

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

Môn: Mạng xã hội

Đề tài: Flipkart Mobiles

Lớp: IS353.N11

GVHD: Nguyễn Thị Kim Phụng

Sinh viên: Nguyễn Hữu Thắng - 19522208

Lời cảm ơn

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 em có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt, em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới Cô *Nguyễn Thị Kim Phụng*. Cô đã trực tiếp hướng dẫn tận tình, sửa chữa và đóng góp nhiều ý kiến quý báu giúp em hoàn thành tốt báo cáo môn học của mình.

Trong thời gian một học kỳ thực hiện đề tài, em đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới. Từ đó, em vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, em không tránh khỏi những thiếu sót. Chính vì vậy, em rất mong nhận được những sự góp ý từ phía Cô nhằm hoàn thiện những kiến thức mà em đã học tập và là hành trang để em thực hiện tiếp các đề tài khác trong tương lai.

Xin chân thành cảm ơn các quý Thầy Cô!

[illegible]

Mục lục

Contents

Lời cảm ơn.....	1
Nhận xét của giảng viên	2
1. Tổng quan.....	4
1.1. Giới thiệu	4
1.2. Xác định bài toán	4
2. Dữ liệu.....	5
2.1. Giới thiệu nguồn dữ liệu	5
2.2. Mô tả dữ liệu.....	5
2.3. Xử lý và phân tích dữ liệu.....	6
• Làm sạch dữ liệu.....	6
• Chuyển đổi DataFrame thành đồ thị	7
3. Thuật toán Centrality	13
3.1. Page rank	13
• Gephi	13
• Python.....	13
3.2. Closeness centrality	14
• Gephi	14
• Python.....	14
3.3. Betweenness centrality	15
• Gephi	15
• Python.....	15
4. Thuật toán phân cụm	16
4.1. Louvain.....	16
• Python.....	16
• Gephi	20
4.2. Girvan Newman.....	20
4.3. KMean	22
5. Trích 10 Nodes	26
5.1. Trích xuất.....	26
5.2. Đồ thị 1 phía	27
5.3. Đồ thị 2 phía	29
5.4. Tính tay.....	29
6. Trích 10 Nodes theo yêu cầu của cô	30
6.1. Trích xuất.....	30
6.2. Tính tay.....	31
7. Tham khảo.....	31

1. Tổng quan

1.1. Giới thiệu

Điện thoại ngày nay bất kể từ thương hiệu gì, đều được trang bị nhiều dung lượng RAM khác nhau. Nên mua điện thoại RAM bao nhiêu là đủ? Chúng ta hãy cùng phân tích độ phổ biến sản xuất của các loại dung lượng RAM theo từng hãng (nhãn hiệu).

1.2. Xác định bài toán

- Input: Tập dữ liệu chứa thông số kỹ thuật của nhiều thương hiệu Di động khác nhau ở Ấn Độ đã được lấy từ một trang web thương mại điện tử 'Flipkart'. Bộ dữ liệu này có 2647 mẫu với 8 thuộc tính.
- Output: Đưa ra độ đo, đưa ra cộng đồng phục vụ cho việc phân tích mạng xã hội “Flipkart Mobiles”

2. Dữ liệu

2.1. Giới thiệu nguồn dữ liệu

- Link Dataset: <https://www.kaggle.com/datasets/devsubhash/flipkart-mobiles-dataset>
- Dữ liệu gồm 2647 mẫu với 8 thuộc tính.

2.2. Mô tả dữ liệu

STT	Thuộc tính	Kiểu dữ liệu	Mô tả	Giá trị
1	Brand	Char	Tên của nhà sản xuất điện thoại di động	
2	Model	Char	Số kiểu của Điện thoại di động	
3	Color	Char	Màu sắc của mô hình.	
4	Memory	Char	RAM(4GB,6GB, 8GB, v.v.)	4GB,6GB,8GB, v.v.
5	Storage	Char	ROM(32GB,64GB,128GB,256GB, v.v.)	32GB,64GB,128GB,256GB, v.v.
6	Rating	Int	Xếp hạng của mô hình dựa trên đánh giá (trong số	

			5). Giá trị Thiếu hoặc Không cho biết không có xếp hạng nào cho mô hình.	
7	Selling Price	Int	Giá bán/Giá chiết khấu của mô hình	
8	Original Price		Giá thực tế của kiểu máy tính bằng INR	

Bảng 2.1. Bảng mô tả dữ liệu

2.3. Xử lý và phân tích dữ liệu

- Làm sạch dữ liệu

Bộ dữ liệu nhìn chung khá đầy đủ nhưng có thuộc tính bị thiếu dữ liệu, cần loại bỏ khỏi dataset.

```
import pandas as pd

df = pd.read_csv('/content/Flipkart_mobile_brands_scraped_data.csv', usecols=['Brand','Memory'])

df['Memory'] = df['Memory'].str.replace(' ','')

df['Memory'] = df['Memory'].str.replace('512MB','0.5GB')

df = df.dropna()

df = df.drop_duplicates()

df
```

	Brand	Memory
0	OPPO	4GB
2	OPPO	6GB
7	OPPO	3GB
15	OPPO	8GB
21	OPPO	2GB
...
2483	Xiaomi	8GB
2486	Xiaomi	6GB
2513	Xiaomi	4GB
2522	Xiaomi	2GB
2523	Xiaomi	3GB

122 rows × 2 columns

Hình 2.2. Đọc dữ liệu từ file csv và loại bỏ dữ liệu trùng và rỗng

Dữ liệu sẽ được đọc từ file csv đưa vào dataframe. Sau đó xóa bỏ dữ liệu trùng lặp và rỗng đi. Kết quả nhận được là một bộ dữ liệu 122 dòng và 2 cột

- Chuyển đổi DataFrame thành đồ thị

- Đồ thị 2 phía

- Node: là loại (nhãn hiệu) điện thoại (Brand) và Loại bộ nhớ (Memory)
- Edge: là mối quan hệ giữa việc nhãn hiệu có sản xuất loại điện thoại có bộ nhớ đó không.


```
import networkx as nx
~~~~~
from networkx.algorithms import bipartite
~~~~~
B = nx.Graph()

Brand = df['Brand']

Memory = df['Memory']

print("Brand ", Brand.nunique())

print("Memory", Memory.nunique())

print("Số cạnh", len(df))
```

