Final Project Report

Luyi Huang lh311

1. Background

1.1 Introduction

The problem for this project comes from the intuition when I was completing my K-means homework. For K-means, we mainly assign all points to the closet cluster center using different metric spaces for distance. The goal for the algorithm is to minimize the cost. But what I am confused about is that it does not tell anything about the relation between each point. Some data is OK with it, but in some cases, K-means is not enough. This is because it cannot tell people the correlation between each point. Imagine that each point represents a person, using K means can not tell us anything except for the fact that some people are closer to each other because of some attributes. But people need interaction. They can not just stay at the specific point on the K-means plot without talking to others. Maybe some people will hate others who have the exact same attributes as them, and extremely likes the opposite features. For instance, for heterosexual person, it is not reasonable to assume that she will have an intimate relation with another woman. Thus, doing clustering totally on attributes is not enough. And I am planning to learn some new things to answer my question.

After careful consideration, I noticed that there exists an algorithm, which is network analysis, that can perfectly solve my question. Thus, for my project, I first tried to use K-Modes(I use KModes because nearly all of my data are categorical data, this means using KMeans is not suitable), and then, I moved on to a totally new algorithm for me, which is Network Analysis to analyze the data I found on Kaggle. I believed network analysis is interesting and of vital importance for us,

since in addition to do clustering, we need to understand how people in different clusters are interacting with others and visualize their relation. Besides, I guess this process will earn huge popularity in the future.

1.2 Data Description

The data I used was data describing *"Risk of being drawn into online sex work-Detecting individuals at risk using semi-supervised learning"* by Panos Kostakos on Kaggle. It comes from the study in Oulu University, Finland. The data was used to predict the risk of getting into online sex work for users of social media. And researchers in that research concluded 78 agencies risk level by doing ethnography interview. Based on this train data, they made a prediction on other users on social media about their potential risk of getting into online sex work.

Overall, the data has 18 variables, which are

User_ID Gender

Age Location

Verification Sexual-Orientation.

Looking for: The gender he is looking for

For my use, I made some data processing and feature selecting. Unfortunately, the data of risk contained too much "unspecified risk". It composes nearly 97% of the data. Thus, I had to abandon this feature since it makes nearly no influence on prediction on test data.

Also, since I needed to do network analysis, I had to have data describing the interaction between people. This requirement could be satisfied using the feature "friends ID". This variable made a list on ID of friends the user is following and forms a perfect source to analysis the interaction between people. However, there was a missing data issue that hindered my data

processing. The situation was that some people are not having friend followed and also are not following others. Adding these people to network might largely reduced the interpretability of network and will make the graph probably a mess. Thus, I deleted these noises by deleting all data that have null value in friend_ID.

Also, there was some problem of missing data on other variables. For instance, there are some missing values on looking-for. But luckily, I could use sexual orientation to predict the values of looking for. For instance, if the agency is male and is heterosexual, he may look for a woman instead of nobody or male.

As for feature selecting, since my main interest was to analyze the relation between people and take a look at who may be in the center of the relation network and what their attributes may be, I used several variable as attributes. I have selected gender, sexual-orientation, looking for, Age, location as attributes. The reason I selected them was that these factors may be decisive in these social media, especially on occasion when they are used for blind dating or sex transaction

1.3 Previous work

I have served the Kaggle. Up to now, no people have done network analysis or K-means on this data, and as for the original paper, the author mainly wanted to predict the risk level of agency, which indicated that they also did not use K-means or network analysis.

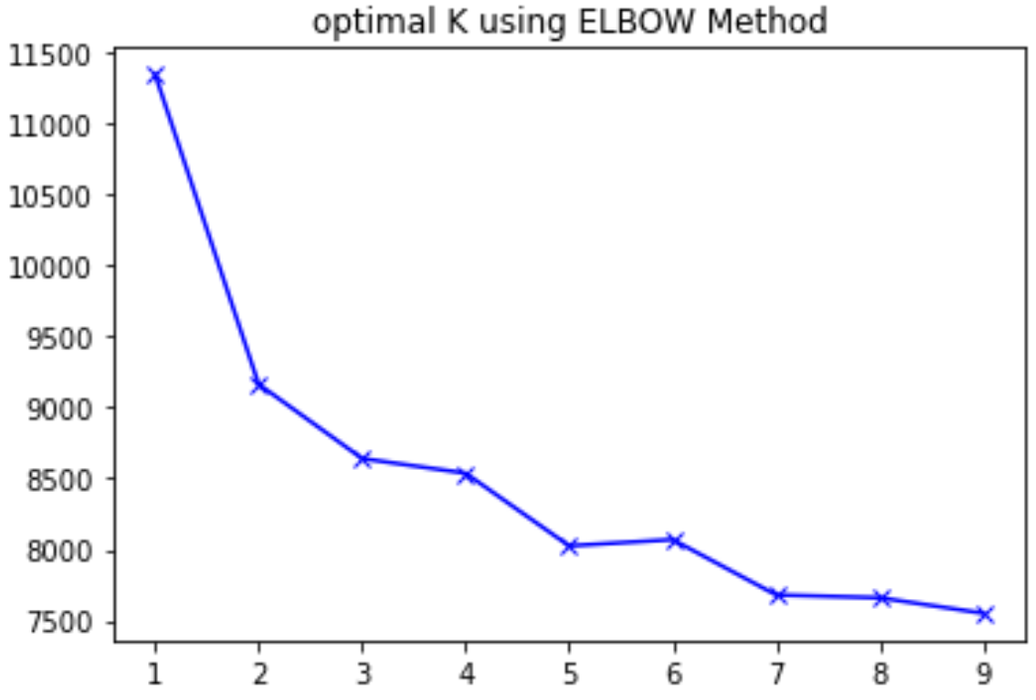
2. Methods

As I have previously said, I mainly used KModes and network analysis to analyze the data in my project. The reason why I chose cluster as my algorithm is that I was really interested on whether these people on the dating social media have cluster and their correlation with others.

