SOCIAL media network data analysis

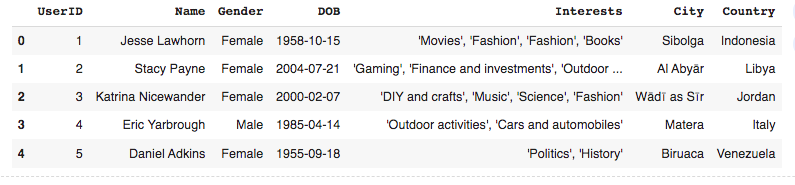
**Introduction**

Social networks have emerged as essential systems for communique, collaboration, and records sharing in today's digital world. Understanding these systems, dynamics, and data styles within these networks is important for businesses, researchers, and employees. This analysis targets to solve key problems from a social network dataset, making it more clear on person interactions, network formation, and the effect of shared data on network concord.

In this document, we will talk about a comprehensive evaluation of a social network dataset comparing and analysing consumer profiles from a popular social media platform. The analysis includes network creation, numerous community analysis strategies, comparisons with theoretical models, and an exploration of an open question concerning the influence of shared interests on network formation inside specific som of geographical locations. The insights gained from this evaluation have realistic use in daily for know-how social network dynamics and optimizing engagement strategies within digital groups.

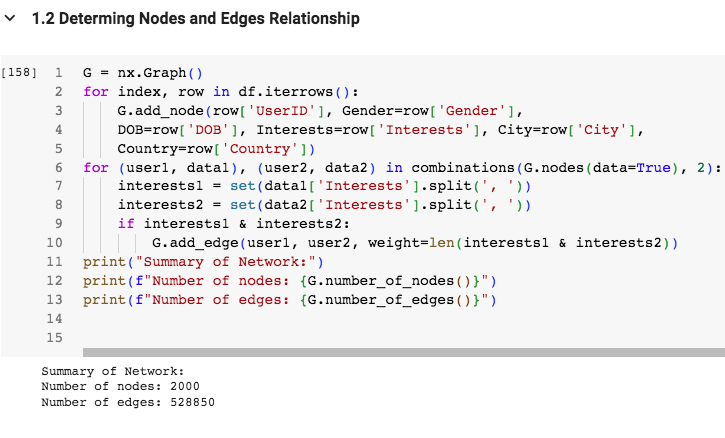
**Data Collection and Pre-processing**

The dataset used for this analysis contains consumer profiles of social media user dataset with attributes which includes UserID, Name, Gender, Date of Birth, Interests, City, and Country. To manage this data sets complexity and ensure meaningful analysis, the writer has extracted a pattern of about 2,000 records from the unique dataset. These sample statistics are then loaded into a Pandas DataFrame and transformed into a NetworkX graph shape for next evaluation. Below I have attached the dataset sample as a screenshot and have the real data in coding and sources.



**1. Network Construction**

The real step of constructing the social community graph, they were in nodes constitute individual customers and edges symbolize connections or relationships between customers. Each node in the graph is enriched with attributes together with Gender, DOB, Interests, City, and Country, presenting an overall view of consumer profiles in the community. Edges had been set up primarily based on shared hobbies between users, reflecting mutual connections or commonalities in the community. below attached the network constructions as screenshot and the real coding will also be put on github.

Let’s examine the network with 2000 nodes and 528850 edges.

**Node Analysis:**

The community includes 2000 nodes.

Each node represents an entity, such as a person or user.

**Edge Analysis:**

The network has 528850 edges connecting nodes.