Brand 16
Memory 28
Số cạnh 122

Hình 2.3. Đưa dữ liệu từ DataFrame vào đồ thị vô hướng

- Có 16 nhãn hiệu
- Số loại bộ nhớ là 28
- Số cạnh là 122

Code hiển thị đồ thị 2 phía:

```
import matplotlib.pyplot as plt

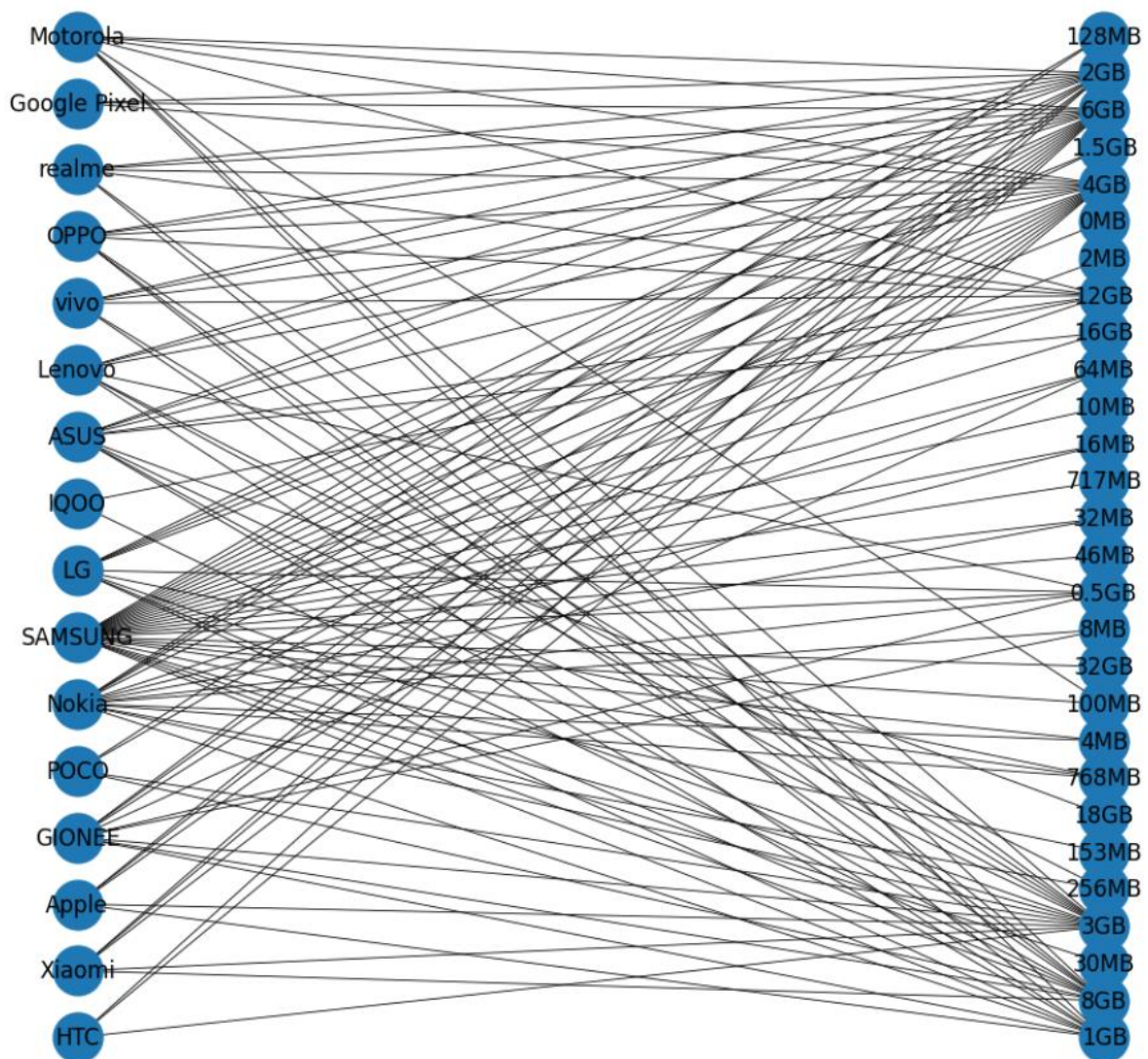
plt.figure(figsize=(12, 12))

pos = nx.spring_layout(B)

fig, ax = plt.subplots(1,1, figsize=(8,8), dpi=150)

nx.draw_networkx(B, pos = nx.drawing.layout.bipartite_layout(B,Brand), font_size=8,width=0.4)
```

Hình 2.4. Code hiển thị đồ thị 2 phía



Hình 2.5. Đồ thị 2 phía

→ Nhìn vào đồ thị ta có thể thấy một nhãn hiệu sản xuất nhiều loại điện thoại có bộ nhớ khác nhau và mỗi loại bộ nhớ cũng được nhiều nhãn hiệu sản xuất.

- Đồ thị 1 phía
 - Node: là các nhãn hiệu (Brand)
 - Edge: Hai nhãn hiệu cùng có sản xuất chung loại bộ nhớ
 - Weight: Trọng số là số loại bộ nhớ khi hai nhãn hiệu sản xuất điện thoại cùng sản xuất.

```
G = bipartite.weighted_projected_graph(B, Brand)

plt.figure(figsize=(12,12))

layout = nx.spring_layout(G)

nx.draw_networkx_nodes(G,
                        layout,
                        nodelist=Brand,
                        node_size=150,
                        node_color='blue')

nx.draw_networkx_edges(G, layout, edge_color="#cccccc")

node_labels = dict(zip(Brand, Brand))

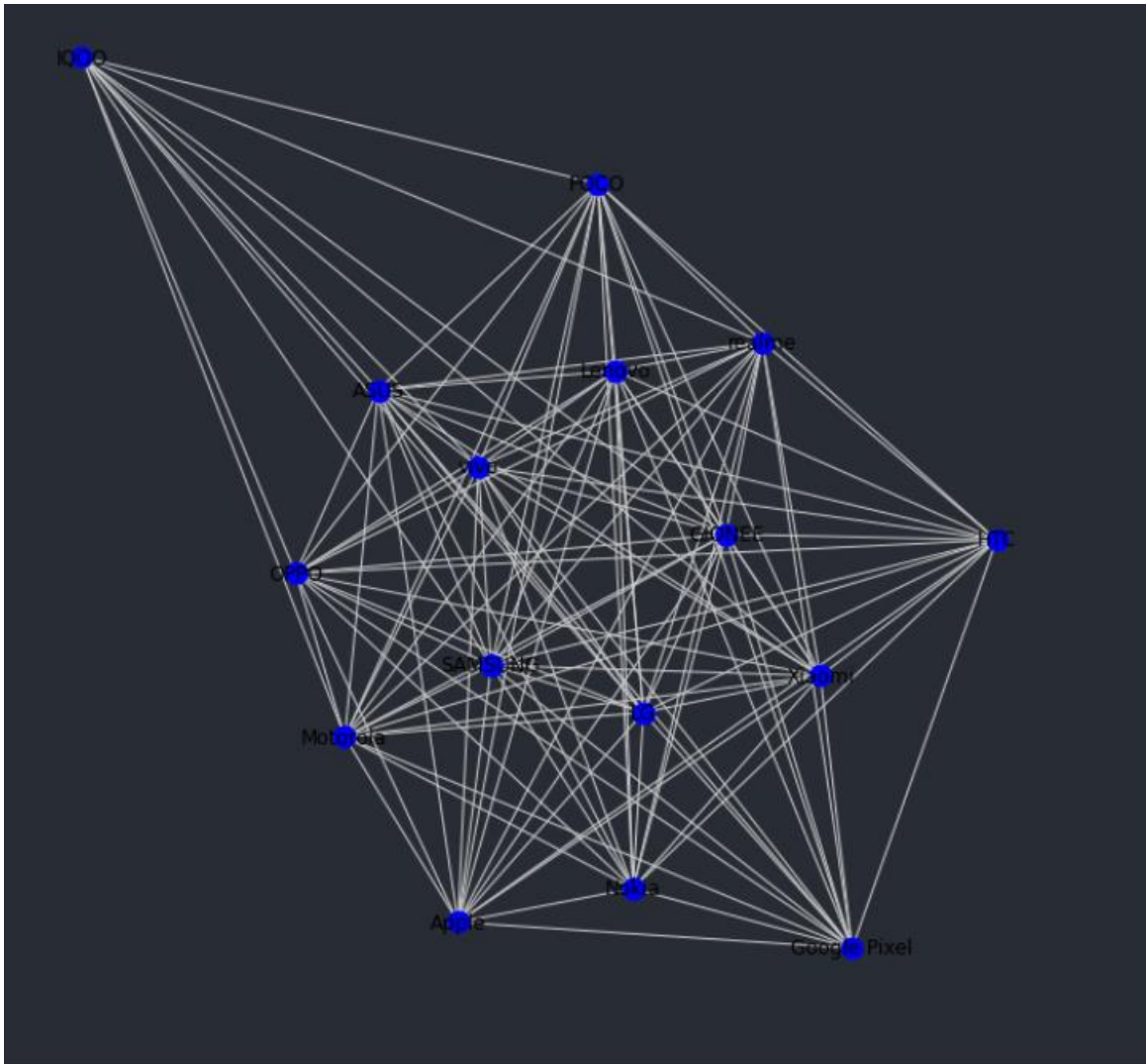
nx.draw_networkx_labels(G, layout, labels=node_labels)

plt.axis('off')

plt.title("Graph Brand")

plt.show()
```

