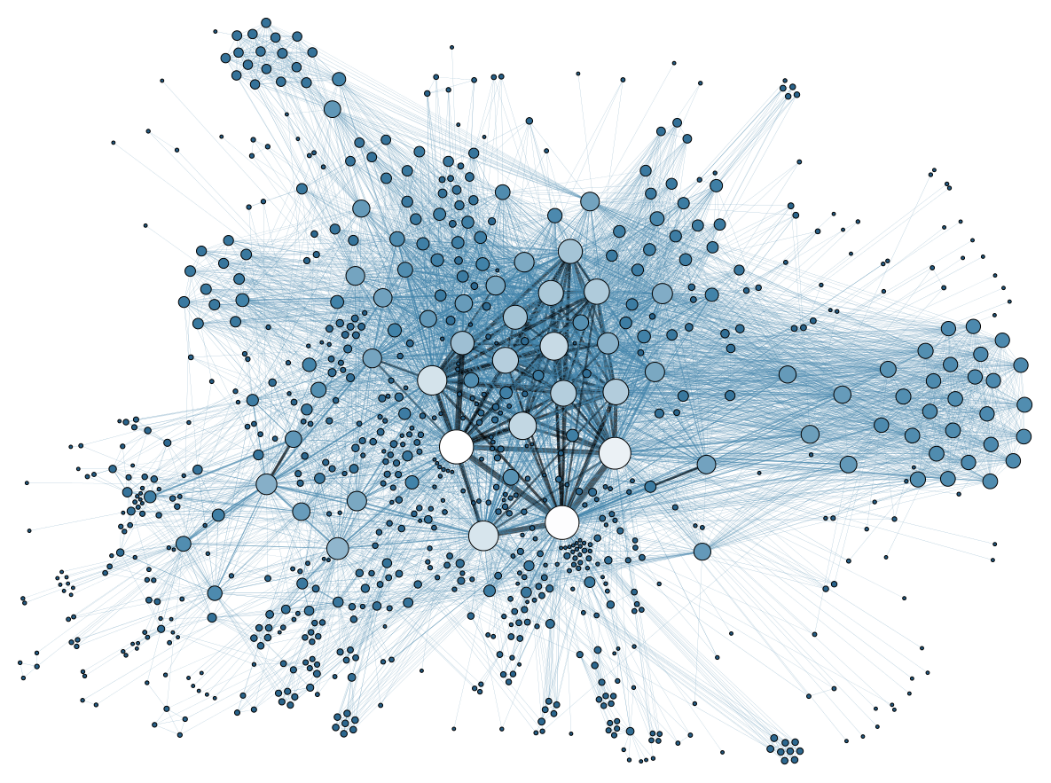


**Rapport du projet**

Matière :

