

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC

MẠNG XÃ HỘI

IS353.N22.HTCL

< DATASET BOOKING HOTEL >

Sinh viên thực hiện:

Lê Hoàng Huy 20521392

Huỳnh Thị Thanh Ngân 20521645

Giảng viên:

Nguyễn Thị Kim Phụng

Thành phố Hồ Chí Minh, tháng 5 năm 2023

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

Mục Lục

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI	2
1.Lý do chọn đề tài:	2
2.Nội dung đề tài	3
3. Xác định bài toán	3
4. Mô tả dữ liệu gốc	3
CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU VÀ TRỰC QUAN HÓA DỮ LIỆU	4
1. Tiền xử lý dữ liệu:	4
2. Tạo mạng liên kết:	5
3. Trực quan hóa mạng liên kết bằng code python	6
4. Trực quan hóa đồ thị 1 phía trên gephi	8
CHƯƠNG 3: PHÂN TÍCH VÀ TRỰC QUAN HÓA BẰNG CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG	11
1. Thuật toán phát hiện cộng đồng Louvain	11
2. Thuật toán Girvan Newman	19
CHƯƠNG 4: PHÂN TÍCH VÀ TRỰC QUAN HÓA BẰNG CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG	25
1. Đo độ Closeness centrality.	25
2.Đo độ Betweenness centrality.	28
3. Thuật toán PageRank	31
4. Eigenvector và Eigenvalue	33
4.1. Eigenvector centrality	33
4.2. Eigenvalue	35
TÀI LIỆU THAM KHẢO	36
[1] Link Code Python (Google Colab)	36
[2] CodePython_Sampleupdate.doc	36
[3] Link Dataset	36

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1. Lý do chọn đề tài:

Đề tài về dataset booking hotel là một lựa chọn tuyệt vời để chạy các thuật toán centrality. Lý do vì đây là một lĩnh vực đang phát triển mạnh mẽ trong ngành du lịch và khách sạn, với một lượng dữ liệu lớn và đa dạng. Dữ liệu booking hotel bao gồm thông tin về các lượt đặt phòng, vị trí khách sạn, giá cả, độ phổ biến của các dịch vụ, đánh giá của khách hàng về chất lượng dịch vụ và các tiện ích khác.

Việc chạy các thuật toán centrality sẽ giúp phân tích và tìm ra các yếu tố quan trọng nhất trong việc đặt phòng khách sạn. Ví dụ như thuật toán degree centrality có thể giúp xác định các khách sạn đang được quan tâm nhiều nhất hoặc thuật toán betweenness centrality giúp phân tích các địa điểm vận chuyển quan trọng nhất để tiện cho khách hàng đến khách sạn.

Kết quả phân tích của các thuật toán centrality giúp cho các nhà quản lý khách sạn có thông tin để tối ưu hóa chất lượng dịch vụ và quản lý thị trường, từ đó cải thiện năng suất kinh doanh và tăng lợi nhuận. Việc chạy các thuật toán centrality trên dataset booking hotel còn có thể giúp đưa ra các đề xuất về phân khúc khách hàng, nhằm tạo ra các gói dịch vụ phù hợp với các nhu cầu khác nhau của khách hàng. Chẳng hạn, các thuật toán clustering có thể phân cụm các loại khách hàng khác nhau và đưa ra các gợi ý dịch vụ tương ứng để thu hút khách hàng đến với khách sạn.

Ngoài ra, dataset booking hotel cũng cung cấp thông tin phong phú về các yếu tố liên quan đến khách sạn như địa điểm, giá cả, chất lượng dịch vụ, tiện ích, vị trí và lịch sử đặt phòng. Các thuật toán tìm kiếm tương tự (similarity searching) và thuật toán recommendation system có thể sử dụng các thông tin này để đưa ra các gợi ý khách sạn và gói dịch vụ phù hợp cho khách hàng.

Tóm lại, chọn đề tài về dataset booking hotel để chạy các thuật toán centrality là một lựa chọn hợp lý trong việc nghiên cứu và phân tích trong lĩnh vực du lịch và khách sạn. Việc áp dụng các thuật toán này sẽ giúp tối ưu hóa chất lượng dịch vụ, quản lý thị trường và tăng lợi nhuận, đồng thời cũng giúp tạo ra các gợi ý phù hợp cho khách hàng và tăng cường trải nghiệm của họ khi ở khách sạn.

2. Nội dung đề tài

- Tên dataset : Booking Hotel
- Link dataset : <https://www.kaggle.com/datasets/mojtaba142/hotel-booking>
- Mô tả : Dữ liệu ban đầu được lấy từ bài viết Bộ dữ liệu nhu cầu đặt phòng khách sạn, được viết bởi Nuno Antonio, Ana Almeida và Luis Nunes cho Tóm tắt dữ liệu, Tập 22, tháng 2 năm 2019.
- Bộ dữ liệu này chứa 119390 quan sát đối với Khách sạn Thành phố và Khách sạn Nghỉ dưỡng. Mỗi quan sát đại diện cho một đặt phòng khách sạn trong khoảng thời gian từ ngày 1 tháng 7 năm 2015 đến ngày 31 tháng 8 năm 2017, bao gồm cả đặt phòng đã đến và đặt phòng đã bị hủy.
- Công cụ hỗ trợ :
 - Code python : Google Colab
 - Thực hiện thuật toán và vẽ đồ thị : Gephi

3. Xác định bài toán

- Input : Tập dữ liệu ban đầu trên nguồn dữ liệu của kaggle được tải về để thực hiện đồ án.
- Output : Đưa ra độ đo, đưa ra cộng đồng phục vụ phân tích trực quan hóa bằng các thuật toán để giải quyết dataset : “ Booking Hotel”

4. Mô tả dữ liệu gốc

STT	Thuộc tính	Kiểu dữ liệu	Mô tả	Các giá trị
1	adr	int	Số tiền khách sạn nhận khi thực hiện phục vụ khách hàng đặt phòng	0,10,50,...
2	required_car_parking_spaces	int	Lượt gửi xe của khách hàng khi đến khách sạn	0,1,2
3	total_of_special_requests	int	Yêu cầu đặc biệt của khách hàng trong quá trình đặt phòng	0,1,2,3,4
4	reservation_status	varchar	Trạng thái khách hàng khi sử dụng phòng	canceled, check in , check out
5	reservation_status_date	date	Thời gian khách hàng bắt đầu sử dụng phòng	dd/mm/yyyy
6	name	varchar	Tên khách hàng	Ernest Barnes
7	email	varchar	Email khách hàng	Williams_Paul@xfinity.com
8	phone-number	char	Số điện thoại khách hàng	669-792-1661
9	credit_card	char	Mã số thẻ mà khách hàng dùng để thanh toán	*****7006

Hình 1. Mô tả dữ liệu

CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU VÀ TRỰC QUAN HÓA DỮ LIỆU

1. Tiền xử lý dữ liệu:

* Làm sạch dữ liệu : Bộ dữ liệu nhìn chung có vài thuộc tính bị thiếu dữ liệu, cần loại bỏ khỏi dataset.

```
import pandas as pd
df = pd.read_csv('/content/datahotel.csv', usecols=['adr', 'total_of_special_requests'])
df.dropna()
df.drop_duplicates()
df
```

Hình 2. Code python làm sạch và tách dữ liệu

* Kết quả sau khi tách 2 cột ra khỏi dataset

	adr	total_of_special_requests
0	0.00	0
1	0.00	0
2	75.00	0
3	75.00	0
4	98.00	1
...
994	179.38	0
995	166.00	2
996	202.00	2
997	172.00	2
998	277.00	1

Hình 3. Làm sạch và tách dữ liệu

2. Tạo mạng liên kết:

- Sử dụng đoạn code python để tính số nút và số cạnh của đồ thị từ dataset đã tách

```
import networkx as nx

#Tạo đồ thị từ DataFrame
G = nx.from_pandas_edgelist(df, 'adr', 'total_of_special_requests', create_using=nx.Graph())

#Tính số cạnh
num_edges = G.number_of_edges()

#Tính số nút
num_nodes = G.number_of_nodes()

#In ra kết quả
print(f"Số cạnh của đồ thị là: {num_edges}")
print(f"Số nút của đồ thị là: {num_nodes}")
```

Số cạnh của đồ thị là: 539
Số nút của đồ thị là: 421

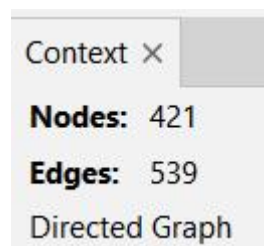
Hình 4. Đếm số cạnh và nút của đồ thị bằng code python

Có thể thấy sau khi tách và làm sạch dữ liệu để chọn 2 cột đối lập nhau thì số cạnh và số nút của dữ liệu đã giảm xuống và hoàn toàn khớp với dữ liệu bên Gephi

* Google colab:

- Số cạnh của đồ thị là: 539
- Số nút của đồ thị là : 421

* Gephi:



Hình 5. Nút và cạnh của đồ thị khi thêm dữ liệu vào gephi

3. Trực quan hóa mạng liên kết bằng code python

* Code đồ thị 2 phía trên Google colab :

```
import networkx as nx
from networkx.algorithms import bipartite
B = nx.Graph()
AD=df['adr']
CC=df['total_of_special_requests']
print("Doanh thu",AD.nunique())
print("Yêu cầu của khách hàng", CC.nunique())
print("Số cạnh", len(df))
```