Hình 2.6. Code hiển thị đồ thị 1 phía

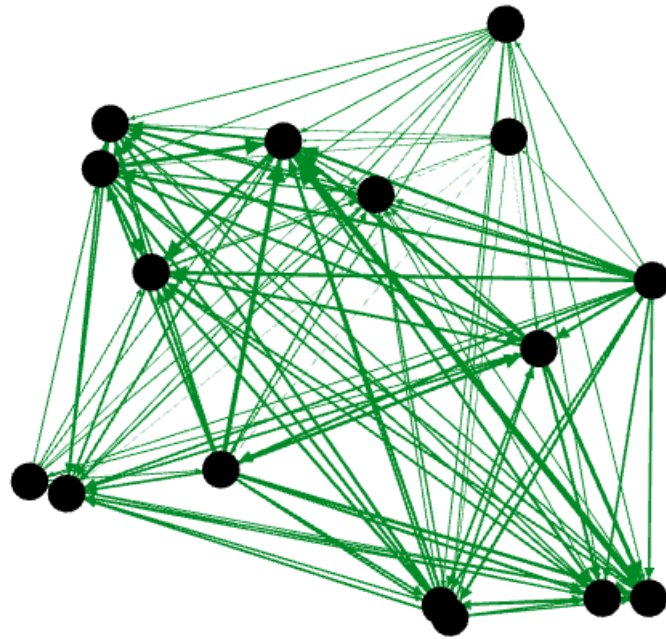


Hình 2.7. Đồ thị 1 phía

- Gephi
- Xuất dữ liệu đồ thị 1 phía ra file csv để thực hiện trên Gephi

```
labels = nx.get_edge_attributes(G, 'weight')  
  
df2 = pd.DataFrame(columns=['source', 'target', 'weight'])  
  
for key, value in labels.items():  
    df2 = df2.append({'source': key[0], 'target': key[1], 'weight': value}, ignore_index=True)  
  
df2.to_csv(r'data_for_gephi.csv', index = False, header=True)
```

- Đồ thị 1 phía trên Gephi



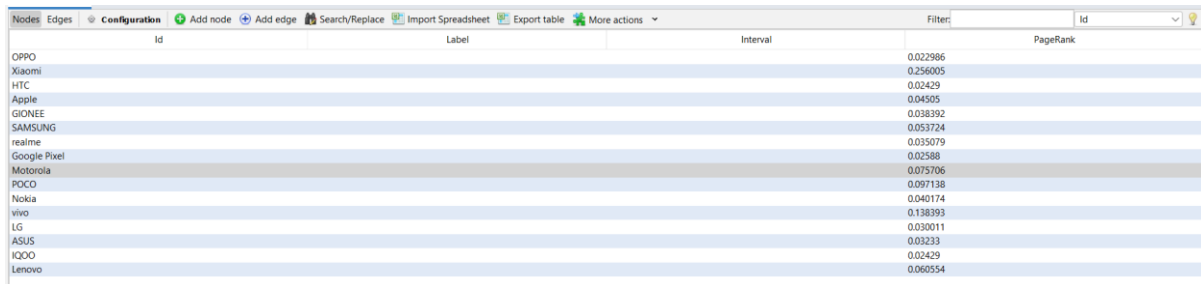
Hình 2.8. Đồ thị 1 phía trên Gephi

3. Thuật toán Centrality

3.1. Page rank

- Gephi

Kết quả tính page rank trên gephi



Id	Label	Interval	PageRank
OPPO			0.022966
Xiaomi			0.256005
HTC			0.02429
Apple			0.04505
GIONEE			0.038392
SAMSUNG			0.053724
realme			0.035079
Google Pixel			0.02588
Motorola			0.075706
POCO			0.097138
Nokia			0.040174
vivo			0.138393
LG			0.030011
ASUS			0.03233
IQOO			0.02429
Lenovo			0.060554

- Python

Kết quả tính page rank bằng python

Page rank

```
page_rank = nx.pagerank(G)

for node in sorted(page_rank, key=page_rank.get, reverse=True):
    print(node, page_rank[node])
```

```
SAMSUNG 0.08531075379207942
ASUS 0.0739418210069697
Motorola 0.0739418210069697
GIONEE 0.07354263189259955
LG 0.07353168685902728
OPPO 0.07314214956945492
Nokia 0.07058887693220424
realme 0.06661806279213572
vivo 0.06661806279213572
Lenovo 0.06334485836314034
Xiaomi 0.06136588055369238
Apple 0.059301305555650496
POCO 0.05146043907195641
HTC 0.04289833994110801
Google Pixel 0.04203297592229822
IQOO 0.022360333948577914
```

3.2. Closeness centrality

- Gephi

Nodes	Edges	Configuration	Add node	Add edge	Search/Replace	Import Spreadsheet	Export table	More actions
Id	Label	Interval	Pa...	Cluster...	Eigenv...	Infer...	Eccen...	Closeness Centrality ^
Xiaomi			0.2...	0.47619	1.0	6	0.0	0.0
IQOO			0.0...	0.5	0.0027...	2	2.0	0.8
realme			0.0...	0.47619	0.0579...	0	1.0	1.0
LG			0.0...	0.47619	0.0208...	0	1.0	1.0
SAMSUNG			0.0...	0.47619	0.2483...	1	1.0	1.0
ASUS			0.0...	0.47619	0.0360...	1	1.0	1.0
GIONEE			0.0...	0.47619	0.0882...	2	1.0	1.0
OPPO			0.0...	0.47619	0.0	3	1.0	1.0
Apple			0.0...	0.5	0.1759...	3	1.0	1.0
Google Pixel			0.0...	0.5	0.0077...	4	1.0	1.0
vivo			0.1...	0.47619	0.7752...	4	1.0	1.0
Nokia			0.0...	0.5	0.1243...	5	1.0	1.0
Lenovo			0.0...	0.5	0.3333...	5	1.0	1.0
Motorola			0.0...	0.47619	0.4501...	6	1.0	1.0
HTC			0.0...	0.5	0.0027...	7	1.0	1.0
POCO			0.0...	0.47619	0.59443	7	1.0	1.0