As for the KModes, since nearly all my attributes are categorical data, using KMeans with dummy

variable is not a good way. Thus, I chose KModes, the algorithm that can deal perfectly with categorical data as my first algorithm. For KModes, I used the external packages KModes. The method I used was based on Huang's paper: Clustering large data sets with mixed numeric and categorical values. For feature space engineering, from previous paper, sexual orientation, gender, age, are effective factor analyzing the people on dating social media. Also, since people come from various places, I also used location as one of the variables. As for tuning the number of cluster of KModes, I basically used the Elbow Method to find out the optimal cluster for my KModes algorithm. As Elbow Method said, I plot the relation between cost and K and choose the K where it forms an elbow as my optimal number of clusters.

This is my Elbow Method plot.



We can see that there are basically 3 elbows, and I choose 2, and 5 as my number of cluster. Since a lot of my variables are binary variables, choosing 2 as my cluster number seems to be a good way.

As for network analysis, I used the same feature space as KModes to better form the comparison between the results. I used modularity maximization to classify agencies and assortativity to evaluate

the relation between different clusters. If the assortativity is positive, it means that agency in the same cluster is attractive to each other and vice versa.

As for training and test data splitting, I guess it is not required for my project. For KModes, the main objective is to cluster the data and spitting into train and test dataset will not make significant different. Since I did not do prediction work, the common indicator such as accuracy, or F1 sore do not apply to my method. As for network analysis part, I also personally think test data is not suitable for network analysis for my project. Since I used the interaction between people as my training set to analyze the relation between people. Randomly choosing data as my test data will destroy the human relationship in the original data. Therefore, I did not use test data to compare the modularity to the modularity in training data.

3. Experiments

3.1 Results

Below are the results from KModes.

The cluster center for K=2 is:

```
['female' 'Heterosexual' '26.3' 'A' 'Submisive' 'Nobody']
['male' 'Heterosexual' '38.1' 'A' 'Dominant' 'Women']]
```

The cluster center for K=5 is:

```
[['female' 'Heterosexual' '26.3' 'A' 'Submisive' 'Nobody']
['female' 'Heterosexual' '26.9' 'A' 'Submisive' 'Men']
['male' 'Heterosexual' '40.9' 'A' 'Dominant' 'Nobody']
['male' 'Heterosexual' '28.6' 'A' 'Dominant' 'Women']
 ['male' 'Heterosexual' '28.5' 'A' 'Switch' 'Women']]
```

The result for K=2 reveals the fact that most females are heterosexual. Most of them are nearly 26 years old. They are also submissive in the relationship and they are not so high standard in finding people. By saying that, they mostly write “Nobody” instead of “male” in person looking

for column. As for males, most of them are 38 years old. They basically tend to be more dominant in the relationship and they have clear objective: they prefer females. Interestingly, most people in the dataset are from A location. And heterosexuality takes the most part of people on the social media.

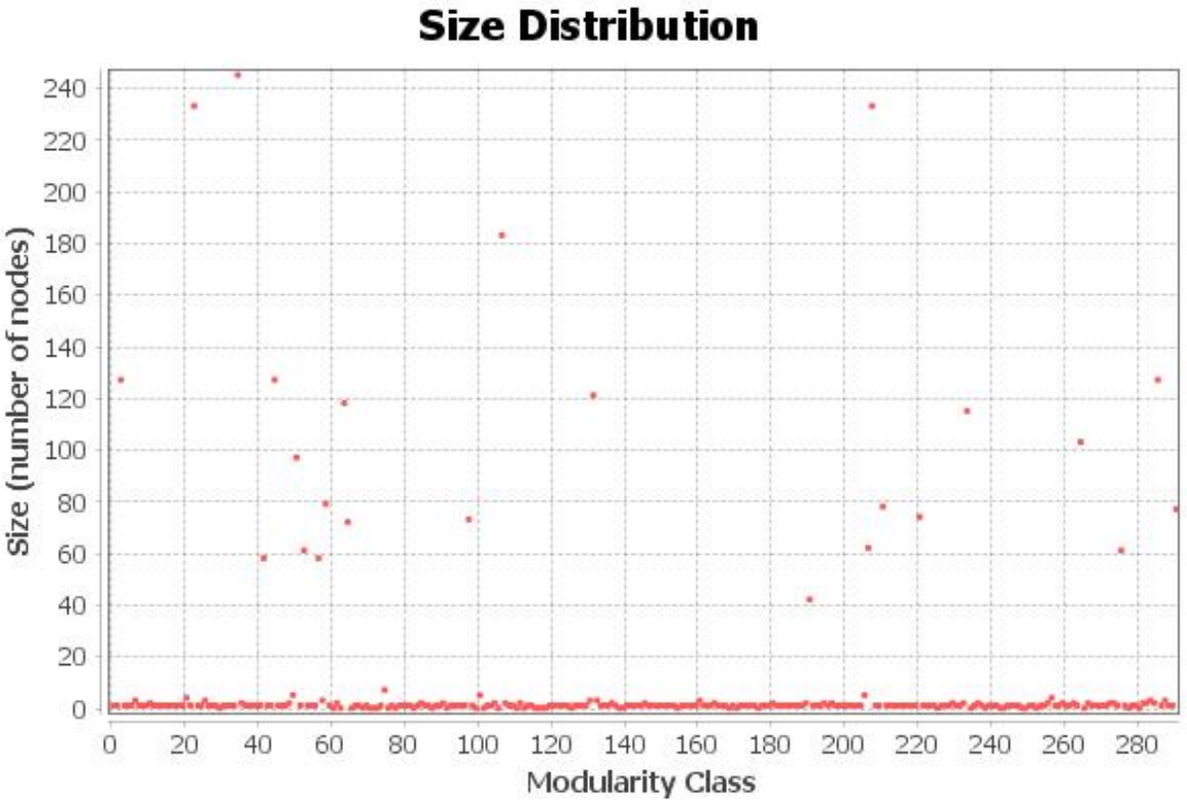
However, when I used 5 as the number of my cluster, the result for males changed. The result suggested that the age of men was various. There was a cluster of men aging 40, while other clusters had age around 28. This suggests that they are mainly two groups of males. One group is elder, the other is younger. However, for female, the age remained nearly the same as the result when K equaled to 2.

