, 2, 0, 2, 2, 0, 1,
       0, 2, 2, 2, 0, 0, 2, 0, 2, 0, 0, 2, 2, 1, 0, 0, 1, 1, 1, 2, 0, 2,
       2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 0, 0, 0,
       1, 1, 2, 2, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 2, 0, 1, 2, 0, 1, 0, 0, 0, 1, 1, 2, 1, 1, 2, 0, 1,
       2, 2, 2, 2, 2, 0, 2, 2, 1, 2, 1, 0, 0, 2, 0, 2, 2, 2, 2, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 2, 0, 2, 0, 1, 1, 0, 1, 0, 2, 0,
       1, 0, 1, 0, 2, 2, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2,
       1, 0, 2, 0, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 2, 2, 2,
       2, 1, 1, 0, 2, 1, 2, 2, 2, 2, 1, 1, 1, 0, 0, 2, 1, 2, 2, 1, 0,
       0, 2, 2, 1, 1, 0, 0, 0, 1, 2, 1, 0, 2, 2, 2, 0, 0, 0, 0, 1, 1,
       0, 2, 2, 0, 1, 2, 1, 1, 2, 1, 1, 1, 1, 0, 2, 2, 2, 1, 1, 2, 0, 0,
       2, 0, 2, 0, 2, 2, 1, 0, 2, 1, 1, 2, 0, 1, 2, 1, 0, 0, 0, 1, 0, 1,
       0, 2, 2, 0, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 0, 2, 0, 1,
       1, 2, 2, 0, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 0, 2, 0, 1,
       1, 2, 1, 2, 0, 2, 0], dtype=int32)

```

- Xuất file Excel

```

dt['cluster'] = y_predicted
dt

```

	industry	City	CityID	IndustryID	cluster
0	IT Services and IT Consulting	Delhi	18	42	0
1	IT Services and IT Consulting	New Delhi	55	42	1
2	IT Services and IT Consulting	Greater Bengaluru Area	26	42	0
5	Telecommunications	Gurugram	34	92	2
6	IT Services and IT Consulting	Bengaluru	8	42	0
...
5532	IT Services and IT Consulting	Sriperumbudur	66	42	1
5537	Software Development	Kochi	43	87	2