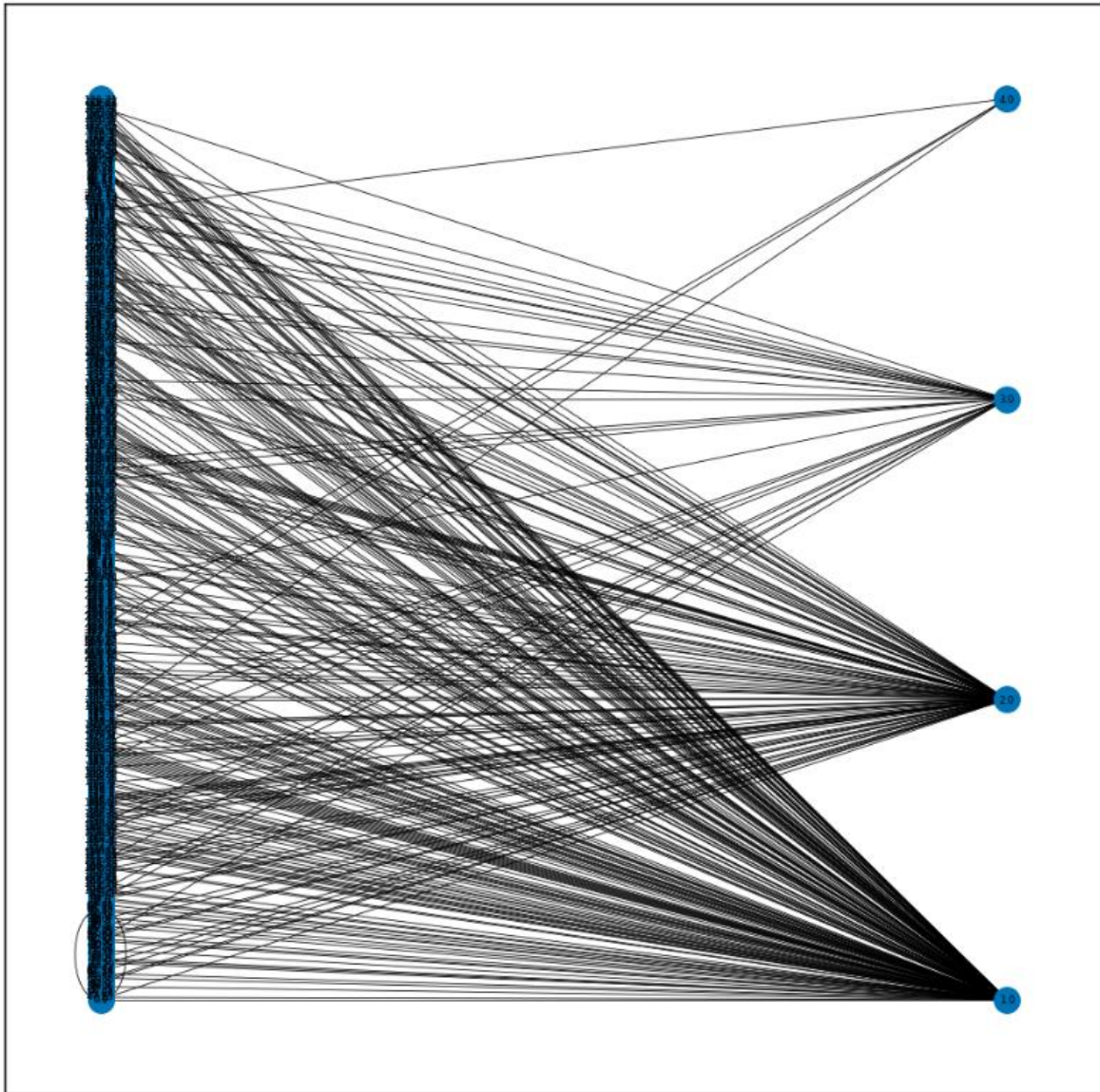
Doanh thu 410
Yêu cầu của khách hàng 5
Số cạnh 977

```
[4] from networkx.algorithms import bipartite
B = nx.Graph()
for index, row in df.iterrows():
    B.add_edge(row['adr'], row['total_of_special_requests'], weight=1)
B.add_nodes_from(AD, bipartite=0)
B.add_nodes_from(CC, bipartite=1)
```

```
[5] import matplotlib.pyplot as plt
plt.figure(figsize=(30,30))
pos = nx.spring_layout(B, scale = 40)
fig, ax = plt.subplots(1,1, figsize=(8,8), dpi = 150)
nx.draw_networkx(B, pos = nx.drawing.layout.bipartite_layout(B, AD), font_size=4, width=0.4, node_size=100)
```

Hình 6. Code python vẽ đồ thị 2 phía

* Kết quả đồ thị 2 phía :



Hình 7. Đồ thị 2 phía

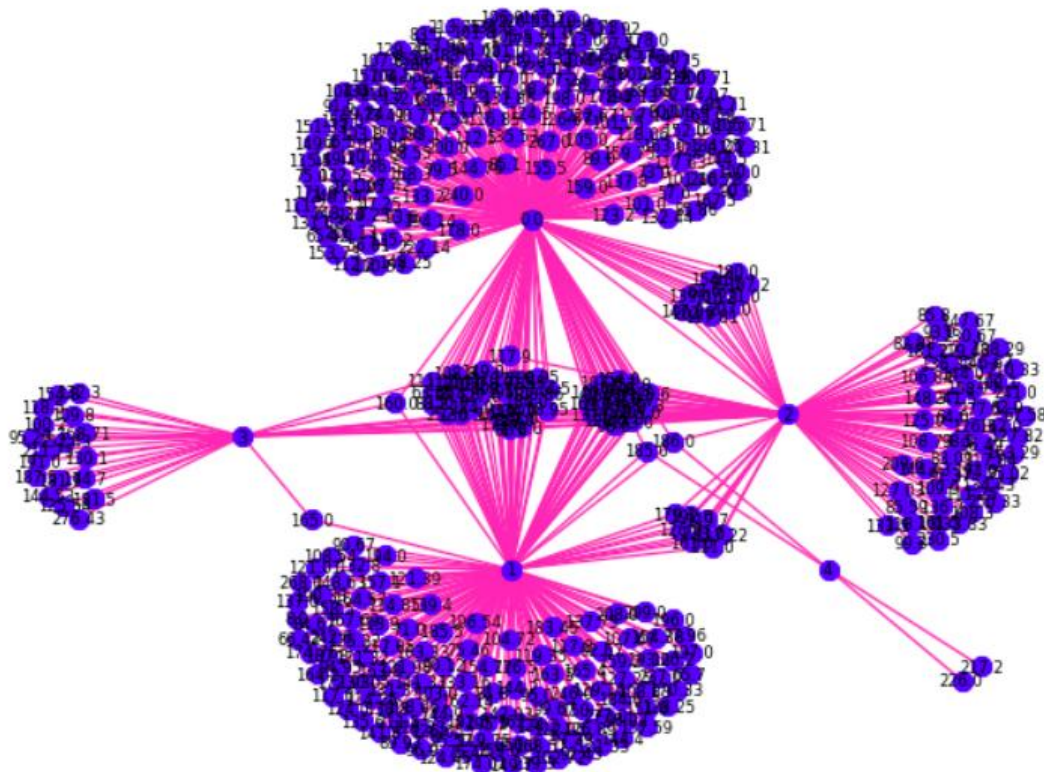
* **Nhận xét :** Nếu tổng số yêu cầu đặc biệt mà khách hàng yêu cầu trong quá trình đặt phòng ít, có thể cho thấy rằng các khách hàng của khách sạn không quá đòi hỏi và yêu cầu nhiều dịch vụ hơn so với những khách sạn khác. Điều này có thể làm tăng tính khả thi của khách sạn về việc đáp ứng các yêu cầu đặc biệt của khách hàng, đồng thời giảm thiểu thành công việc đặt phòng khách hàng bị hủy hoặc không xuất hiện.

4. Trực quan hóa đồ thị 1 phía trên gephi

- Node: Số doanh thu khách sạn nhận được từ khách hàng có yêu cầu đặc biệt trong quá trình đặt phòng.
- Edge: Hai nút doanh thu khách sạn nhận được từ khách hàng có yêu cầu đặc biệt trong quá trình đặt phòng được nối với nhau tạo thành cạnh, ý nghĩa nói lên sự thu nhập doanh thu của dịch vụ phát triển khi có chung số lượng yêu cầu đặc biệt.
- Weight: Trọng số là số yêu cầu đặc biệt của khách hàng trùng nhau.

```
[161] G = nx.from_pandas_edgelist(df, 'adr', 'total_of_special_requests', create_using=nx.Graph())  
y nx.draw(G, with_labels=True, node_color='blue', node_size=50, edge_color='hotpink', font_size=6, width=1.0)  
plt.show()
```