Below are the scores from network analysis. The result is the assortativity.

```
gender_score:-0.5733280916988478
location_score: 0.31954096231320117
age_score:0.0022663235077739706
```

We can see that even though people are clustering using gender, they do not like people with same gender. This is because the assortative is -0.5, which is significant different from 0. People tend to communicate with people in the same location because the assortativity is 0.3. Beisdes, age does not matter when people are dating online.

Below is the modularity maximization plot. We cannot detect any obvious community among people, this reflet the fact that they do not mind the attributes of others. As for the network plot, we can see that there is no obvious clustering. This suggests that basically people really do not consider the attributes of others on social media.



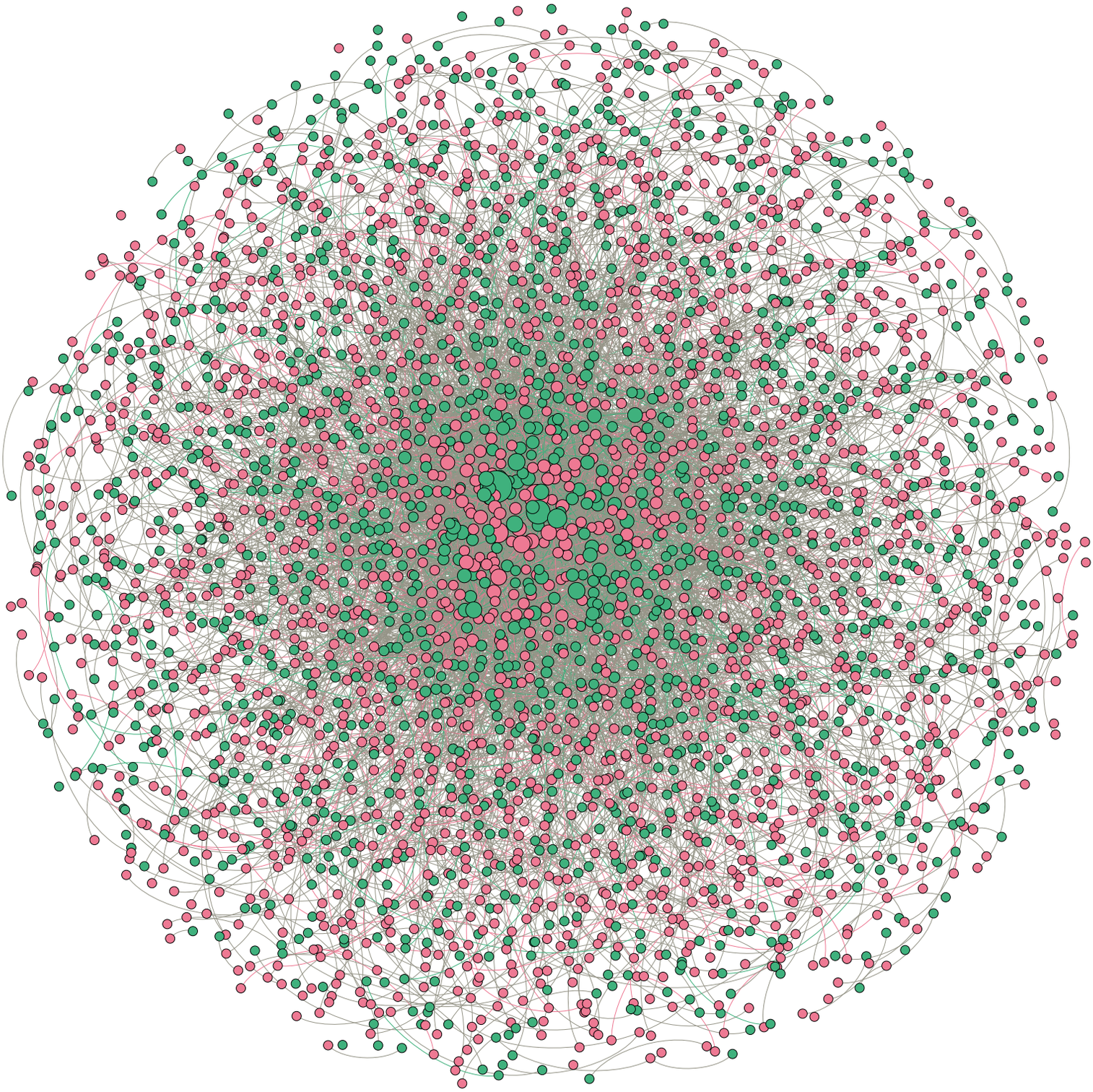
3.2 Hyperparameter Selection

As I have previously mentioned, I uses elbow method to choose the optimal K.