5549	Business Consulting and Services	Greater Kolkata Area	31	15	0
5559	Technology Information and Internet	Navi Mumbai	54	91	2
5565	Hospitals and Health Care	Bangalore Urban	6	39	0

403 rows × 5 columns

```

[54] from google.colab import drive
drive.mount('drive')
dt.to_csv("job_kmeans.csv", index = False, header = True, encoding = "utf-8")
!cp job_kmeans.csv "/content/drive/MyDrive/Colab Notebooks/"

```

Drive already mounted at drive; to attempt to forcibly remount, call drive.mount("drive", force_remount=True).

	A	B	C	D	E	F	G	H
1	industry	City	cluster					
5	Telecommunications	Gurugram	2					
7	Renewable Energy Semiconductor Manufacturing	Gurugram	2					
1	Technology Information and Internet	Bengaluru	2					
3	Pharmaceutical Manufacturing	Bengaluru	2					
8	Staffing and Recruiting	Mumbai	2					
3	Software Development	Bengaluru	2					
7	Outsourcing and Offshoring Consulting	Bengaluru	2					
1	Software Development	Pune	2					
2	Software Development	Hyderabad	2					
7	Technology Information and Internet	Greater Bengaluru Area	2					
5	Retail	Bangalore Urban	2					
6	Staffing and Recruiting	Bengaluru	2					
8	Retail Groceries	Bengaluru	2					
9	Retail	Bengaluru	2					
1	Software Development	Chennai	2					
3	Telecommunications	Kochi	2					
4	Software Development	Bangalore Urban	2					
5	Software Development	Greater Bengaluru Area	2					
7	Staffing and Recruiting	Pune	2					
5	Research Services	Bengaluru	2					
6	Oil and Gas	Indore	2					
2	Technology Information and Internet	Noida	2					
3	Software Development	Noida	2					
4	Semiconductors	Hyderabad	2					
6	Staffing and Recruiting	Hyderabad	2					