Analyse des réseaux sociaux

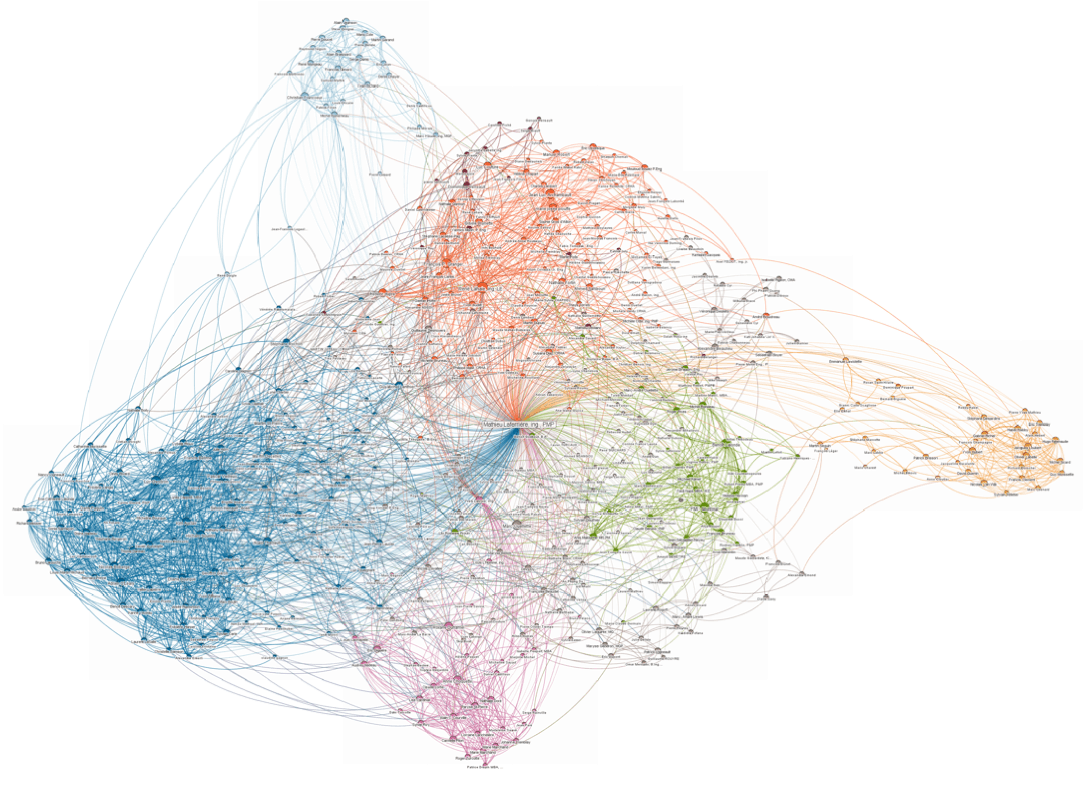


Réaliser par : Zhioua Mahmoud

2022-2023

**Analyse des réseaux sociaux**

L'analyse des réseaux sociaux est le processus d'investigation des structures sociales à travers l'utilisation des réseaux et de la théorie des graphes. Il caractérise les structures en réseau en termes de nœuds et de liens, d'arêtes ou de liens qui les relient.



Les données contiennent un total de 918 instances (ou lignes) avec 12 attributs (ou colonnes). Sur ces 12 attributs, 5 sont de nature catégorielle et 7 sont de nature numérique.

En premier lieu, nous avons importé les bibliothèques nécessaires pour l'analyse de données et la visualisation graphique. Nous avons utilisé la bibliothèque pandas pour charger la datait et des maisons, qui sont stockées dans un fichier CSV

# Charger la dataset en utilisant la bibliothèque Pandas

df = pd.read\_csv(heart.csv')

La première étape dans la construction du graphe consiste à ajouter les nœuds. Cela se fait en parcourant la dataset et en ajoutant chaque maison et région unique comme un nœud du graphe. La méthode set() est utilisée pour éliminer les doublons dans les colonnes de la dataset.

# Ajouter les nœuds

for node in set(df['Id']).union(set(df['Label'])):

G.add\_node(node)

Ensuite, les liens entre les identifiants et les labels sont ajoutés au graphe. Cela se fait en parcourant la dataset à nouveau et en ajoutant une arête entre chaque label et la id correspond.

# Ajouter les liens

for i, row in df.iterrows():

G.add\_edge(row['Id'], row['Label'])

Apres avoir ajouté les liens , on va afficher le graphe correspond

# Afficher le graphe

nx.draw(G, with\_labels=True)

plt.show()

Après avoir le graphe, plusieurs analyses sont effectuées pour mieux comprendre sa structure et ses propriétés. La première mesure calculée est la centralité degré de chaque nœud, qui est la fraction de nœuds avec lesquels un nœud est connecté directement. Cette mesure permet de déterminer les nœuds les plus centraux dans le réseau, qui sont souvent les plus importants

# Calculer la centralité degré

degree\_centrality = nx.degree\_centrality(G)

print("Centralité degré :", degree\_centrality)

La deuxième mesure calculée est la centralité intermédiarité de chaque nœud, qui mesure le nombre de fois qu'un nœud apparaît sur les plus courts chemins entre deux autres nœuds du graphe.

# Calculer la centralité intermédiarité

betweenness\_centrality = nx.betweenness\_centrality(G)

print("Centralité intermédiarité :", betweenness\_centrality)

Ensuite, la densité du graphe est calculée, qui mesure le nombre de liens existants par rapport au nombre total possible de liens dans le graphe. Cette mesure est importante car elle donne une idée de la densité globale du réseau.

# Analyse de la densité

density = nx.density(G)

print("Densité du graphe :", density)

Le coefficient de clustering est également calculé. Cette mesure mesure la probabilité que les voisins d'un nœud soient également connectés entre eux.