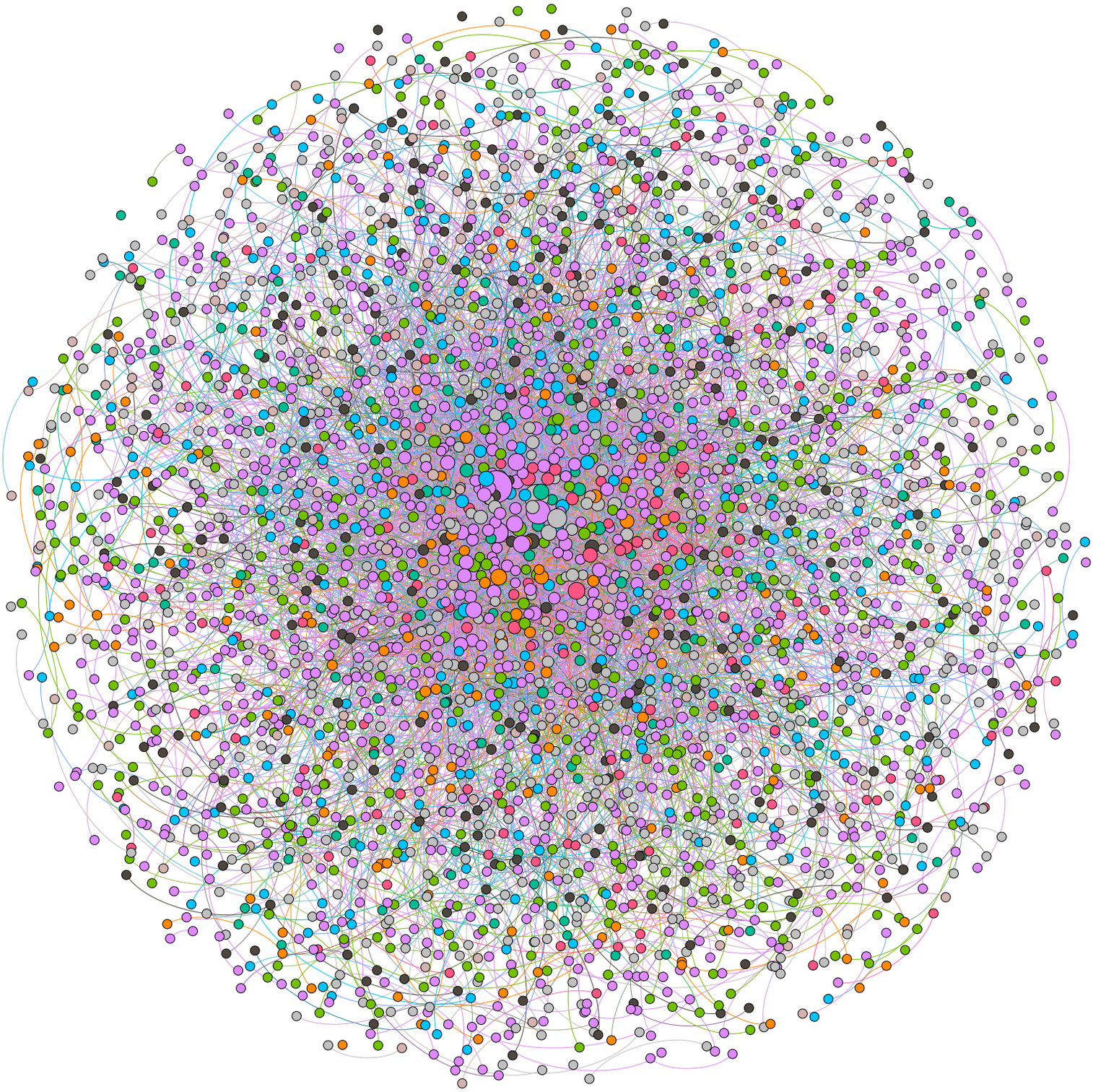
3.3 Visualization

I have used Dephi to visualize the network between users are the social media.
Below is the plot.