- Tọa độ trung tâm cụm



```

noteGenre = dt.drop_duplicates(subset='CityID', keep="last")
print('S6 City: ',len(noteGenre[['City','CityID']]))
print(noteGenre[['City','CityID']].to_string())

```

S6 City: 76

	City	CityID
118	Bhopal	18
122	Surat	67
136	Delhi Cantonment	19
186	Kalyan	41
205	Ludhiana	47
208	Amritsar	4
209	Patna	57
257	Thiruvananthapuram	69
462	Lucknow	46
486	Vapi	72
501	Sahibzada Ajit Singh Nagar	63
779	Kozhikode	45
848	Kanpur	42
1124	Pimpri Chinchwad	58
1259	Greater Coimbatore Area	28
1315	Bhubaneswar	11
1483	Bangalore Urban district	7
1498	Srinagar	65
1680	Aurangabad	5
1757	Bengaluru East	9
1805	Visakhapatnam	74
1941	Allipur	3
2130	Ernakulam	20
2299	Thane	68
2811	Vadodara	71
2826	Jammu	39
2839	Greater Ahmedabad Area	25
2841	Greater Chennai Area	27
3003	Jamshedpur	40
3019	Bhubaneswar	12
3082	Trivandrum	70
3092	Agra	0
3135	Nagpur	53
3163	Pune/Pimpri-Chinchwad Area	60
3201	Faridabad	21
3362	Vijayawada	73
3483	Greater Nagpur Area	32
3910	Indore	36

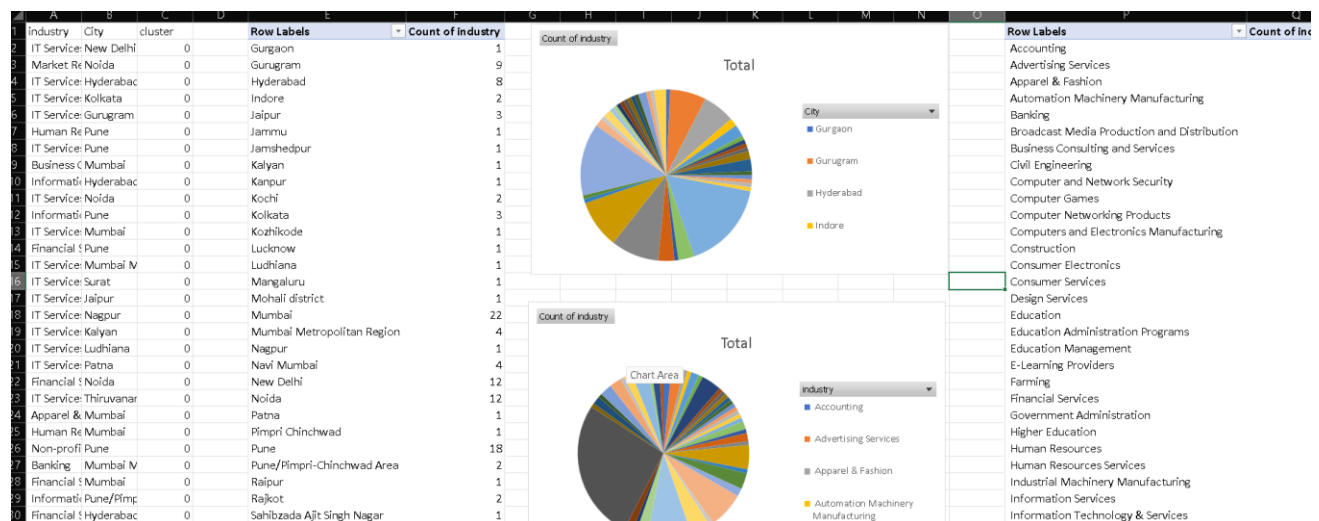
```

noteGenre = dt.drop_duplicates(subset='Industry', keep="last")
print('S6 Industry: ',len(noteGenre[['Industry','IndustryID']]))
print(noteGenre[['Industry','IndustryID']].to_string())

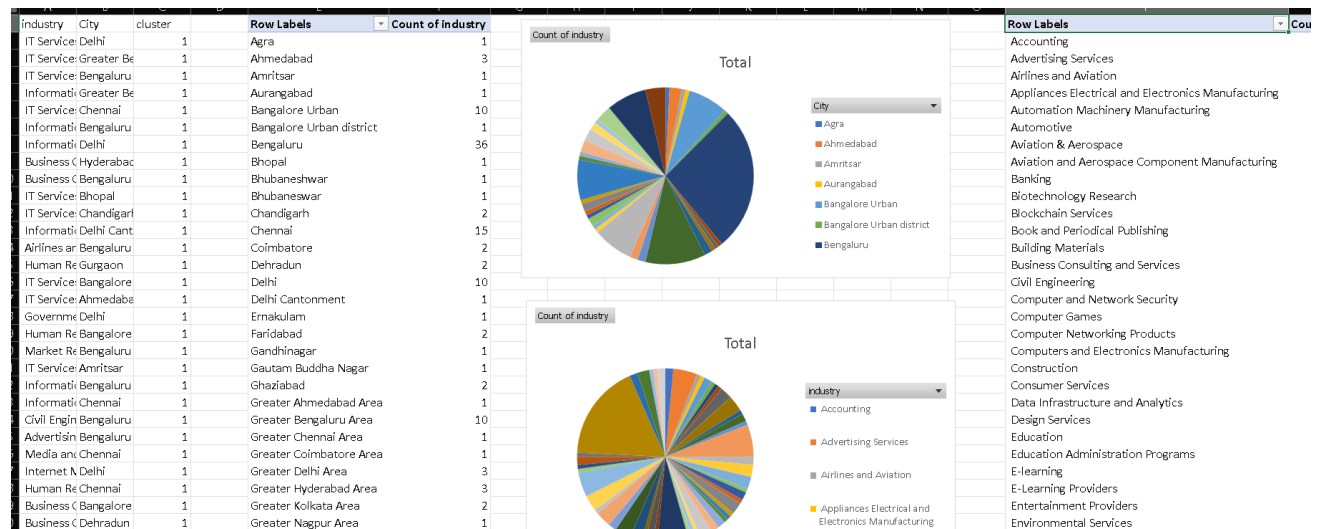
```

S6 Industry: 102

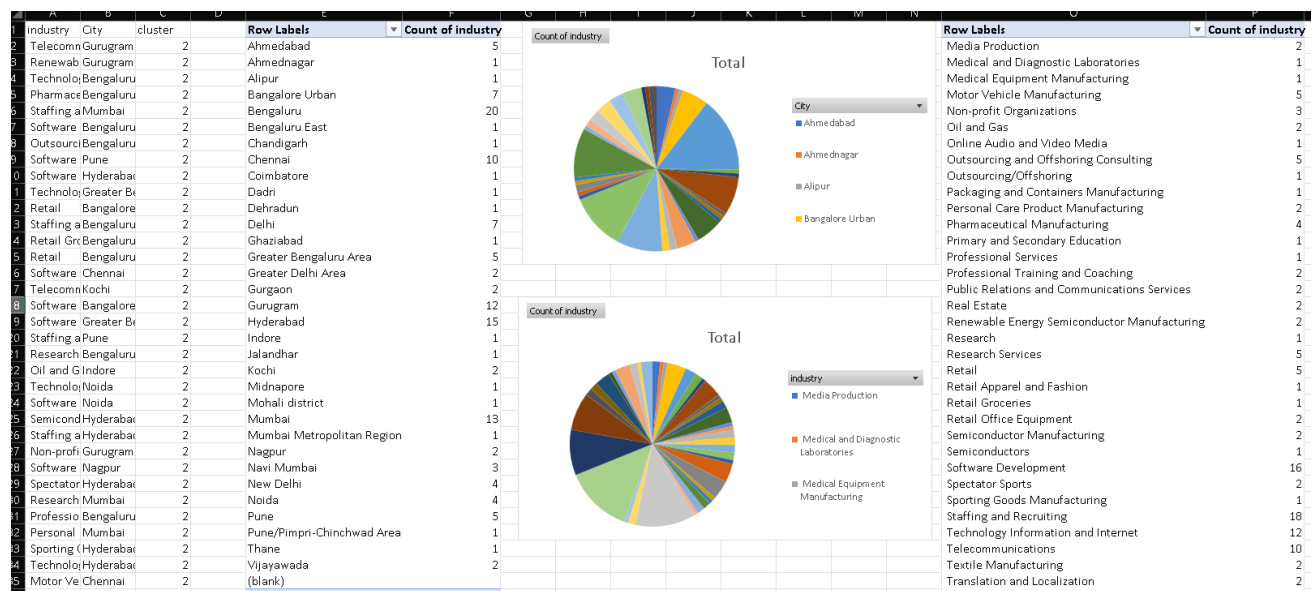
	Industry	IndustryID
145	Airlines and Aviation	2
267	Apparel & Fashion	3
268	Retail Groceries	83
343	Internet Marketplace Platforms	48
412	Semiconductors	86
486	Medical Device	60
505	Law Practice	51
512	Sporting Goods Manufacturing	89