- Python

Kết quả tính Closeness centrality bằng python

```
Closeness centrality

closeness_centrality = nx.closeness_centrality(G)

for node in sorted(closeness_centrality, key=closeness_centrality.get, reverse=True):
    print(node, closeness_centrality[node])

OPPO 1.0
LG 1.0
ASUS 1.0
realme 1.0
GIONEE 1.0
SAMSUNG 1.0
Motorola 1.0
POCO 1.0
vivo 1.0
Xiaomi 1.0
HTC 0.9375
Google Pixel 0.9375
Nokia 0.9375
Apple 0.9375
Lenovo 0.9375
IQOO 0.75
```

3.3. Betweenness centrality

- Gephi

Id	Label	Interval	Pa...	Cluster...	Eigenv...	Infer...	Eccen...	Clos...	Harmoni...	Betweenness Centrality
Xiaomi			0.2...	0.47619	1.0	6	0.0	0.0	0.0	0.0
IQOO			0.0...	0.5	0.0027...	2	2.0	0.8	0.875	0.0
realme			0.0...	0.47619	0.0579...	0	1.0	1.0	1.0	0.003333
LG			0.0...	0.47619	0.0208...	0	1.0	1.0	1.0	0.003333
SAMSUNG			0.0...	0.47619	0.2483...	1	1.0	1.0	1.0	0.000952
ASUS			0.0...	0.47619	0.0360...	1	1.0	1.0	1.0	0.003333
GIONEE			0.0...	0.47619	0.0882...	2	1.0	1.0	1.0	0.003333
OPPO			0.0...	0.47619	0.0	3	1.0	1.0	1.0	0.0
Apple			0.0...	0.5	0.1759...	3	1.0	1.0	1.0	0.0
Google Pixel			0.0...	0.5	0.0077...	4	1.0	1.0	1.0	0.0
vivo			0.1...	0.47619	0.7752...	4	1.0	1.0	1.0	0.0
Nokia			0.0...	0.5	0.1243...	5	1.0	1.0	1.0	0.0
Lenovo			0.0...	0.5	0.3333...	5	1.0	1.0	1.0	0.0
Motorola			0.0...	0.47619	0.4501...	6	1.0	1.0	1.0	0.0
HTC			0.0...	0.5	0.0027...	7	1.0	1.0	1.0	0.0
POCO			0.0...	0.47619	0.59443	7	1.0	1.0	1.0	0.0

- Python

Betweenness centrality

```
#
betweenness_centrality = nx.betweenness_centrality(G)

for node in sorted(betweenness_centrality, key=betweenness_centrality.get, reverse=True):
    print(node, betweenness_centrality[node])
```

```
OPPO 0.0047619047619047615
LG 0.0047619047619047615
ASUS 0.0047619047619047615
realme 0.0047619047619047615
GIONEE 0.0047619047619047615
SAMSUNG 0.0047619047619047615
Motorola 0.0047619047619047615
POCO 0.0047619047619047615
vivo 0.0047619047619047615
Xiaomi 0.0047619047619047615
HTC 0.0
IQOO 0.0
Google Pixel 0.0
Nokia 0.0
Apple 0.0
Lenovo 0.0
```


4. Thuật toán phân cụm

4.1. Louvain

- Python

```
Louvain

import matplotlib.cm as cm

import matplotlib

import community.community_louvain as community_louvain
~~~~~
import matplotlib.pyplot as plt

plt.figure(figsize=(17, 12))

partition = community_louvain.best_partition(G)

pos = nx.spring_layout(G)

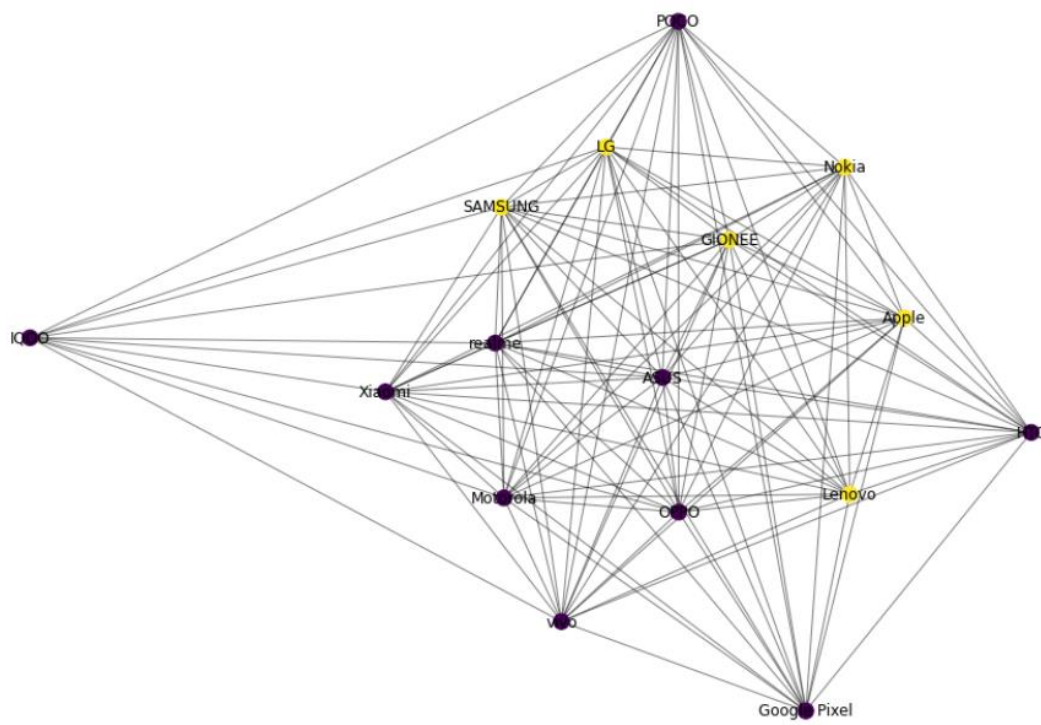
cmap= cm.get_cmap('viridis', max(partition.values()) +1)

nx.draw_networkx_nodes(G, pos, partition.keys(), node_size=150,
| | | | | |      cmap=cmap, node_color=list(partition.values()))
| | | | | |
nx.draw_networkx_edges(G, pos, alpha=0.5)

nx.draw_networkx_labels(G, pos)

plt.show()
```

Hình 4.1. Code thuật toán phân cụm Louvain



Hình 4.2. Đồ thị phân cụm Louvain

```
• import numpy as np

values = list(partition.values())

print('Số cụm', len(np.unique(values)))