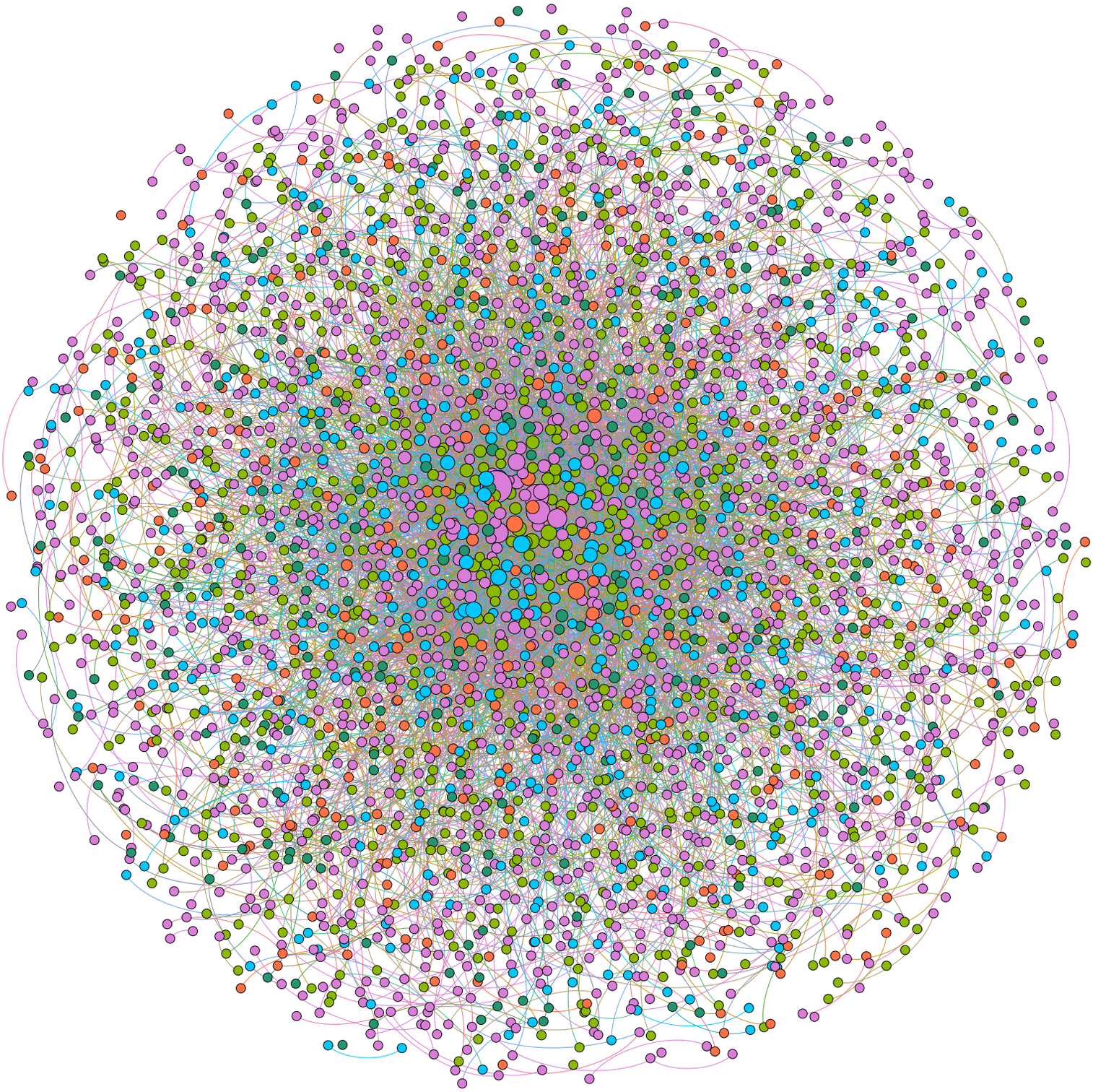
Gender



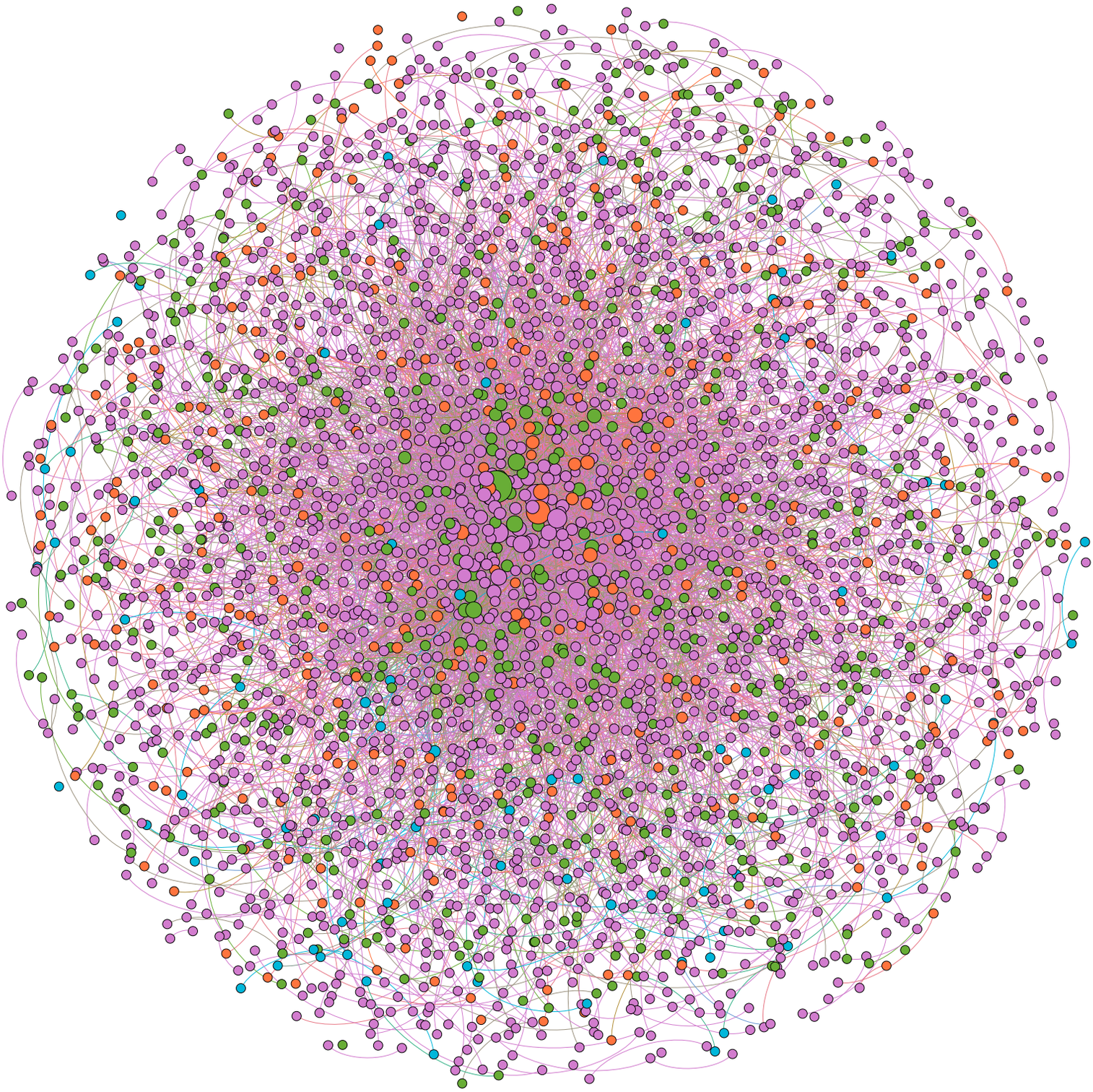
Location



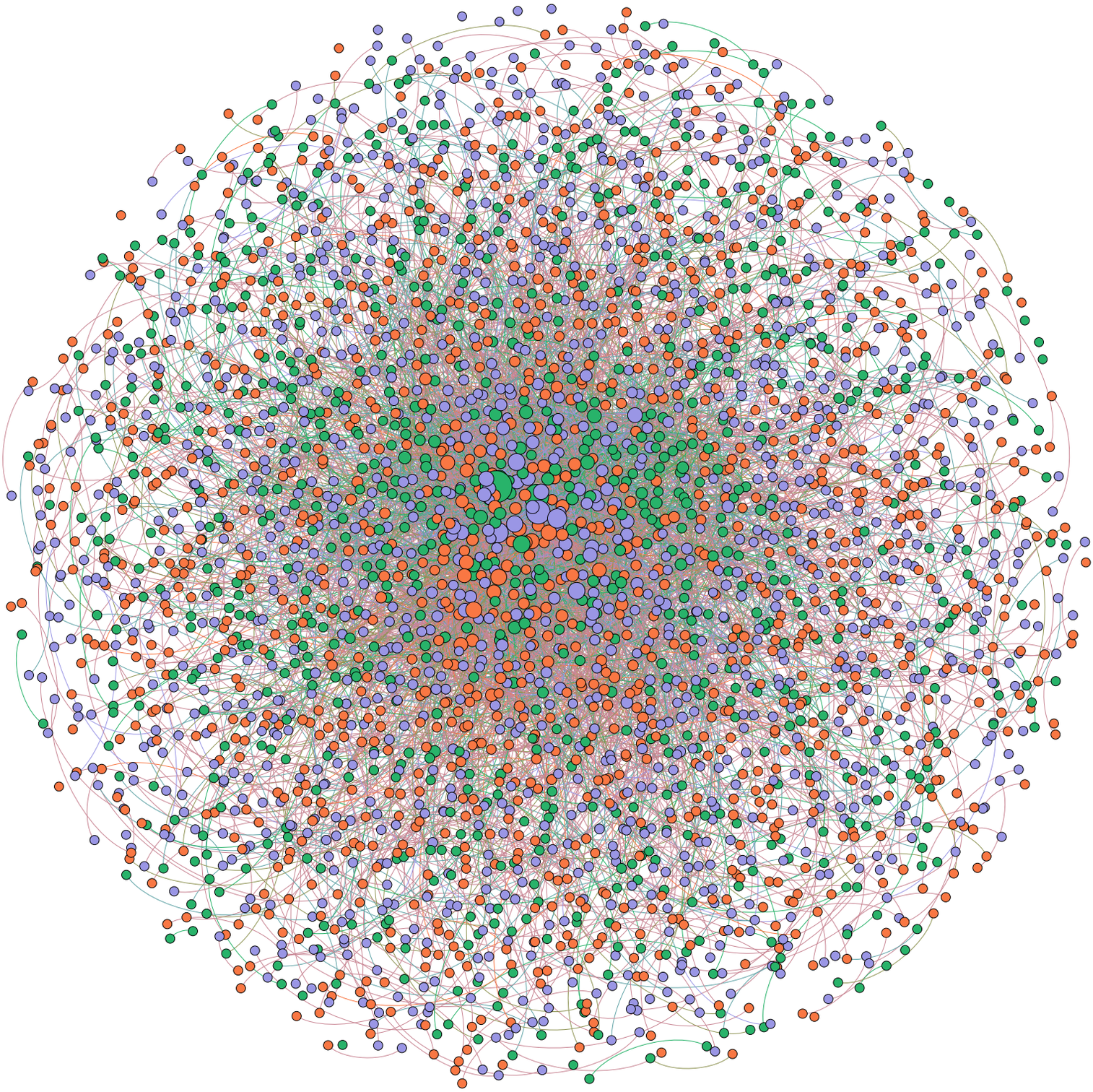
Looking for



Sexual Orientation



Community



4. Errors and Mistakes

One thing that really slowed me down is the data cleaning process. For instance, I had to check the class of each variable, replace the comma by point in variables such as Age. I also needed to delete exactly the agency without any connection with others. Overall, I spent a lot of time recognizing data, wrote code to clean it and debugged my code.

Another thing that was extremely difficult was to learn the new algorithm and Dephi to draw the plot. Since network analysis manipulates terms and evaluation criteria that I have never studied before, I took several days to study this new concept. Writing code totally on my own was also a hard time for me. It utilized brand new algorithm to compute the node and edge, also the modularity to evaluate the relation between same cluster. One bug that really stops me was that when I was computing G for network, I accidentally wiped out all the node and the final modularity remains 0 for every cluster. Actually, I spent several hours fixing this bug.