530	Maritime Transportation	55
554	Professional Training and Coaching	75
584	Book and Periodical Publishing	12
601	Entertainment Providers	31
741	Research	79
844	Data Infrastructure and Analytics	24
1211	Aviation & Aerospace	7
1242	Computer Networking Products	18
1292	Food and Beverage Manufacturing	35
1513	Appliances Electrical and Electronics Manufacturing	4
1861	Wireless Services	101
1906	Blockchain Services	11
1909	Packaging and Containers Manufacturing	70
2127	Automotive	6
2849	Consumer Electronics	22
2855	Aviation and Aerospace Component Manufacturing	8
2883	Information Technology & Services	45
2978	Medical and Diagnostic Laboratories	62
3026	Education	28
3059	Semiconductor Manufacturing	85
3213	Investment Management	50
3321	Internet Publishing	49
3355	Professional Services	74
3360	International Trade and Development	47
3376	Education Management	30
3474	Construction	21
3514	Utilities	97
3515	Outsourcing and Offshoring Consulting	68
3610	Oil and Gas	66



→ **Ta thấy ở Cluster 0:** Ngành nghề IT Services and IT Consulting được đăng tuyển nhiều nhất ở thành phố Mumbai



→ **Ta thấy ở Cluster 1:** Ngành nghề IT Services and IT Consulting được đăng tuyển nhiều nhất ở thành phố Bengaluru

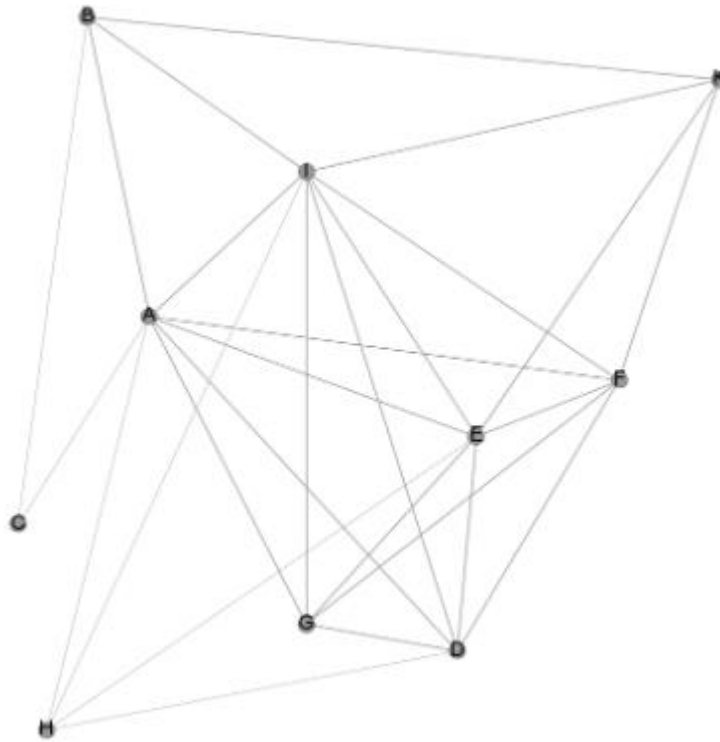


→ Ta thấy ở Cluster 2: Ngành nghề IT Services and IT Consulting được đăng tuyển nhiều nhất ở thành phố Bengaluru

Kết luận: Qua 3 thuật toán phát hiện cộng đồng chúng ta nhận ra rằng thuật toán K-Means sử dụng hiệu quả nhất vì thuật toán đã chia nhỏ ra từng cụm và chỉ rõ chi tiết đặc trưng của dataset hơn so với thuật toán Louvain.