Hình 8. Đồ thị 1 phía thực thi bằng code python



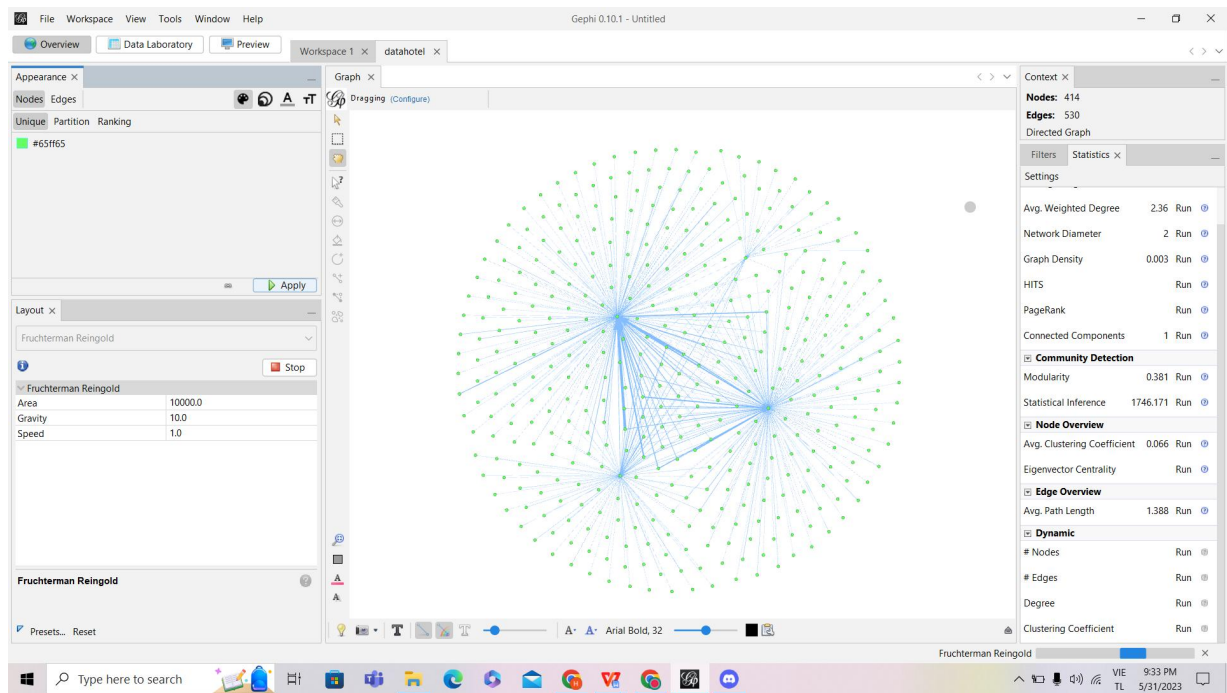
Hình 9. Kết quả đồ thị 1 phía thực thi bằng code python

- Xuất dữ liệu thành file csv rồi tải về để thực hiện sẽ đồ thị trên gephi

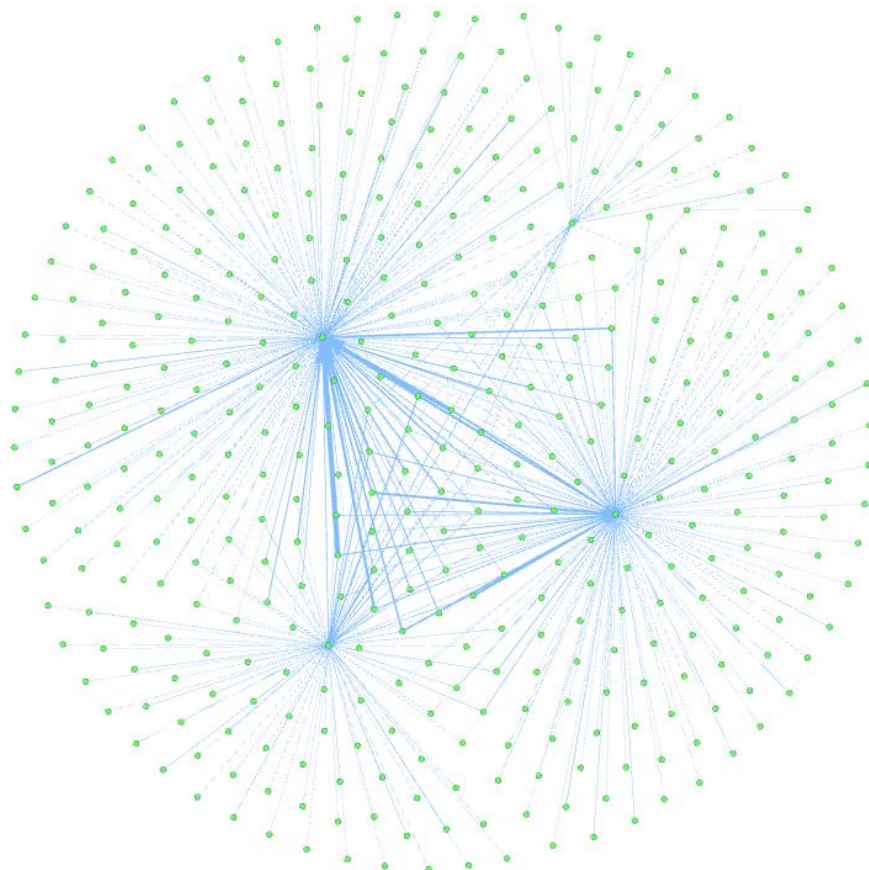
A1 source									
A	B	C	D	E	F	G	H	I	J
source	required_ca	target	reservation	reservation	name	email	phone-num	credit_card	
0	0	0	Check-Out	7/1/2015	Ernest Barn	Ernest.Barn	669-792-16	*****4322	
0	0	0	Check-Out	7/1/2015	Andrea Bak	Andrea_Bak	858-637-69	*****9157	
94.95	0	1	Check-Out	7/1/2015	Angie Sanch	Angie_Sanch	211-889-24	*****8871	
63.6	1	0	Check-Out	7/8/2015	Alexis King	King_Alexis	7103-516-58	*****6809	
79.5	0	0	Check-Out	7/8/2015	Michael Dav	MichaelDav	336-525-24	*****8662	
107	0	2	Canceled	5/11/2015	Jaime Flynn	JaimeFlynn2	549-866-37	*****9660	
94	0	0	Check-Out	7/8/2015	Mrs. Alicia V	Mrs..W61@	427-564-49	*****4445	
87.3	1	1	Check-Out	7/8/2015	Heather Hai	Heather.H@	431-329-66	*****2780	
62	0	2	Check-Out	7/15/2015	Diamond W	Wilson.Dian	870-563-62	*****8017	
63.86	0	0	Check-Out	7/16/2015	Paul William	Williams_Pa	789-736-88	*****7006	
108.3	0	2	Canceled	5/29/2015	Reginald Cu	Reginald_C	800-249-21	*****5699	
65.5	0	0	Check-Out	7/8/2015	Willie Sims	Willie_S@y	790-830-76	*****7682	
108.8	0	1	Canceled	5/19/2015	Alex Brown	Alex.B@zoh	956-737-19	*****4084	
108.8	0	1	Canceled	6/19/2015	Virginia Hol	Virginia.H@	734-480-99	*****8469	
98	0	0	Check-Out	7/6/2015	John Smith	John_Smith	624-965-33	*****6445	
108.8	0	1	Check-Out	7/6/2015	Jennifer Bar	JBarton@hc	902-792-80	*****2544	
108.8	0	0	Canceled	5/23/2015	Richard Hol	Richard.Hol	517-420-63	*****7544	
137	0	1	Check-Out	7/7/2015	Amy Green	Amy.Green	5858-208-45	*****3073	
117.81	0	0	Canceled	5/18/2015	Kimberly Pe	Perez.Kimbe	320-615-35	*****4653	
79.5	0	0	Check-Out	7/7/2015	Carly Armst	CArmstrong	521-784-36	*****7391	
123	0	0	Check-Out	7/7/2015	Scott Hamr	ScottHamm	632-108-83	*****4823	
137	0	1	Check-Out	7/7/2015	Debra Fuller	Debra.Fuller	229-134-37	*****3550	
110.7	0	2	Check-Out	7/9/2015	Rachel Davi	RachelDavis	549-261-25	*****2810	
153	0	0	Canceled	6/2/2015	Michael Mc	Mccormick.	380-830-95	*****2950	
58.05	0	1	Check-Out	7/8/2015	Ann Sullivan	AnnSullivan	847-577-27	*****4792	