5. References

Covert online ethnography and machine learning for detecting individuals at risk of being drawn into online sex work.2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining(ASONAM),Barcelon, Spain,28-31, August

Roshansharma. (2020, July 30). Facebook Social Network Analysis. Retrieved November 19, 2020, from https://www.kaggle.com/roshansharma/facebook-social-network-analysis

Teejmahal20. (2020, January 21). Clustering Categorical Data: K-modes (CAT II). Retrieved November 19, 2020, from https://www.kaggle.com/teejmahal20/clustering-categorical-data-k-

modes-cat-ii

In [3]:
```
import pickle
import requests
import lxml
from bs4 import BeautifulSoup
from urllib.request import urlopen
import os
import pandas as pd
import numpy as np
import networkx as nx
from networkx.algorithms.community import greedy_modularity_communities
import community as community_louvain
```

In [4]:
```
os.chdir("C:/Users/tang/Desktop/Fal2020/STA671/Project")
import csv



sex=pd.read_csv('online_sex_work.csv')

sex=sex.dropna()   ####remove empty row
```

```
D:\360Downloads\Anconda\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (1,2,3,4,5,6,7,8,9,10,11,12,16,17) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [5]:
```
data=sex.dropna(subset=['Friends_ID_list'])      ####drop isolated nodes
data=data.drop_duplicates()
data=data.drop_duplicates(subset=['User_ID'])
data=data.reset_index(drop=True)
```

In [ ]:
```
########################k-mean analysis

data.index = data.index.astype(int)
data['Number_of_Comments_in_public_forum'] = data['Number_of_Comments_in_public_forum'].str.replace(' ', '').astype(int)
data['Number_of_advertisments_posted'] = data['Number_of_advertisments_posted'].astype(int)
data['Number_of_offline_meetings_attended'] = data['Number_of_offline_meetings_attended'].astype(int)
data['Profile_pictures'] = data['Profile_pictures'].astype(int)
data['Friends_ID_list'] = data['Friends_ID_list'].astype(str)
data['Risk'] = data['Risk'].astype(str)



data['Age'] = data['Age'].apply(lambda x: x.replace(',', '.'))
data['Age'] = data['Age'].replace('???', np.nan)
data['Age'] = data['Age'].astype(float)
data['Age'].fillna(data['Age'].mean(), inplace=True)
```

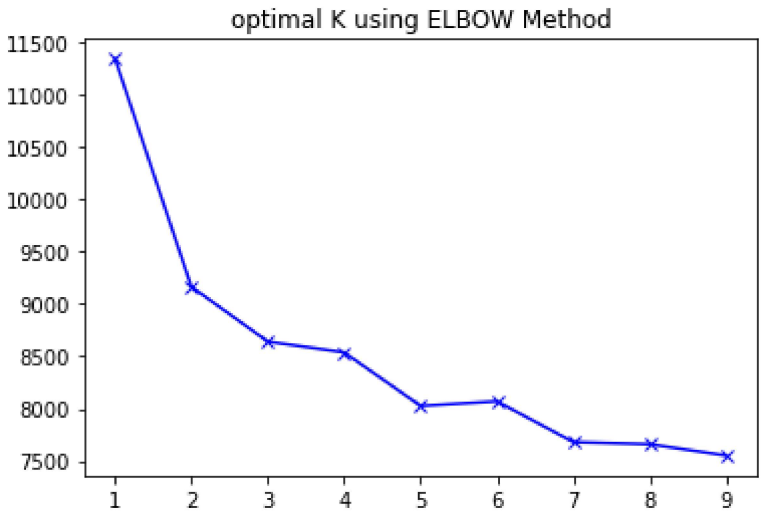
In [52]:
```
cost=[]
from matplotlib import pyplot as plt
K=range(1,10)
for num_clusters in list(K):
    km = KModes(n_clusters=num_clusters, init='Random', n_init=3, verbose=1)
    clusters = km.fit_predict(X)
    cost.append(km.cost_)
plt.plot(K,cost,'bx-')
plt.title("optimal K using ELBOW Method")
plt.show
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 0, cost: 11340.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 0, cost: 11340.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 0, cost: 11340.0
Best run was number 1
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 75, cost: 10333.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 706, cost: 9166.0
Run 2, iteration: 2/100, moves: 631, cost: 9166.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 798, cost: 9163.0
Run 3, iteration: 2/100, moves: 1, cost: 9163.0
Best run was number 3
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 594, cost: 8878.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 561, cost: 8640.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 586, cost: 8978.0
Best run was number 2
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 779, cost: 8935.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 717, cost: 8615.0
Run 2, iteration: 2/100, moves: 47, cost: 8615.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 578, cost: 8537.0
Run 3, iteration: 2/100, moves: 1, cost: 8537.0
Best run was number 3
Init: initializing centroids
```

```
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 525, cost: 8130.0
Run 1, iteration: 2/100, moves: 0, cost: 8130.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 765, cost: 8026.0
Run 2, iteration: 2/100, moves: 306, cost: 8026.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 1156, cost: 8346.0
Run 3, iteration: 2/100, moves: 553, cost: 8063.0
Run 3, iteration: 3/100, moves: 310, cost: 8063.0
Best run was number 2
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 458, cost: 8458.0
Run 1, iteration: 2/100, moves: 124, cost: 8458.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 534, cost: 8070.0
Run 2, iteration: 2/100, moves: 0, cost: 8070.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 774, cost: 8489.0
Run 3, iteration: 2/100, moves: 238, cost: 8484.0
Run 3, iteration: 3/100, moves: 0, cost: 8484.0
Best run was number 2
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 525, cost: 7681.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 739, cost: 7861.0
Run 2, iteration: 2/100, moves: 4, cost: 7861.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 403, cost: 8155.0
Run 3, iteration: 2/100, moves: 47, cost: 8155.0
Best run was number 1
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 187, cost: 7792.0
Run 1, iteration: 2/100, moves: 1, cost: 7792.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 981, cost: 7715.0
```