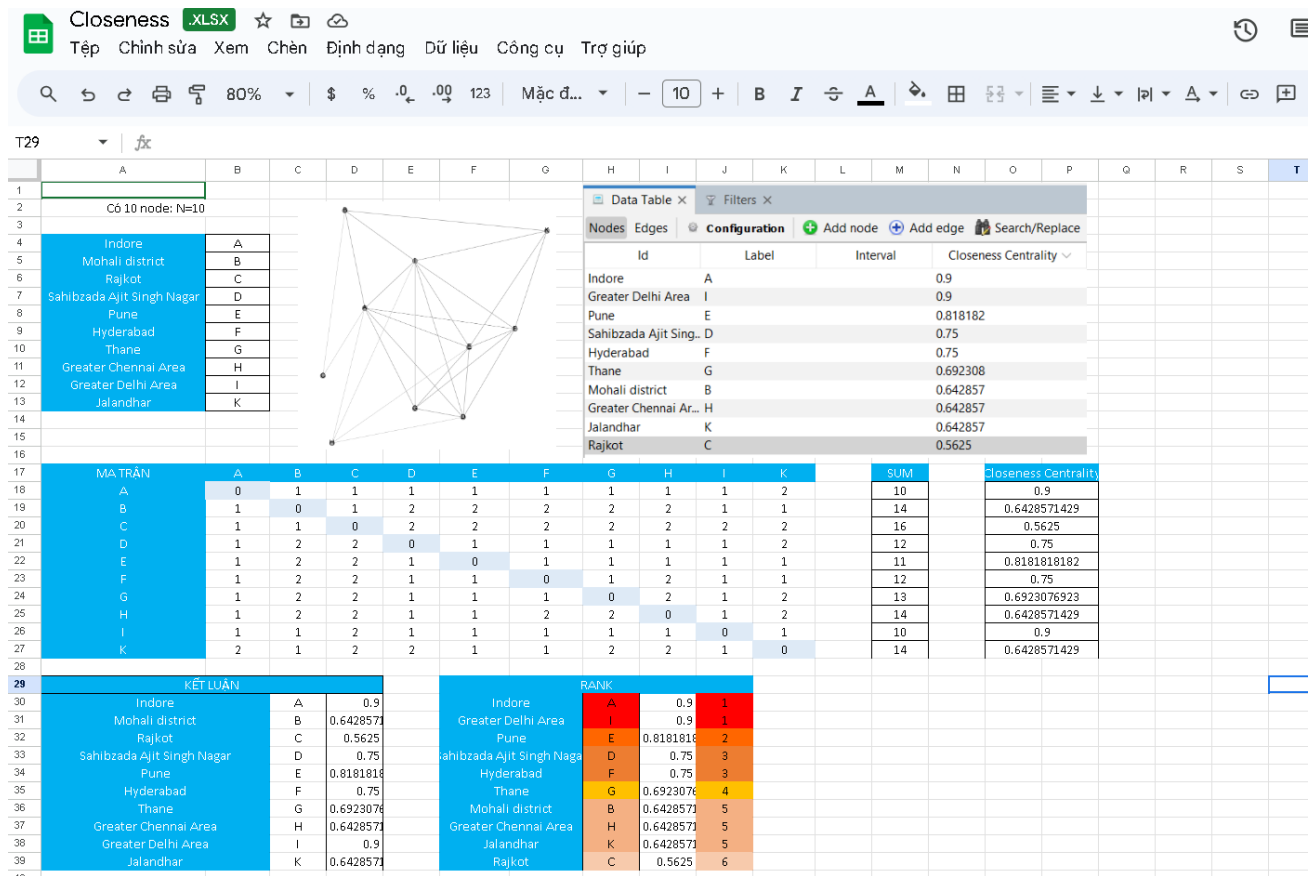
CHƯƠNG V: TRÍCH XUẤT 10 NODE VÀ THỰC HIỆN TÍNH TAY

5.1 Mô hình Gephi

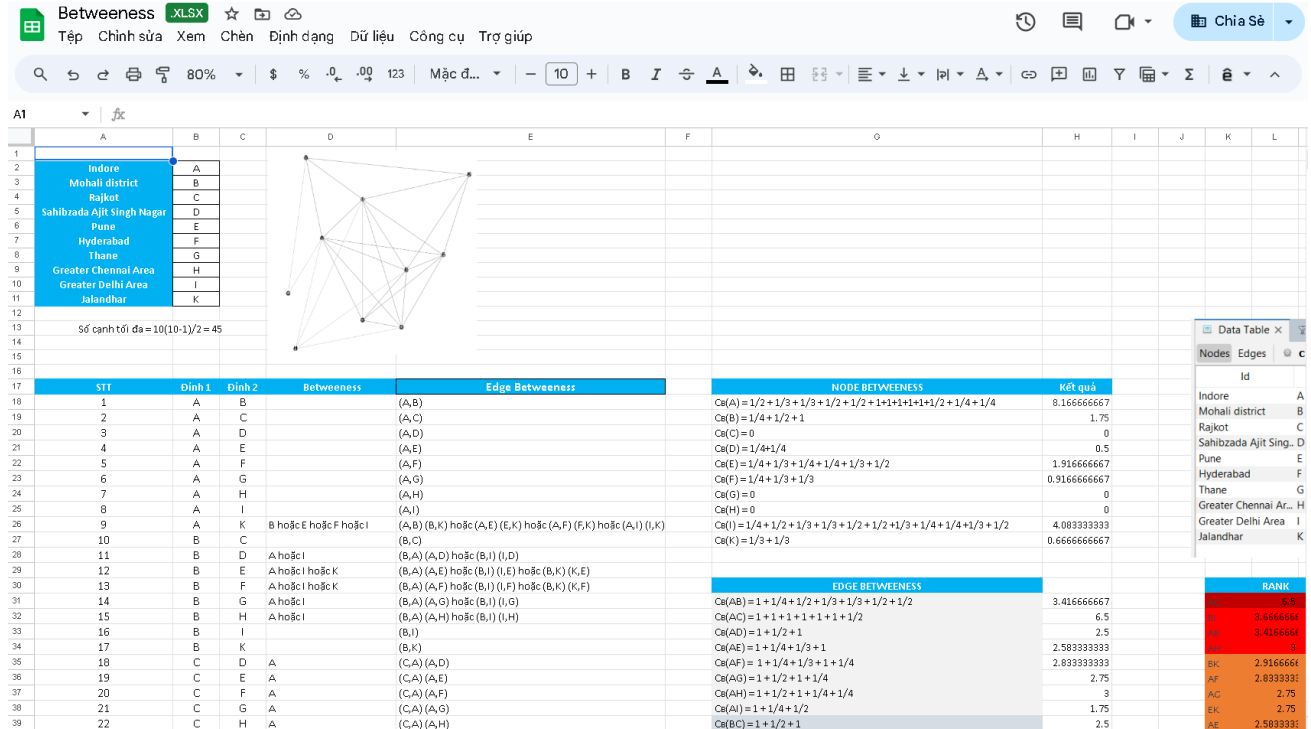


5.2 Các độ đo

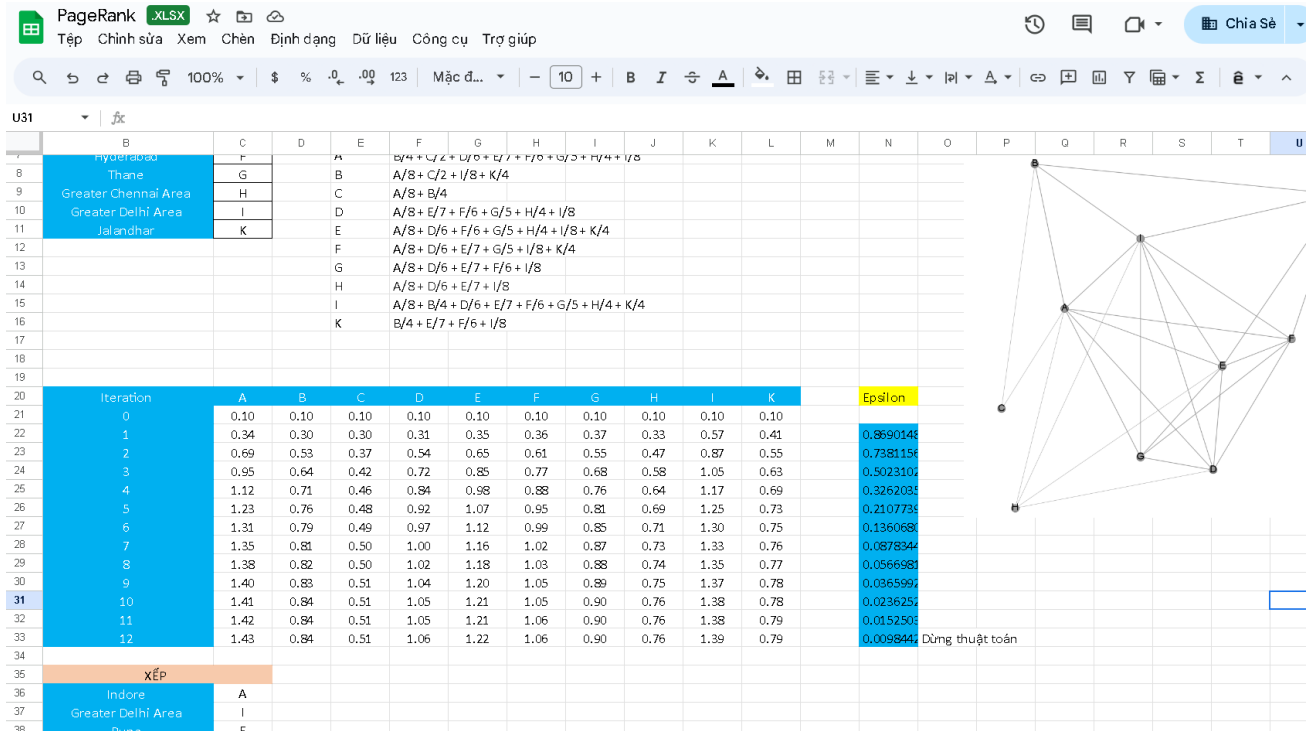
5.2.1 Closeness Centrality



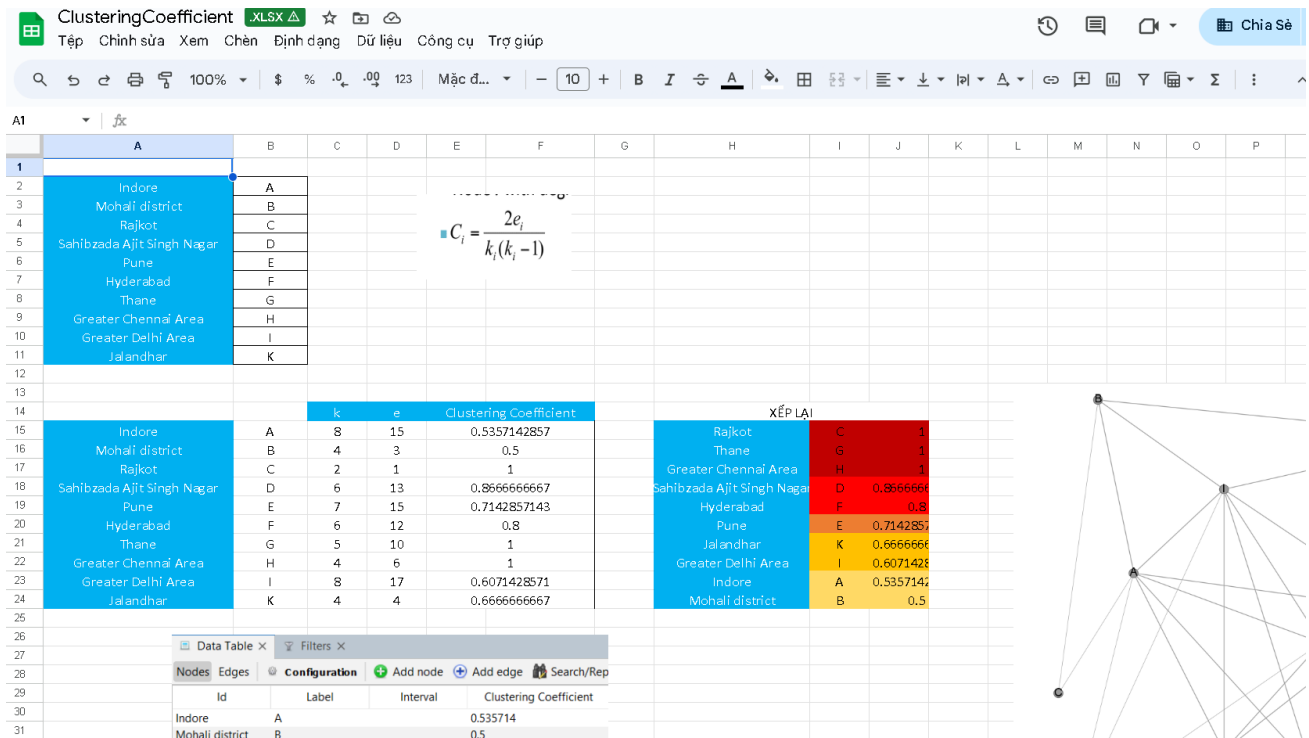
5.2.2 Betweenness Centrality



5.2.4 PageRank



5.2.5 Clustering Coefficient



5.2.6 Harmonic

Harmonic .XLSX ☆ Đã lưu vào Drive
 Tập Chỉnh sửa Xem Chèn Định dạng Dữ liệu Công cụ Trợ giúp

100% 123 Mặc đ... 10 + B I A

J20

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
14														
15														
16														
17		HARMONIC												
18	A	8.5												
19	B	6.5												
20	C	5.5												
21	D	7.5												
22	E	7.5												