✓ for i in range(len(np.unique(values))):
    | print("-- Cụm ", i+1, " --")
    |
    | ✓ for name, k in partition.items():
    |
    | ✓ if k == i:
    |     | print(name)
```

Số cụm 2
-- Cụm 1 --
OPPO
HTC
IQOO
Google Pixel
ASUS
realme
Motorola
POCO
vivo
Xiaomi
-- Cụm 2 --
LG
GIONEE
Nokia
Apple

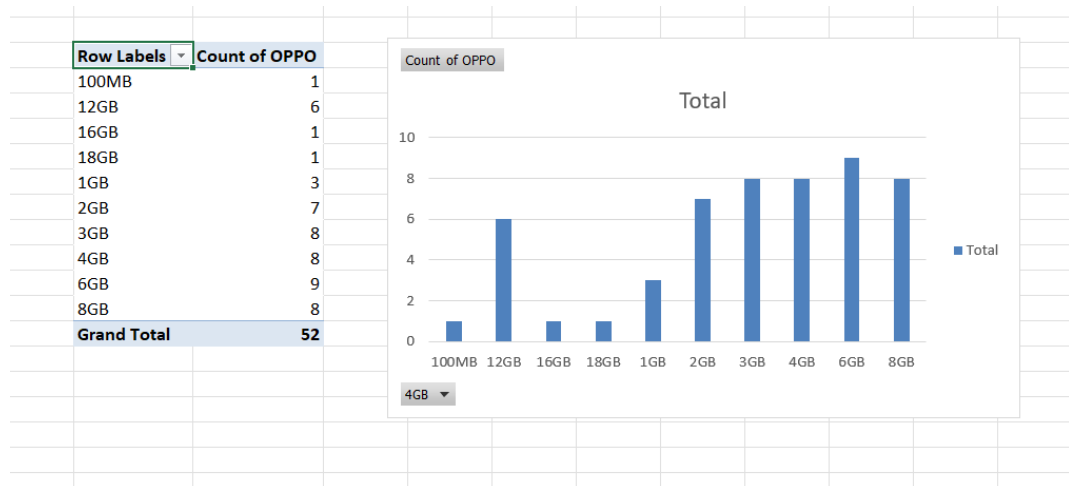
Hình 4.3. Code tính số cụm và nội dung cụm

- Số cụm: 2

→ -- Cụm 1 --

- OPPO
- HTC
- IQOO

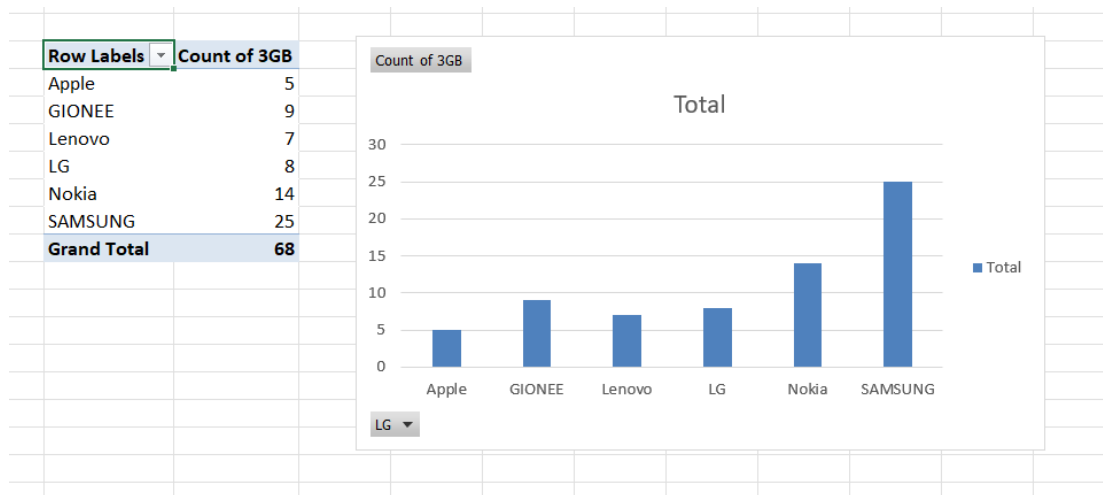
- Google Pixel
- ASUS
- realme
- Motorola
- POCO
- vivo
- Xiaomi



Hình 4.4. vẽ biểu đồ cho cụm 1 biểu diễn số loại bộ nhớ được sản xuất

→ -- Cụm 2 --

- LG
- GIONEE
- Nokia
- Apple
- SAMSUNG
- Lenovo



Hình 4.5. vẽ biểu đồ cho cụm 2 biểu diễn loại bộ nhớ được sản xuất theo từng nhãn hiệu

- Ý nghĩa cụm 1: các hãng sản xuất điện thoại 2GB, 3GB, 4GB, 6GB, 8GB, 12GB
- Ý nghĩa cụm 2: các hãng sản xuất điện thoại 30MB, 256MB, 0.5GB, 1GB

- Gephi

Hình 4.4. Đồ thị phân cụm Louvain sử dụng Gephi

4.2. Girvan Newman

- Code thực thi

```

from networkx.algorithms.community centrality import girvan_newman
~~~~~

communities = girvan_newman(G)

node_groups = []

for com in next(communities):
    node_groups.append(list(com))

print(node_groups)

color_map = []

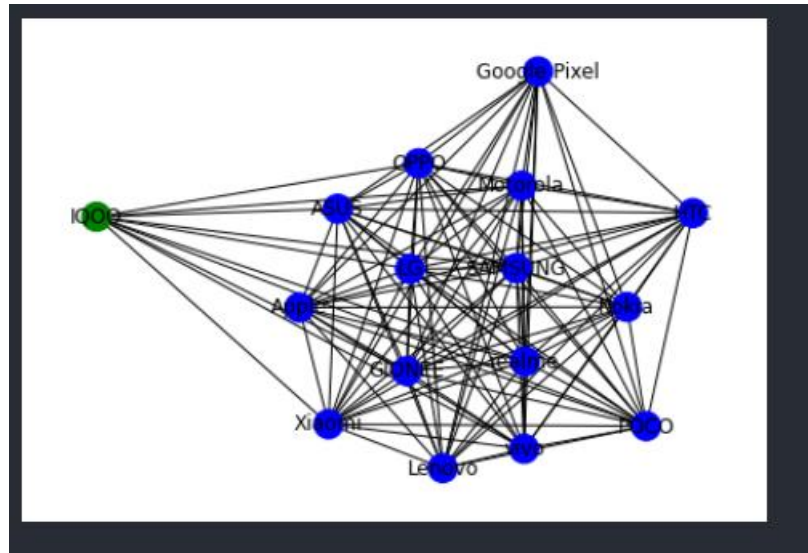
for node in G:
    if node in node_groups[0]:
        color_map.append('blue')
    else:
        color_map.append('green')

nx.draw(G, node_color=color_map, with_labels=True)

plt.show()

```

- Trực quan hóa



Hình 4.5. Trực quan hóa Girvan Newman

- Số cụm: 2
- Cụm 1 – xanh blue: ['Xiaomi', 'HTC', 'Apple', 'GIONEE', 'POCO', 'Nokia', 'SAMSUNG', 'vivo', 'LG', 'ASUS', 'Lenovo', 'OPPO', 'realme', 'Google Pixel', 'Motorola']
- Cụm 2 – xanh green: ['IQOO']
- *Chỉ có 1 nhãn hiệu sản xuất điện thoại có bộ nhớ 30MB là 'IQOO'*