```
Run 2, iteration: 2/100, moves: 81, cost: 7715.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 1295, cost: 7693.0
Run 3, iteration: 2/100, moves: 78, cost: 7661.0
Run 3, iteration: 3/100, moves: 2, cost: 7661.0
Best run was number 3
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 477, cost: 7807.0
Run 1, iteration: 2/100, moves: 89, cost: 7805.0
Run 1, iteration: 3/100, moves: 3, cost: 7805.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 767, cost: 7621.0
Run 2, iteration: 2/100, moves: 106, cost: 7623.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 986, cost: 7555.0
Run 3, iteration: 2/100, moves: 548, cost: 7552.0
Run 3, iteration: 3/100, moves: 3, cost: 7552.0
Best run was number 3
```

Out[52]: <function matplotlib.pyplot.show(*args, **kw)>



In [49]:
```
X=data[['Gender','Sexual_orientation','Age','Location','Sexual_polarity','Looking_for']]
km = KModes(n_clusters=2, init='Huang', n_init=5, verbose=1)
clusters = km.fit_predict(X)
print(km.cluster_centroids_)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 555, cost: 9162.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 553, cost: 9164.0
Run 2, iteration: 2/100, moves: 129, cost: 9164.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 564, cost: 9163.0
Run 3, iteration: 2/100, moves: 0, cost: 9163.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 4, iteration: 1/100, moves: 475, cost: 9163.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 5, iteration: 1/100, moves: 604, cost: 9163.0
Best run was number 1
[['female' 'Heterosexual' '26.3' 'A' 'Submisive' 'Nobody']
 ['male' 'Heterosexual' '38.1' 'A' 'Dominant' 'Women']]
```

In [53]:
```
X=data[['Gender','Sexual_orientation','Age','Location','Sexual_polarity','Looking_for'
]]
km = KModes(n_clusters=5, init='Huang', n_init=5, verbose=1)
clusters = km.fit_predict(X)
print(km.cluster_centroids_)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 526, cost: 8300.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 938, cost: 8114.0
Run 2, iteration: 2/100, moves: 53, cost: 8114.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 1160, cost: 8213.0
Run 3, iteration: 2/100, moves: 6, cost: 8213.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 4, iteration: 1/100, moves: 1168, cost: 8432.0
Run 4, iteration: 2/100, moves: 574, cost: 8181.0
Run 4, iteration: 3/100, moves: 134, cost: 8126.0
Run 4, iteration: 4/100, moves: 11, cost: 8126.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 5, iteration: 1/100, moves: 903, cost: 8017.0
Best run was number 5
[['female' 'Heterosexual' '26.3' 'A' 'Submisive' 'Nobody']
 ['female' 'Heterosexual' '26.9' 'A' 'Submisive' 'Men']
 ['male' 'Heterosexual' '40.9' 'A' 'Dominant' 'Nobody']
 ['male' 'Heterosexual' '28.6' 'A' 'Dominant' 'Women']
 ['male' 'Heterosexual' '28.5' 'A' 'Switch' 'Women']]
```

In [7]:
```
member_id=data['User_ID']
nodelist=member_id.drop_duplicates()   ###create the list of node
nodelist=nodelist.tolist()     ###convert the series to list
nodelist=[int(i) for i in nodelist]  ###convert float to int

edgelist=[]


for i in range(len(data)):
    friend_ID=list(data['Friends_ID_list'][i].split(','))
    friend_ID=[int(i) for i in friend_ID]

    for j in friend_ID:
        edge_link=(int(data['User_ID'][i]),j)   ###create the edge
        edgelist.append(edge_link)    ###add the edge to the list


G=nx.Graph()
G.add_nodes_from(nodelist)
G.add_edges_from(edgelist)



node=list(G.nodes)

delete=list(list(set(node)-set(nodelist)) + list(set(nodelist)-set(node)))

G.remove_nodes_from(delete)    ####remove redundant nodes that are not appeared in the
 user_ID


edgelist=list(G.edges())   ### the edge and node after data cleaning
nodelist=list(G.nodes)
nodelist=[int(i) for i in nodelist]  ###convert float to int


edge=pd.DataFrame()
edge['Source']=[None for i in range(0,len(edgelist))]
edge['Target']=[None for i in range(0,len(edgelist))]
for i in range(len(edgelist)):
    edge['Source'][i]=edgelist[i][0]
    edge['Target'][i]=edgelist[i][1]

edge.to_csv('edgelist-luyi.csv')

nodelist=pd.DataFrame(nodelist)

nodelist.to_csv('nodelist.csv')
```

In [8]:
```
attribute_gender=dict(zip(list(data['User_ID']),list(data['Gender'])))  ###add the gend
er attribute to the node

attribute_age=dict(zip(list(data['User_ID']),list(data['Age'])))      ### add the age t
o the node

attribute_location=dict(zip(list(data['User_ID']),list(data['Location'])))    ### add t
he location to the node

nx.set_node_attributes(G, attribute_gender, "gender")

nx.set_node_attributes(G, attribute_age, "age")


nx.set_node_attributes(G, attribute_location, "location")

####test similarity of connections in the graph with respect to the given attribute
gender_sore=nx.attribute_assortativity_coefficient(G,'gender')
location_score=nx.attribute_assortativity_coefficient(G,'location')
age_score=nx.attribute_assortativity_coefficient(G,'age')



####test for community
c = list(greedy_modularity_communities(G))
```

In [9]:
```
print(gender_sore)
print(location_score)
print(age_score)
```

```
-0.5733280916988478
0.31954096231320117
0.002266323507773970 6
```