23	F	7.5												
24	G	7												
25	H	6.5												
26	I	8.5												
27	K	6.5												
28														
29														
30	Xếp lại	Harmonic	Sau khi chia 9 thì giống với Gepi	RANK										
31	Indore	A	8.5	0.9444444444	1									
32	Greater Delhi Area	I	8.5	0.9444444444	1									
33	Pune	E	7.5	0.8333333333	2									
34	Sahibzada Ajit Singh Nagar	D	7.5	0.8333333333	2									
35	Hyderabad	F	7.5	0.8333333333	2									
36	Thane	G	7	0.7777777778	3									
37	Mohali district	B	6.5	0.7222222222	4									
38	Greater Chennai Area	H	6.5	0.7222222222	4									
39	Jalandhar	K	6.5	0.7222222222	4									
40	Rajkot	C	5.5	0.6111111111	5									
41														

Data Table x **Filters** x

Nodes **Edges** **Configuration** + Add node + Add edge Search/Replace

Id	Label	Interval	Harmonic Closeness Centrality
Indore	A		0.944444
Greater Delhi Area	I		0.944444
Pune	E		0.888889
Sahibzada Ajit Singh Nagar	D		0.833333
Hyderabad	F		0.833333
Thane	G		0.777778
Mohali district	B		0.722222
Greater Chennai Area	H		0.722222
Jalandhar	K		0.722222
Rajkot	C		0.611111

- Thực hiện trong file Manual Calculation đã nộp trên gg drive

TÀI LIỆU THAM KHẢO

- [1] Website môn học Mạng Xã Hội
- [2] "Page Rank Algorithm and Implementation," [Online]. Available: <https://www.geeksforgeeks.org/page-rank-algorithm-implementation/>.
- [3] "Girvan–Newman algorithm," [Online]. Available: https://en.wikipedia.org/wiki/Girvan%E2%80%93Newman_algorithm.
- [4] "PageRank," [Online]. Available: <https://vi.wikipedia.org/wiki/PageRank>.
- [5] "Betweenness centrality," [Online]. Available: https://en.wikipedia.org/wiki/Betweenness_centrality.
- [6] M. Telatnik, "How To Get Started with Social Network Analysis," 27 05 2020. [Online]. Available: <https://towardsdatascience.com/how-to-get-started-with-social-network-analysis-6d527685d374>.
- [7] "Is it possible to find closeness centrality using Gephi?," [Online]. Available: <https://stackoverflow.com/questions/28727120/is-it-possible-to-find-closeness-centrality-using-gephi>.
- [8] D. Liyan , L. Yongli , Y. Han , L. Huang and R. Mao , "The Algorithm of Link Prediction on Social Network," 17 09 2013. [Online]. Available: <https://www.hindawi.com/journals/mpe/2013/125123/>.