# Analyse du coefficient de clustering

clustering\_coefficient = nx.average\_clustering(G)

print("Coefficient de clustering moyen :", clustering\_coefficient)

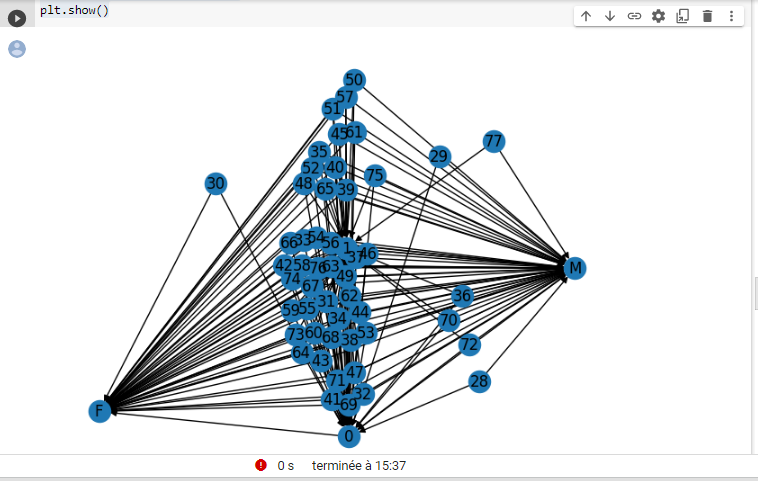
Dans cette commande, on va analyser les composants qui sont connectés

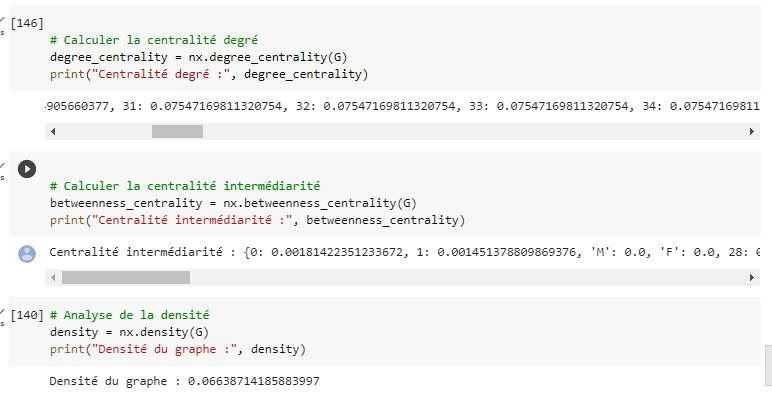
# Analyse des composants connectés

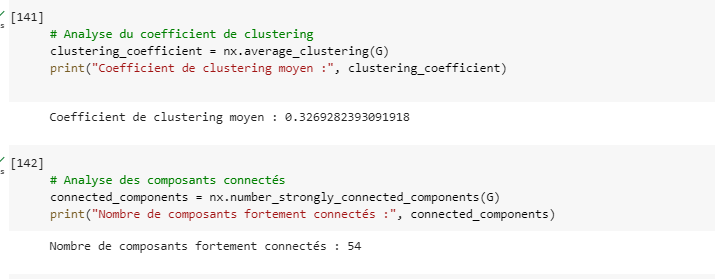
connected\_components = nx.number\_strongly\_connected\_components(G)

print("Nombre de composants fortement connectés :", connected\_components)

Voilà les résultats obtenus :







Maintenant, on a choisi trois algorithmes de détection de communautés sont appliqués sur le graphe. Le premier algorithme K-clique est utilisé pour détecter des groupes de nœuds qui ont au moins k liens entre eux.

# Algorithme K-clique

k\_clique = list(k\_clique\_communities(G, 2))

print("Communautés K-clique :", k\_clique)

Deuxième algorithme c’est la propagation de labels cherche à partitionner le graphe en groupes denses et fortement connectés.

# Propagation des labels

label\_propagation\_communities = label\_propagation.label\_propagation\_communities(G)

print("Communautés par propagation des labels :", label\_propagation\_communities)

Finalement, l'algorithme de Louvain cherche également à partitionner le graphe en groupes, mais en maximisant la modularité du graphe.

#Louvain

louvain\_communities = community.greedy\_modularity\_communities(G)

print("Communautés Louvain :", louvain\_communities)

Malheureusement cette partie du code n’est pas exécutable :



**Conclusion**

Ces mesures et algorithmes permettent d'analyser en détail la structure réseau de notre base de données et les relations entre son identifiant et ses étiquettes. Cette analyse peut aussi s'avérer utile dans la prévision d'événements futurs.