4.3. KMean

- Code thực thi

```
def Label_enc(feat):
    LabelE = LabelEncoder()

    LabelE.fit(feat)

    print(feat.name, LabelE.classes_)

    return LabelE.transform(feat)
```

```
for col in df.columns:

    df[str(col)] = Label_enc(df[str(col)])
```

```
N_Brand ['ASUS' 'Apple' 'GIONEE' 'Google Pixel' 'HTC' 'IQOO' 'LG' 'Lenovo'
'Motorola' 'Nokia' 'OPPO' 'POCO' 'SAMSUNG' 'Xiaomi' 'realme' 'vivo']
N_Memory ['0.5GB' '0MB' '1.5GB' '100MB' '10MB' '128MB' '12GB' '153MB' '16GB' '16MB'
'18GB' '1GB' '256MB' '2GB' '2MB' '30MB' '32GB' '32MB' '3GB' '46MB' '4GB'
'4MB' '64MB' '6GB' '717MB' '768MB' '8GB' '8MB']
```



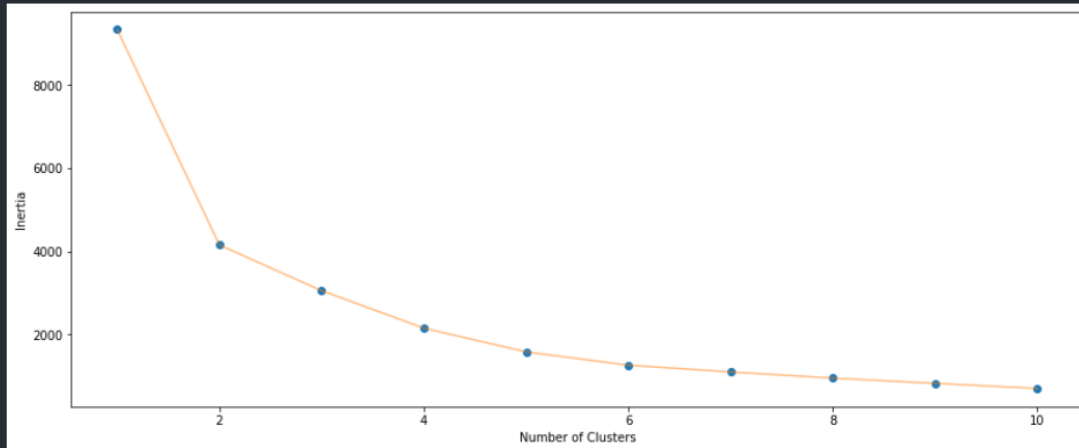
```
plt.figure(1, figsize = (15,6))

plt.plot(np.arange(1,11), inertia, 'o')

plt.plot(np.arange(1,11), inertia, '-',alpha=0.5)

plt.xlabel('Number of Clusters'), plt.ylabel('Inertia')

plt.show()
```



Từ biểu đồ trên có thể xác nhận, số cụm tối ưu là 2 ($k=2$)

Hình 4.6. Kết quả thực thi phương pháp ELBOW

- Code thực thi với số cụm $k = 2$

```

• algorithm = KMeans(n_clusters =2, init='random')

algorithm.fit(X)

labels1 = algorithm.labels_

centroids1 = algorithm.cluster_centers_

#Biểu diễn kết quả gom cụm

plt.figure(1, figsize = (15,7))

plt.clf()

plt.scatter(x='N_Memory', y='N_Brand', data=df02, s=200, c=labels1)

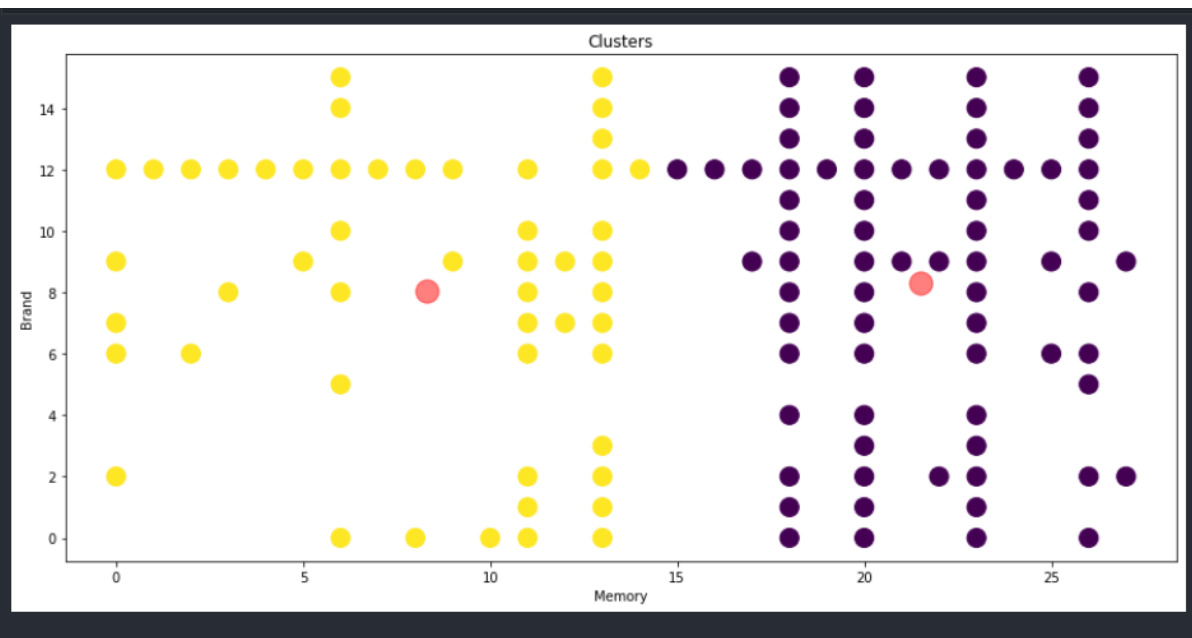
plt.scatter(centroids1[:,0], centroids1[:,1], s=300, c='red', alpha=0.5)

plt.xlabel('Memory'), plt.ylabel('Brand')

plt.title('Clusters')

plt.show()

```



Hình 4.7. Kết quả thuật toán KMean

5. Trích 10 Nodes

5.1. Trích xuất Từ data ban đầu

```
import networkx as nx
from networkx.algorithms import bipartite

B = nx.Graph()

Brand = df['Brand']
Memory = df['Memory']

print("Brand ", Brand.nunique())
print("Memory", Memory.nunique())
print("Số cạnh", len(df))
```

Brand 16
Memory 28
Số cạnh 122

Hình 5.1

Chọn ra 10 nhãn hiệu

```
df_count = df_count.sort_values('count').head(10)
```

df_count

	Brand	count
15	IQOO	2
13	HTC	3
14	Google Pixel	3
12	POCO	4
10	Apple	5
11	Xiaomi	5
8	realme	6
9	vivo	6
6	OPPO	7
7	Lenovo	7