Hình 10. File csv tải về

- Đồ thị 1 phía trên gephi



Hình 11. Đồ thị 1 phía trên gephi

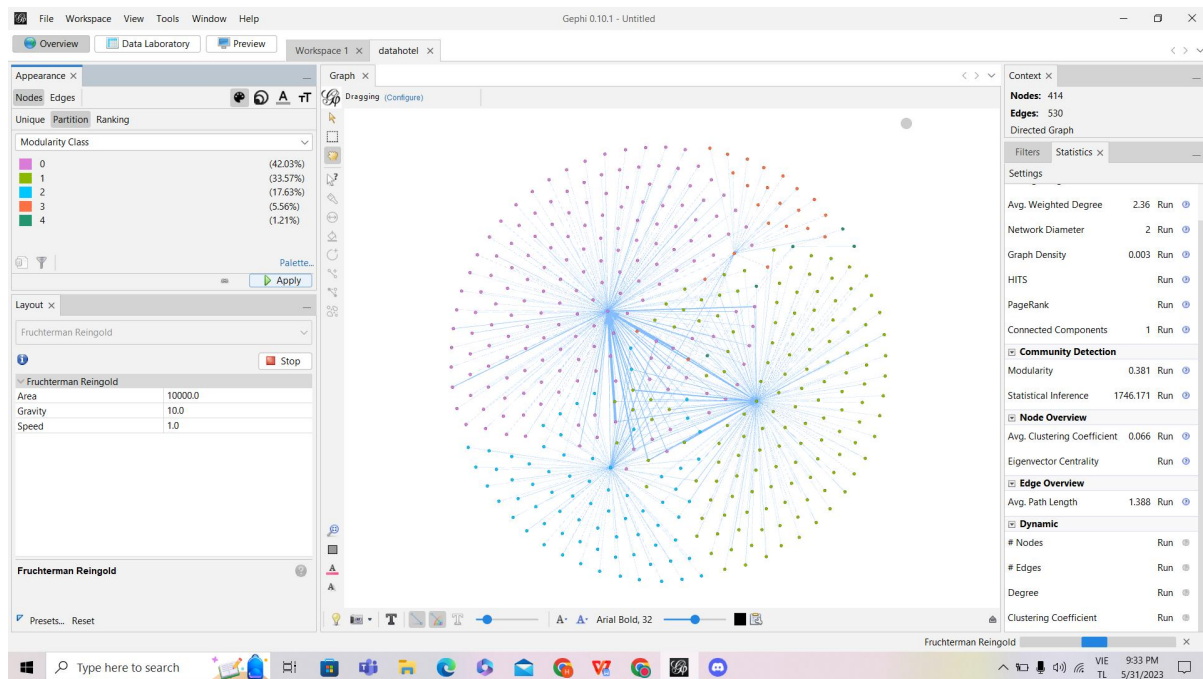


Hình 12. Đồ thị 1 phía trên gephi full

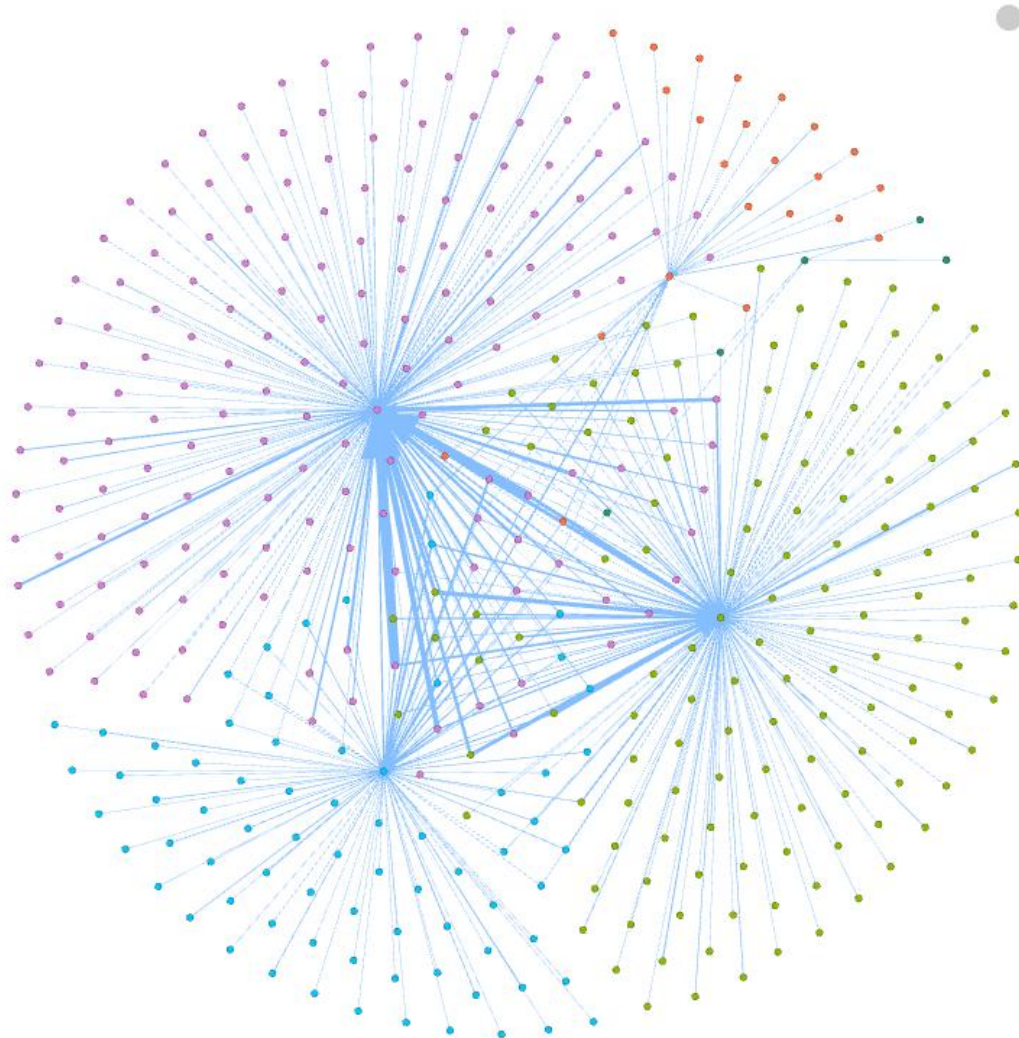
CHƯƠNG 3: PHÂN TÍCH VÀ TRỰC QUAN HÓA BẰNG CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG

1. Thuật toán phát hiện cộng đồng Louvain

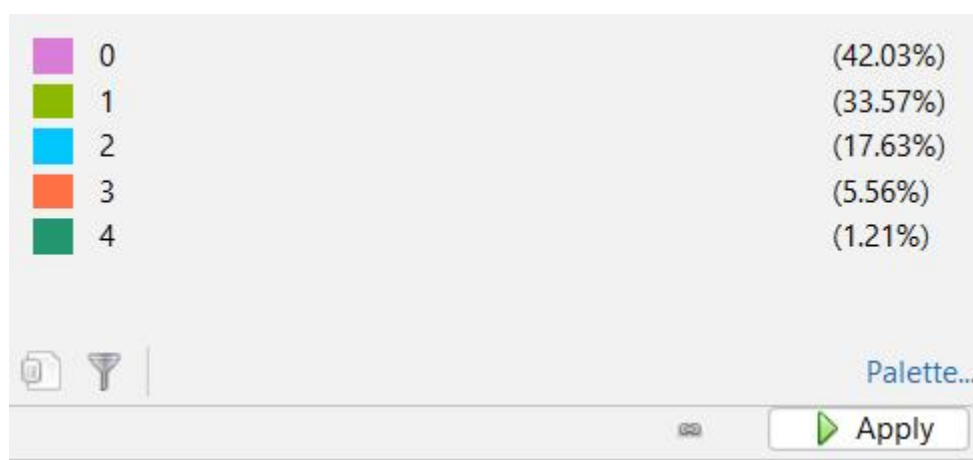
* Thực thi thuật toán Louvain bằng Gephi



Hình 13. Sử dụng thuật toán Louvain



Hình 14. Sử dụng thuật toán Louvain full



Hình 15. Thuật toán chia thành 5 cụm màu

Nhận xét: Thuật toán chia ra thành 5 cụm

* Thực thi bằng Python

```
!pip install python-igraph
!pip install numpy
!pip install pandas
!pip install networkx
!pip install python-louvain
```

Hình 16. Khai báo , cài đặt thư viện code python

```
import networkx as nx
from community import community_louvain
import matplotlib.pyplot as plt

# Tạo đồ thị
G = nx.from_pandas_edgelist(df, 'adr', 'total_of_special_requests', create_using=nx.Graph())

# Chạy thuật toán Louvain
partition = community_louvain.best_partition(G)

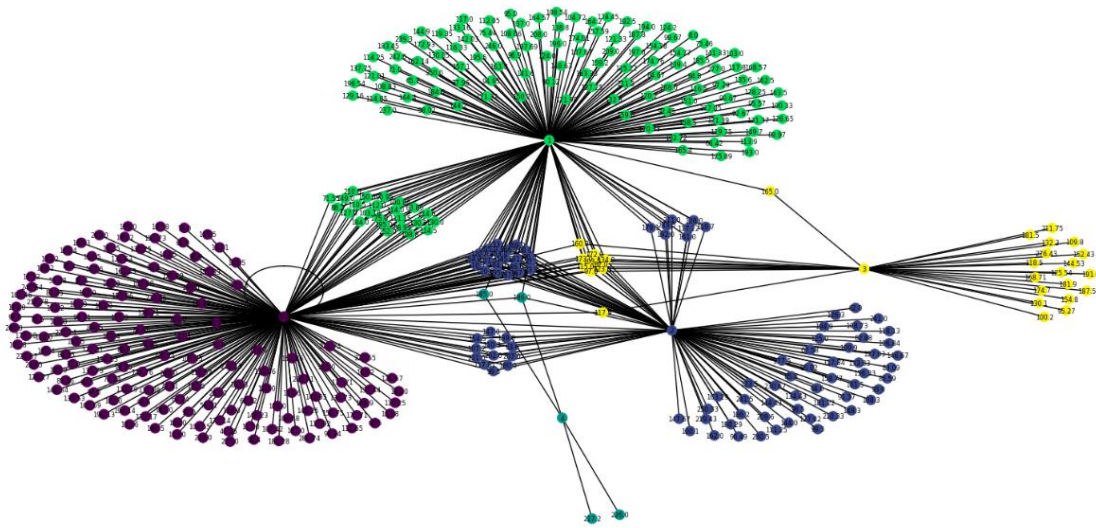
# Đếm số lượng màu sắc
num_colors = len(set(partition.values()))

# Vẽ đồ thị với màu sắc tương ứng với cộng đồng
pos = nx.spring_layout(G)
plt.figure(figsize=(17, 8))
colors = [partition[node] for node in G.nodes()]
nx.draw(G, pos=pos, node_color=colors, cmap='viridis', node_size=80, with_labels=True, font_size=6, width=1.0)

# Hiển thị số lượng màu
plt.text(0.5, -0.1, f"Số lượng màu: {num_colors}", transform=plt.gca().transAxes, ha="center")

# Hiển thị đồ thị
plt.show()
```