Hình 5.2

Mapping với cột Memory để lấy đủ hết data theo 10 nhãn hiệu vừa chọn

```
df = df[df['Brand'].isin(df_count['Brand'])]
```

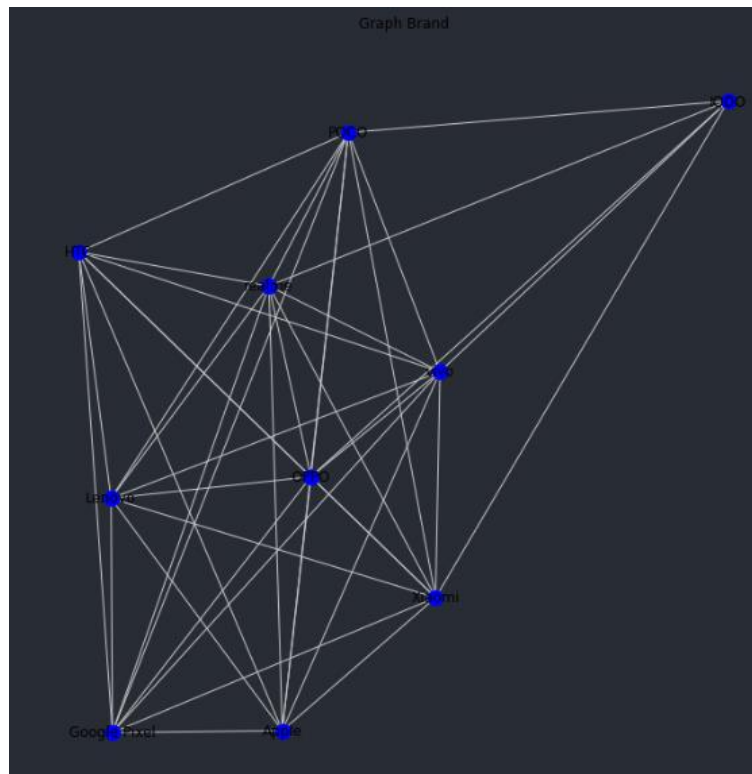
df

	Brand	Memory
0	OPPO	4GB
2	OPPO	6GB
7	OPPO	3GB
15	OPPO	8GB
21	OPPO	2GB
48	OPPO	12GB
115	OPPO	1GB
244	HTC	6GB
245	HTC	3GB
246	HTC	4GB
249	IQOO	8GB
253	IQOO	12GB
254	Google Pixel	6GB
255	Google Pixel	4GB
311	Google Pixel	2GB
495	realme	4GB
496	realme	2GB
506	realme	6GB
516	realme	8GB
521	realme	3GB
569	realme	12GB
1121	Apple	2GB

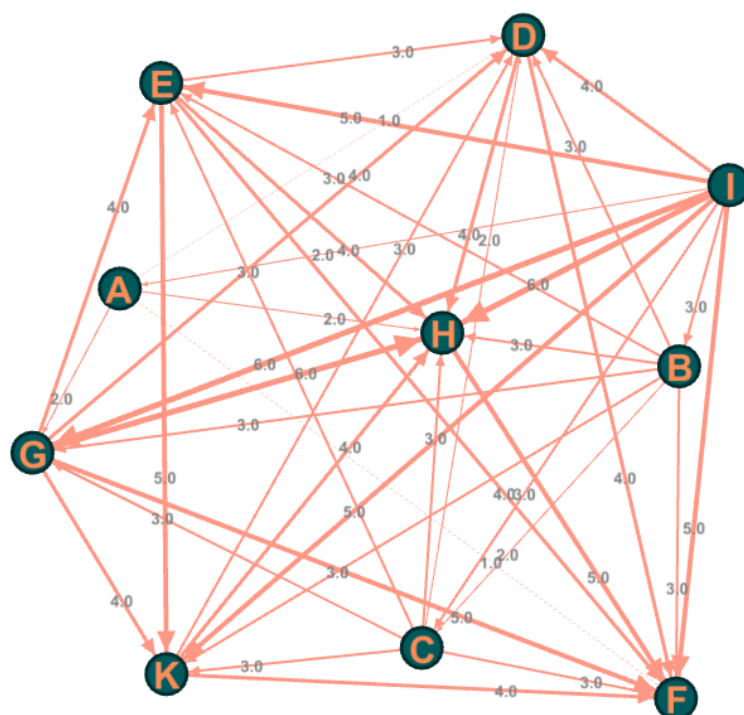
Hình 5.3

5.2. Đồ thị 1 phía

Code tương tự như ở phần data đầy đủ, ta có thể xuất được đồ thị 1 phía với 10 nodes đã chọn



Hình 5.3. Đồ thị 1 phía xuất bằng python



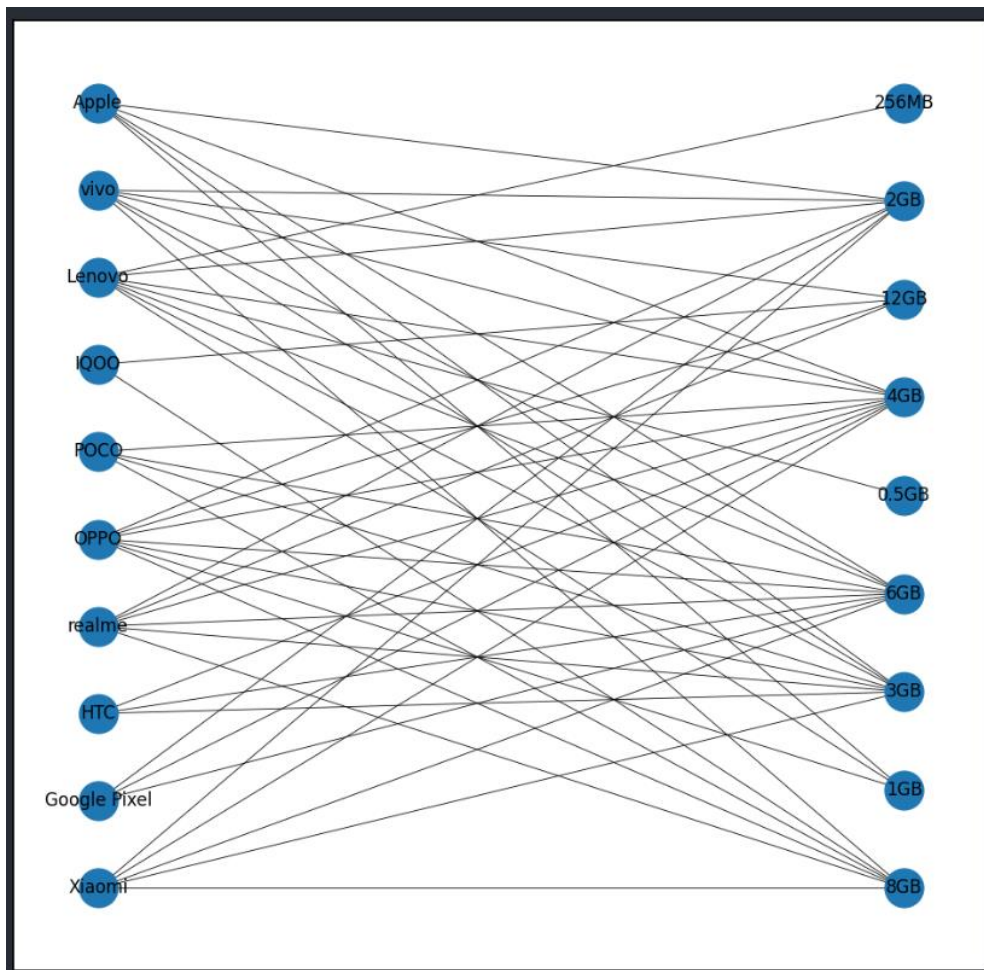
Hình 5.3. Đồ thị 1 phía xuất bằng gephi

Chú thích

Id	Label
OPPO	I
Xiaomi	F
Google Pixel	C
HTC	B
realme	G
POCO	D
IQOO	A
Lenovo	K
vivo	H
Apple	E