Hình 17. Code thực thi thuật toán louvain bằng python



Hình 18. Đồ thị thuật toán louvain bằng python

```

import numpy as np
values = list(partition.values())

#kt số lượng cộng đồng
print('số lượng cụm: ', len(np.unique(values)))

```

số lượng cụm: 5

Hình 19. Số lượng cụm của thuật toán Louvain

```

for i in range(len(np.unique(values))):
    print("cụm:",i)
    for name, k in partition.items():
        if k == i:
            print(name, end=" ")
    print("")

```

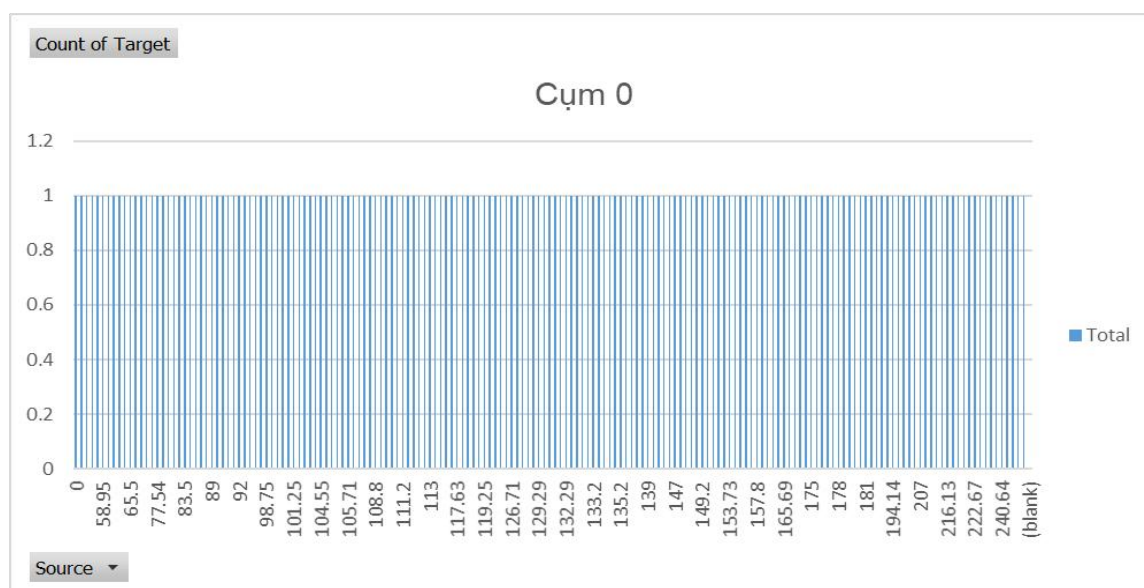
cụm: 0
0.0 79.5 94.0 65.5 82.35 56.01 91.5 90.9 122.0 85.86 55.68 134.73 94.71 163.0 67.24 105.0 113.0 85.1 89.0 80.1 101.0 225.0 105.5 98.4 74.07 134.
cụm: 1
123.0 117.9 82.0 172.0 109.8 3 130.1 165.0 125.54 95.27 100.2 134.0 187.5 103.5 160.0 132.3 118.5 173.0 115.0 152.43 154.8 144.53 174.7 168.71 1
cụm: 2
94.95 1 63.86 108.8 137.0 58.95 119.35 88.2 110.5 114.5 124.0 111.15 92.45 117.0 196.54 90.95 92.67 71.55 116.5 149.0 164.2 97.29 114.0 175.0 12
cụm: 3
185.0 186.0 4 217.2 226.0
cụm: 4
63.6 107.0 2 87.3 62.0 108.3 98.0 117.81 110.7 153.0 82.88 67.58 147.0 135.0 133.0 136.33 97.0 110.3 73.8 108.73 131.0 99.3 96.49 85.8 96.3 139.

Hình 20. Code đếm số lượng cụm

Số cum : 5

*Cụm 0: [0.0, 79.5, 94.0, 65.5, 82.35, 56.01, 91.5, 90.9, 122.0, 85.86, 55.68, 134.73, 94.71, 163.0, 67.24, 105.0, 113.0, 85.1, 89.0, 80.1, 101.0, 225.0, 105.5, 98.4, 74.07, 134.1, 117.63, 47.25, 73.0, 93.6, 83.5, 61.0, 89.68, 155.0, 105.08, 65.42, 119.25, 149.4, 152.0, 159.75, 112.2, 110.6, 105.9, 153.96, 126.67, 100.0, 104.68, 213.75, 178.0, 129.29, 132.29, 105.71, 78.84, 90.71, 128.0, 103.8, 175.71, 110.0, 134.25, 101.8, 73.41, 132.44, 106.9, 131.86, 216.13, 98.75, 148.23, 123.2, 198.0, 151.33, 144.5, 155.5, 107.6, 133.75, 92.0, 128.27, 136.0, 168.3, 132.6, 132.5, 249.0, 186.5, 177.0, 153.73, 124.5, 110.53, 117.71, 116.85, 135.2, 111.65, 104.0, 133.17, 240.64, 167.5, 148.34, 111.2, 200.0, 151.86, 178.4, 101.46, 112.5, 222.67, 161.25, 118.0, 240.0, 176.0, 131.63, 233.05, 88.55, 133.2, 167.2, 115.5, 188.0, 141.65, 99.24, 280.74, 137.8, 126.71, 219.5, 111.92, 57.6, 157.8, 118.06, 177.14, 222.14, 194.14, 193.13, 210.78, 135.53, 180.28, 159.0, 104.55, 144.76, 220.55, 77.54, 171.9, 148.25, 183.0, 149.2, 267.0, 181.0, 200.71, 228.0, 138.0, 165.69, 127.31, 145.4, 101.25, 179.38]

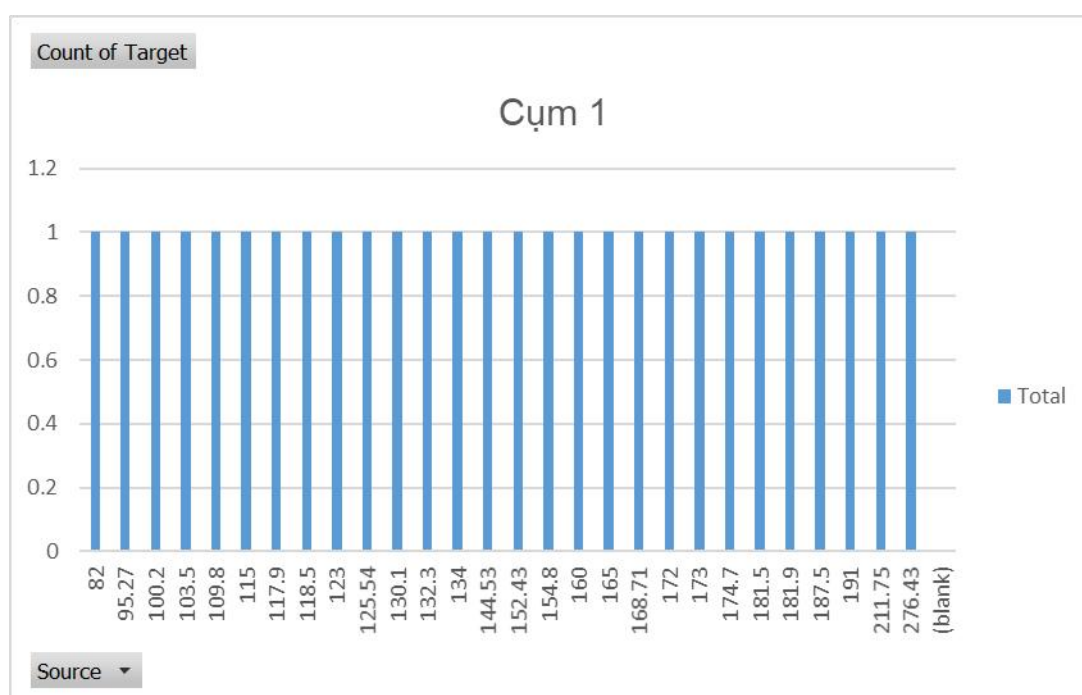
Biểu đồ cụm 0 có ý nghĩa : Cụm 0 chứa các giá trị giá cả thấp đến trung bình. Đa dạng về giá trị và không có sự tương đồng. Trong cụm này có giá trị cao nhất là 280.74 và thấp nhất là 47.25.



Hình 21. Cụm 0

*Cụm 1: [123.0, 117.9, 82.0, 172.0, 109.8, 3, 130.1, 165.0, 125.54, 95.27, 100.2, 134.0, 187.5, 103.5, 160.0, 132.3, 118.5, 173.0, 115.0, 152.43, 154.8, 144.53, 174.7, 168.71, 181.5, 181.9, 191.0, 211.75, 276.43]

Biểu đồ cụm 1 có ý nghĩa : Cụm 1 chứa các giá trị giá cả trung bình đến cao. Có các giá trị tập trung trong khoảng từ khoảng 100 đến 200. Các giá trị cao nhất trong cụm là 276.43 và thấp nhất là 82.

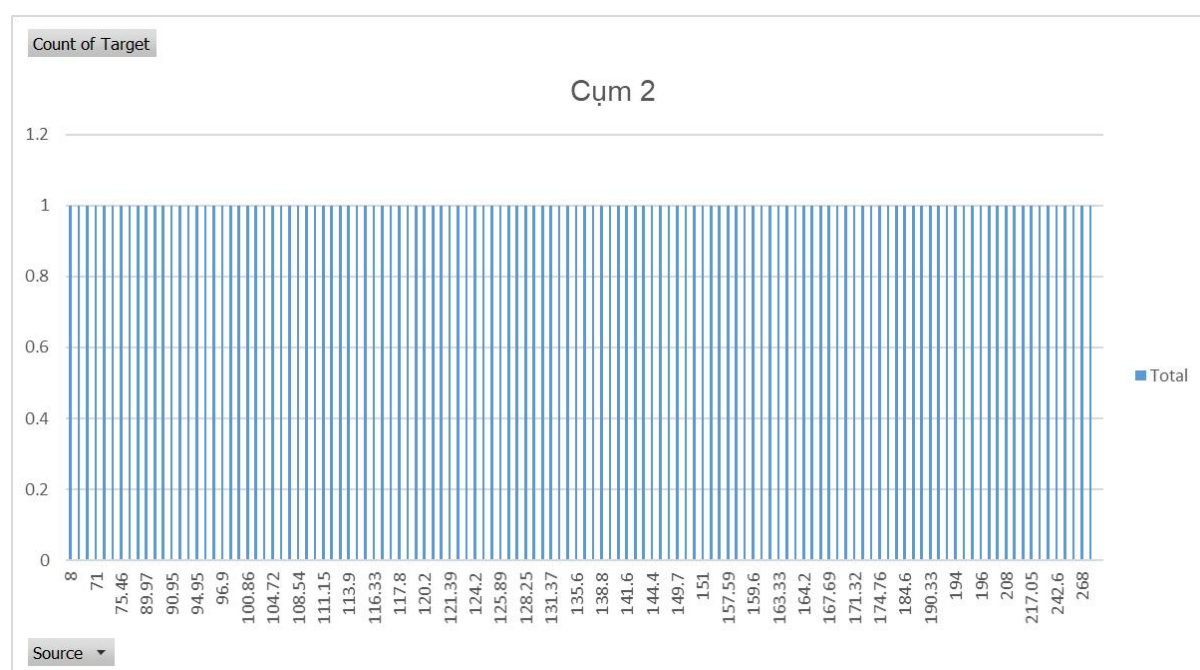


Hình 22. Cụm 1

*Cụm 2: [94.95, 1, 63.86, 108.8, 137.0, 58.95, 119.35, 88.2, 110.5, 114.5, 124.0, 111.15, 92.45, 117.0, 196.54, 90.95, 92.67, 71.55, 116.5, 149.0, 164.2, 97.29, 114.0, 175.0, 124.45, 171.32, 210.0, 117.8, 71.0, 151.0, 157.1, 185.5, 195.0, 193.0, 150.0, 126.65, 104.72, 66.42, 77.96, 112.0, 75.46, 197.0, 108.83, 90.67, 144.4, 139.4, 89.97, 144.9, 113.9, 164.57, 125.22, 130.0, 133.16, 120.2, 130.05, 196.0, 100.86, 132.8, 144.2, 163.8, 130.5, 184.6, 8.0, 127.0, 65.7, 168.57, 107.67, 96.9, 190.33, 135.6, 184.0, 121.33, 150.2, 75.44, 127.25, 84.8, 116.33, 137.75, 98.02, 95.0, 128.25, 158.5, 131.37, 129.16, 214.0, 141.33, 121.01, 172.93, 192.5, 141.6, 163.33, 95.57,

112.05, 103.18, 119.75, 167.69, 194.0, 217.05, 149.7, 159.2, 121.98, 142.03, 90.12, 114.25, 209.0, 174.01, 124.2, 183.45, 159.6, 102.72, 114.85, 138.8, 108.54, 165.4, 108.06, 148.63, 125.89, 170.33, 252.0, 237.0, 162.5, 195.5, 163.5, 174.76, 162.14, 187.8, 242.6, 268.0, 239.3, 154.38, 121.39, 250.0, 208.0, 157.59, 246.0, 211.5, 277.0]

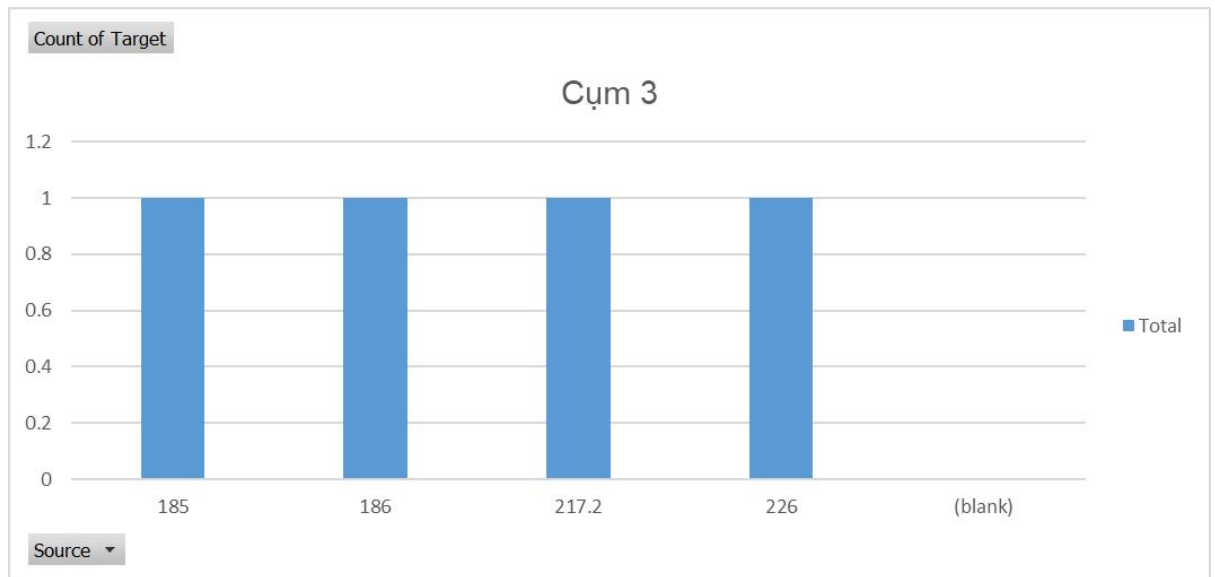
Biểu đồ cụm 2 có ý nghĩa : Cụm 2 chứa các giá trị giá cả thấp đến trung bình, nhưng cũng có một số giá trị cao. Bao gồm rất nhiều giá trị, bao phủ một khoảng giá trị rộng từ khoảng 60 đến 280. Các giá trị cao nhất trong cụm là 277 và thấp nhất là 1. Tuy nhiên, số lượng giá trị ở phần lớn khoảng giá trị nhỏ.



Hình 23. Cụm 2

*Cụm 3: [185.0, 186.0, 4, 217.2, 226.0]

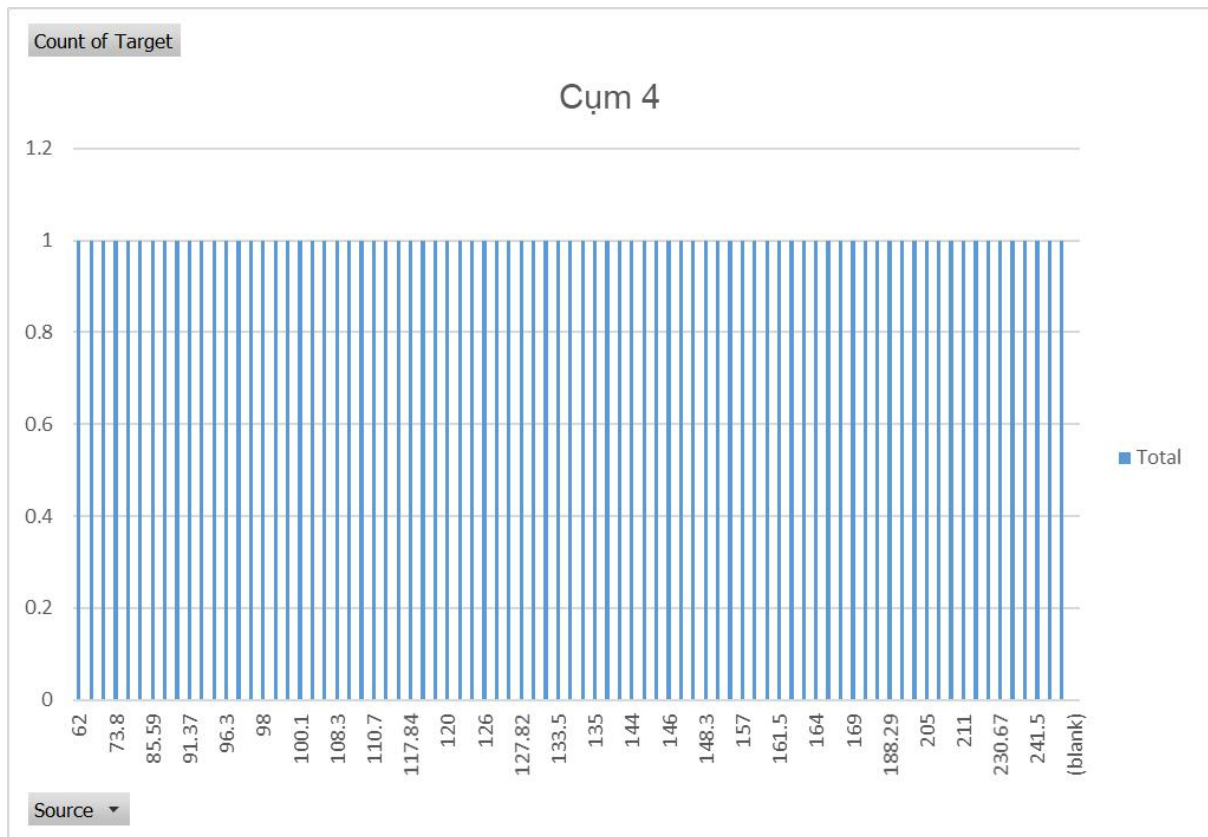
Biểu đồ cụm 3 có ý nghĩa : Cụm 3 chỉ có một số giá trị và giá cả của chúng cao hơn rất nhiều so với các cụm khác. Bao gồm các giá trị cao trong khoảng từ 185 đến 226, có chỉ một giá trị nằm trong khoảng còn lại. Điểm chung giữa các giá trị trong cụm này là chúng đều nằm trong khoảng 185-226.