5.3. Đồ thị 2 phía

Tương tự ta có đồ thị 2 phía



5.4. Tính tay

- Đồ thị và ma trận kề
- Closeness
- Betweenness

- Kmean
- Page rank
- Eigen Vector

6. Trích 10 Nodes theo yêu cầu của cô

6.1. Trích xuất

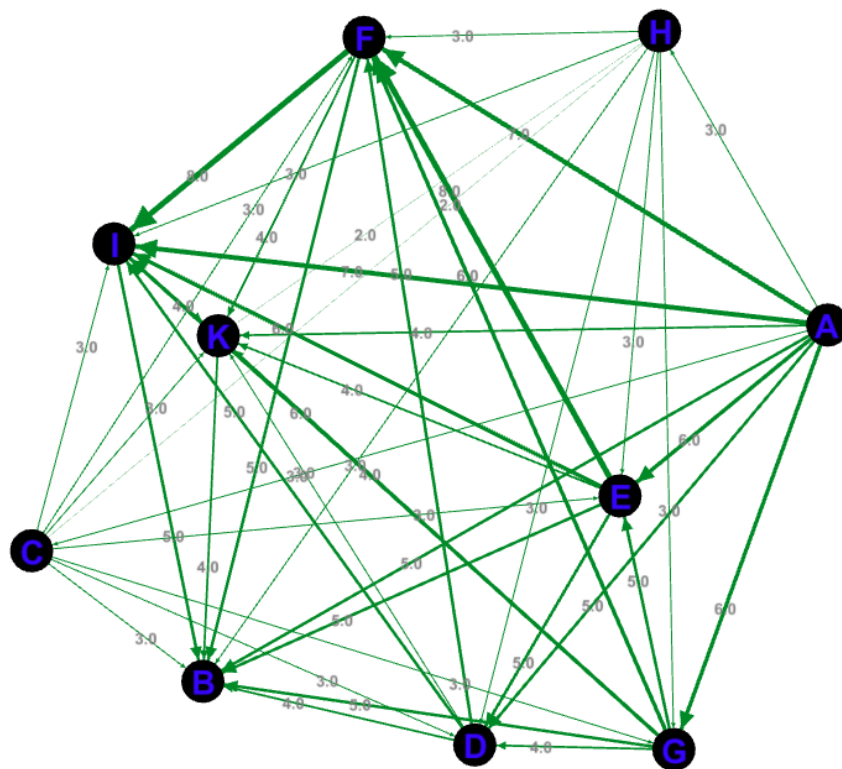
Từ danh sách tất cả các nodes ban đầu

Id	Label	Interval	PageRank	Clustering Coefficient	Eigenvector Centrality	Inferred Class	Eccentricity	Closeness Centrality	Harmonic Closeness Centrality	Betweenness Centrality
OPPO		0.022986	0.47619	0.0	3	1.0	1.0	1.0	0.0	0.0
Xiaomi		0.256005	0.47619	1.0	6	0.0	0.0	0.0	0.0	0.0
HTC		0.02429	0.5	0.002774	7	1.0	1.0	1.0	0.0	0.0
Apple		0.04505	0.5	0.175946	3	1.0	1.0	1.0	0.0	0.0
GIONEE		0.038392	0.47619	0.088251	2	1.0	1.0	1.0	0.003333	0.0
SAMSUNG		0.053724	0.47619	0.248319	1	1.0	1.0	1.0	0.000952	0.0
realme		0.035079	0.47619	0.057931	0	1.0	1.0	1.0	0.003333	0.0
Google Pixel		0.02588	0.5	0.007759	4	1.0	1.0	1.0	0.0	0.0
Motorola		0.075706	0.47619	0.450158	6	1.0	1.0	1.0	0.0	0.0
POCO		0.097138	0.47619	0.59443	7	1.0	1.0	1.0	0.0	0.0
Nokia		0.040174	0.5	0.124364	5	1.0	1.0	1.0	0.0	0.0
vivo		0.138393	0.47619	0.775236	4	1.0	1.0	1.0	0.0	0.0
LG		0.030011	0.47619	0.020801	0	1.0	1.0	1.0	0.003333	0.0
ASUS		0.03233	0.47619	0.036082	1	1.0	1.0	1.0	0.003333	0.0
IQOO		0.02429	0.5	0.002774	2	2.0	0.8	0.875	0.0	0.0
Lenovo		0.060554	0.5	0.333302	5	1.0	1.0	1.0	0.0	0.0

Lọc tay ra 10 nodes bất kì và bỏ sang Workspace mới

Id	Label	Interval	PageRank	Clustering Coefficient	Eigenvector Centrality	Inferred Class	Eccentricity	Closeness Centrality	Harmonic Closeness Centrality	Betweenness Centrality
OPPO	A	0.022986	0.47619	0.0	3	1.0	1.0	1.0	0.0	0.0
Xiaomi	B	0.256005	0.47619	1.0	6	0.0	0.0	0.0	0.0	0.0
HTC	C	0.02429	0.5	0.002774	7	1.0	1.0	1.0	0.0	0.0
Apple	D	0.04505	0.5	0.175946	3	1.0	1.0	1.0	0.0	0.0
GIONEE	E	0.038392	0.47619	0.088251	2	1.0	1.0	1.0	0.003333	0.0
SAMSUNG	F	0.053724	0.47619	0.248319	1	1.0	1.0	1.0	0.000952	0.0
realme	G	0.035079	0.47619	0.057931	0	1.0	1.0	1.0	0.003333	0.0
Google Pixel	H	0.02588	0.5	0.007759	4	1.0	1.0	1.0	0.0	0.0
Motorola	I	0.075706	0.47619	0.450158	6	1.0	1.0	1.0	0.0	0.0
POCO	K	0.097138	0.47619	0.59443	7	1.0	1.0	1.0	0.0	0.0

Đồ thị trên gephi



6.2. Tính tay

- Đồ thị và ma trận kề
- Closeness
- Betweenness
- Kmean
- Eigen Vector

7. Tham khảo

- Centrality-Measures_Include(Eigenvectors and Eigenvalues).pdf (course)
- <https://machinelearningcoban.com/2017/01/01/kmeans/>
- <https://gtvseo.com/pagerank-la-gi/>