Hình 24. Cụm 3

*Cụm 4: [63.6, 107.0, 2, 87.3, 62.0, 108.3, 98.0, 117.81, 110.7, 153.0, 82.88, 67.58, 147.0, 135.0, 133.0, 136.33, 97.0, 110.3, 73.8, 108.73, 131.0, 99.3, 96.49, 85.8, 96.3, 139.0, 167.0, 85.59, 120.6, 91.37, 146.0, 100.1, 118.13, 120.0, 106.84, 161.0, 134.43, 144.43, 133.83, 99.5, 166.0, 111.25, 169.0, 93.0, 117.22, 126.0, 148.3, 119.7, 64.0, 107.2, 98.5, 181.22, 83.09, 136.2, 161.5, 230.67, 180.0, 109.9, 164.0, 125.0, 145.0, 194.9, 192.0, 126.3, 154.0, 147.67, 157.0, 241.5, 127.03, 154.5, 117.84, 179.1, 207.0, 233.0, 163.29, 158.77, 95.02, 162.0, 219.43, 250.33, 146.67, 202.0, 211.0, 188.29, 127.82, 144.0, 209.6, 205.0, 221.0, 230.5, 241.0, 277.5, 133.5, 210.33]

Biểu đồ cụm 4 có ý nghĩa : Cụm 4 chứa các giá trị giá cả thấp đến trung bình. Tuy nhiên, có một số giá trị khá thấp. bao gồm các giá trị thấp phân bố khá đều. Các giá trị cao nhất trong cụm là 277.5 và thấp nhất là 2. Cụm này cũng có thể đại diện cho các kết quả hoặc tình huống trung bình. Tuy nhiên, sự phân bố đều hơn so với cụm 1.



Hình 25. Cụm 4

2. Thuật toán Girvan Newman

* Code python

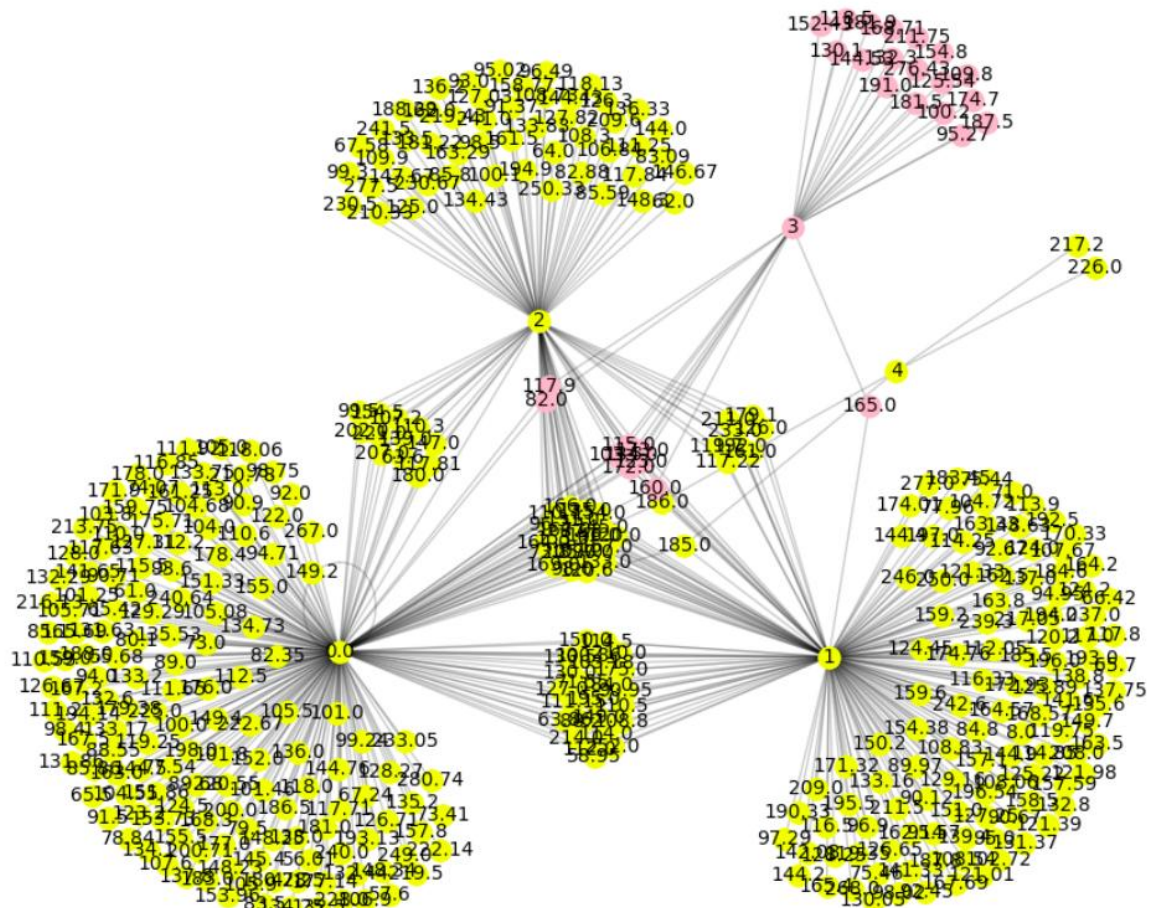
```

22 ✓
giấy ▶ from networkx.algorithms.community centrality import girvan_newman
comp = girvan_newman (G)
K=3
for i in range(k-1):
    comms = next(comp)

colors = ['yellow', 'pink']
plt.figure(figsize=(12, 10))
pos = nx.spring_layout(G)
for nodes, c in zip(comms, colors):
    nx.draw_networkx_nodes (G, pos, nodelist=nodes, node_color=[c], node_size=120)
nx.draw_networkx_edges (G, pos, alpha=0.2)
nx.draw_networkx_labels (G, pos, font_size=10)
plt.show()

```

Hình 26. Code python về thuật toán Girvan Newman



Hình 27. Đồ thị thể hiện thuật toán Girvan Newman

* Thực thi trên Gephi

Id	
0	0
100	0
100.86	0
101	0
101.25	0
101.46	0
101.8	0
103.18	0
103.8	0
104	0
104.55	0
104.68	0
105	0
105.08	0
105.5	0
105.71	0
105.9	0
106.9	0
107.2	0
107.42	0
107.6	0
108.8	0
110	0
110.3	0
110.5	0
110.53	0
110.6	0
111.15	0
111.2	0
111.65	0

Hình 28. Cụm 0 trên gephi

Id	
115	2
117.9	2
118.5	2
123	2
125.54	2
130.1	2
132.3	2
134	2
144.53	2
152.43	2
154.8	2
160	2
165	2
168.71	2
172	2
173	2
174.7	2
181.5	2
181.9	2
187.5	2
191	2
211.75	2
276.43	2
3	2
82	2
95.27	2
97	2

Hình 29. Cụm 1 trên gephi

Id	
209	3
211.5	3
217.05	3
237	3
239.3	3
242.6	3
246	3
250	3
268	3
277	3
65.7	3
66.42	3
71	3
75.44	3
75.46	3
77.96	3
8	3
84.67	3
84.8	3
89.97	3
90.12	3
90.67	3
92.45	3
92.67	3
94.95	3
95	3
95.57	3
96.9	3
97.29	3
98.02	3

Hình 30. Cụm 2 trên gephi

Id	Cluter-ID ^
185	1
186	1
217.2	1
226	1
4	1

Hình 31. Cụm 3 trên gephi

Id	C
100.1	4
106.84	4
107	4
108.3	4
108.73	4
109.9	4
110.7	4
111.25	4
117.22	4
117.84	4
118.13	4
119.7	4
120	4
120.6	4
125	4
126	4
126.3	4
127.03	4
127.82	4
131	4
133	4
133.5	4
133.83	4
134.43	4
135	4
136.2	4
136.33	4
144	4
144.43	4
145	4

Hình 32. Cụm 4 trên gephi

CHƯƠNG 4: PHÂN TÍCH VÀ TRỰC QUAN HÓA BẰNG CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG

1. Đo độ Closeness centrality.

* Thực thi bằng code python

```
import pandas as pd
import networkx as nx

# Chuyển dataframe sang đồ thị
G = nx.from_pandas_edgelist(df, source='adr', target='total_of_special_requests', edge_attr=True)

# Tính closeness centrality của tất cả các đỉnh
closeness centrality = nx.closeness centrality(G, distance='weight')



# Vẽ đồ thị với closeness centrality
plt.figure(figsize=(10,10))
pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_size=list(closeness centrality.values()), node_color='b', alpha=0.5)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif')
plt.axis('off')
plt.show()
```

Hình 33. code python closeness centrality

```
0.0: 0.658307210031348
1: 0.5761316872427984
82.0: 0.4994054696789536
123.0: 0.4994054696789536
97.0: 0.4994054696789536
172.0: 0.4994054696789536
134.0: 0.4994054696789536
103.5: 0.4994054696789536
173.0: 0.4994054696789536
115.0: 0.4994054696789536
186.0: 0.48109965635738833
2: 0.4805491990846682
98.0: 0.4778156996587031
107.0: 0.4778156996587031
145.0: 0.4778156996587031
153.0: 0.4778156996587031
87.3: 0.4778156996587031
110.7: 0.4778156996587031
135.0: 0.4778156996587031
133.0: 0.4778156996587031
73.8: 0.4778156996587031
131.0: 0.4778156996587031
96.3: 0.4778156996587031
167.0: 0.4778156996587031
120.6: 0.4778156996587031
146.0: 0.4778156996587031
120.0: 0.4778156996587031
166.0: 0.4778156996587031
169.0: 0.4778156996587031
```

Hình 34. Kết quả code python closeness centrality

* Thực thi bằng gephi

Nodes Edges  Configuration 			
Id	Label	Closeness C...	li
0		0.658692	
1		0.573611	
103.5		0.499395	
115		0.499395	
123		0.499395	
134		0.499395	
172		0.499395	
173		0.499395	
2		0.482477	
186		0.480792	
107		0.477457	
110.7		0.477457	
120		0.477457	
120.6		0.477457	
131		0.477457	
133		0.477457	
135		0.477457	
145		0.477457	
146		0.477457	
153		0.477457	
154		0.477457	
157		0.477457	
164		0.477457	
166		0.477457	
167		0.477457	
169		0.477457	
205		0.477457	
73.8		0.477457	
87.3		0.477457	
96.3		0.477457	

Hình 35. Closeness centrality trên gephi

2. Đo độ Betweenness centrality.

```
# Tạo đồ thị
G = nx.Graph()
for index, row in df.iterrows():
    G.add_node(row['adr'])
    G.add_node(row['total_of_special_requests'])
    G.add_edge(row['adr'], row['total_of_special_requests'])
# Chạy thuật toán betweenness centrality trên đồ thị G
betweenness centrality = nx.betweenness centrality(G, normalized=False)

# Sắp xếp betweenness centrality theo thứ tự giảm dần
betweenness_dict = dict(sorted(betweenness centrality.items(), key=lambda item: item[1], reverse=True))

# In kết quả betweenness centrality của từng đỉnh
for node, betweenness in betweenness_dict.items():
    print(f"{node}: {betweenness}")
```

Hình 36. Code python betweenness centrality

```

↳ 0.0: 56873.11990382341
   1: 44419.61990382357
   2: 21673.92685901946
   3: 7417.083333333334
  82.0: 915.6021502854051
 123.0: 915.6021502854051
  97.0: 915.6021502854051
 172.0: 915.6021502854051
 134.0: 915.6021502854051
 103.5: 915.6021502854051
 173.0: 915.6021502854051
 115.0: 915.6021502854051
 186.0: 883.5180272108848
    4: 837.3333333333336
 160.0: 612.0806474313852
 185.0: 521.5653061224489
 117.9: 487.72849323637394
 165.0: 270.79032371569235
  98.0: 157.08333333333331
 107.0: 157.08333333333331
 145.0: 157.08333333333331
 153.0: 157.08333333333331
  87.3: 157.08333333333331
 110.7: 157.08333333333331
 135.0: 157.08333333333331
 133.0: 157.08333333333331
  73.8: 157.08333333333331
 131.0: 157.08333333333331
  96.3: 157.08333333333331
 167.0: 157.08333333333331
 120.6: 157.08333333333331
 146.0: 157.08333333333331
 120.0: 157.08333333333331

```

Hình 37. Kết quả code python betweenness centrality

* Thực thi betweenness centrality trên gephi

Id	Label	Betweenness ...
0		55134.537056
1		42349.056886
2		21328.207927
3		7288.626812
115		1024.254601
123		1024.254601
172		1024.254601
134		1024.254601
103.5		1024.254601
173		1024.254601
186		872.090585
4		823.333333
160		702.113245
117.9		542.482125
82		542.482125
185		510.671429
165		326.010462
87.3		155.762014
120		155.762014
166		155.762014
164		155.762014
154		155.762014
161		155.762014
117.22		155.762014
126		155.762014
119.7		155.762014
192		155.762014
179.1		155.762014
233		155.762014
135		155.762014

Hình 38. Betweenness centrality trên gephi

3. Thuật toán PageRank

* Thực thi trên python

```
import networkx as nx
import pandas as pd

# Tính toán PageRank
pagerank = nx.pagerank(G)

# Sắp xếp theo giá trị PageRank giảm dần
sorted_pagerank = sorted(pagerank.items(), key=lambda x: x[1], reverse=True)

# Hiển thị bảng xếp hạng
print("Bảng xếp hạng PageRank:")
print("-----")
print("Player\tPageRank")
print("-----")
for player, rank in sorted_pagerank:
    print(f"{player}\t{rank}")
```

Hình 39. Code python pagerank

➞ Bảng xếp hạng PageRank:

Player	PageRank
0.0	0.19445046223487303
1	0.15622837203885223
2	0.08337970212607691
3	0.024472540568422036
4	0.0037758174114166615
186.0	0.003318727070522933
82.0	0.003233669857468426
123.0	0.003233669857468426
97.0	0.003233669857468426
172.0	0.003233669857468426
134.0	0.003233669857468426
103.5	0.003233669857468426
173.0	0.003233669857468426
115.0	0.003233669857468426
185.0	0.0026303534925453754
160.0	0.0025452962794908682
98.0	0.0025160618281356915
107.0	0.0025160618281356915

Hình 40. Kết quả code python pagerank


* Thực thi trên Gephi

Id	Label	PageR... ▾
0		0.19142
1		0.153493
2		0.0843
3		0.024305
4		0.003817
186		0.003345
103.5		0.003272
115		0.003272
123		0.003272
134		0.003272
172		0.003272
173		0.003272
185		0.002649
160		0.002575
117.9		0.002534
82		0.002534
107		0.002533
110.7		0.002533
120		0.002533

Hình 41. Pagerank trên gephi

4. Eigenvector và Eigenvalue

4.1. Eigenvector centrality

```
 import pandas as pd
import operator

eigenvector_centrality = nx.eigenvector_centrality(G, weight="weight")

data = {
    'total_of_special_requests': eigenvector_centrality.keys(),
    'Eigenvector centrality': eigenvector_centrality.values()
}
eigenvector_centrality_table = pd.DataFrame.from_dict(data)
eigenvector_centrality_table = eigenvector_centrality_table.sort_values(
    by=['Eigenvector centrality', 'total_of_special_requests'],
    ascending=False)

eigenvector_centrality_table.head (15)
```

Hình 42. Code python thuật toán Eigenvector

	<code>total_of_special_requests</code>	<code>Eigenvector centrality</code>
45	172.0	0.110588
110	134.0	0.110588
5	123.0	0.110588
7	97.0	0.110588
3	82.0	0.110588
123	103.5	0.110588
96	169.0	0.106692
55	167.0	0.106692
92	166.0	0.106692
127	164.0	0.106692
12	153.0	0.106692
72	146.0	0.106692
6	145.0	0.106692
27	135.0	0.106692
28	133.0	0.106692

Hình 43. Kết quả code python thuật toán Eigenvector

id	Label	Interval	Eigenvector Centrality ✓
0			1.0
1			0.714728
2			0.437848
123			0.12721
172			0.12721
134			0.12721
103.5			0.12721
173			0.12721
115			0.12721
186			0.123102
107			0.121739
87.3			0.121739
98			0.121739
110.7			0.121739
153			0.121739
135			0.121739
133			0.121739
97			0.121739
73.8			0.121739
131			0.121739
96.3			0.121739
167			0.121739
120.6			0.121739
146			0.121739
120			0.121739
166			0.121739
169			0.121739
164			0.121739
145			0.121739
154			0.121739

Hình 44. Thuật toán Eigenvector trên gephi

4.2. Eigenvalue

```

# Tính toán eigenvalue centrality
eigenvalue_centrality = nx.eigenvector_centrality_numpy(G)

# Sắp xếp kết quả theo thứ tự giảm dần
sorted_values = sorted(eigenvalue_centrality.items(), key=lambda x: x[1], reverse=True)

# In kết quả
for node, value in sorted_values:
    print(f"{node}: {value}")

```

Hình 45. Code python thuật toán Eigenvalue

```

0.0: 0.5715310019031117
1: 0.381026798086837
2: 0.23713859209658503
103.5: 0.06976152580790525
115.0: 0.06976152580790525
123.0: 0.06976152580790522
134.0: 0.06976152580790522
172.0: 0.0697615258079052
173.0: 0.06976152580790519
186.0: 0.06811573297843385
135.0: 0.06771551930861662
166.0: 0.06771551930861662
87.3: 0.0677155193086166
153.0: 0.0677155193086166
131.0: 0.0677155193086166
146.0: 0.0677155193086166

```

Hình 46. Kết quả code python thuật toán Eigenvalue

TÀI LIỆU THAM KHẢO

- [1] [Link Code Python \(Google Colab\)](#)
- [2] [CodePython_Sampleupdate.doc](#)
- [3] [Link Dataset](#)

Thông tin liên hệ

Lê Hoàng Huy	0769699206
Huỳnh Thị Thanh Ngân	0865882023