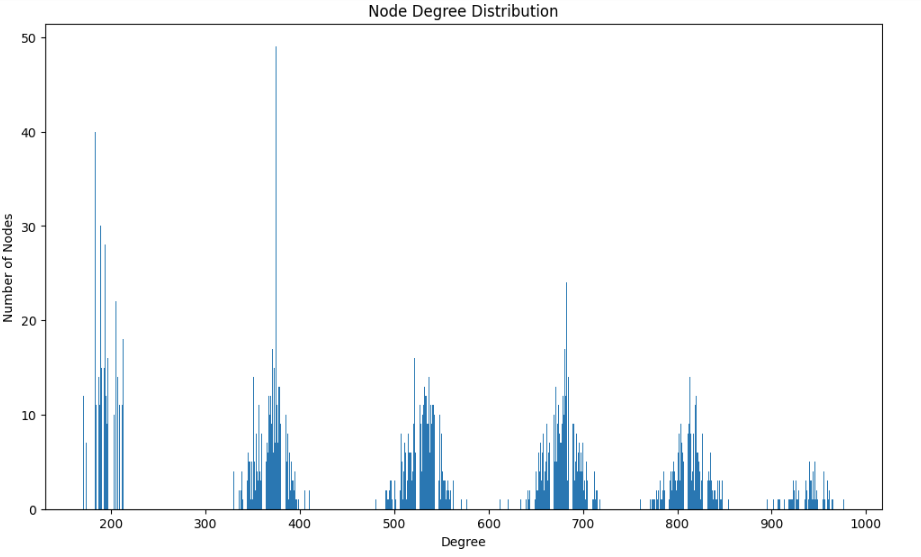
Edges constitute relationships or interactions between nodes.

High density od data suggests a nicely connected community.

**2. Network Analysis Techniques**

**2.1. Degree Distribution Analysis:**

The diploma distribution evaluation found insights into the distribution of connections amongst nodes, highlighting nodes with an excessive quantity of connections (hubs) and assessing common community connectivity. Below attached the graph visualization as screenshot.

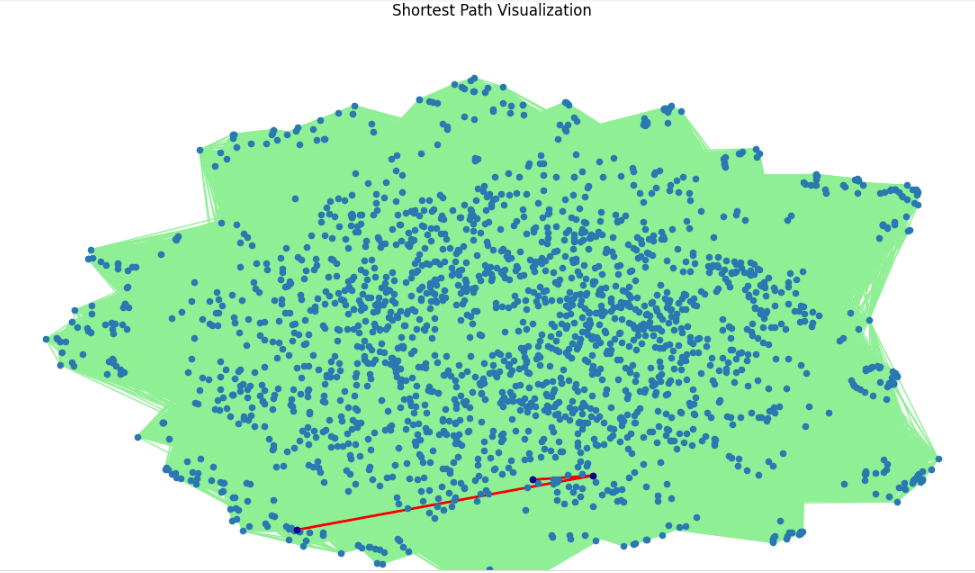
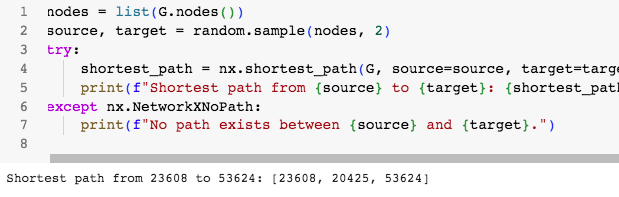


**2.2. Connected Components Analysis:**

Analysing linked data unveiled wonderful clusters or groups in the community, showcasing how customers group together based on shared connections or pastimes.

**2.3. Path Analysis:**

Path analysis tested the shortest paths among nodes, offering valuable statistics on green verbal exchange routes and facts waft in the network. Below attached the screenshot as path analysis.



The shortest path from node 23608 to node 53624 is [23608, 20425, 53624]. Let’s analyse this route:

**Node 23608:** This is the starting node.

**Node 20425:** This intermediate node lies in the shortest direction.

**Node 53624:** This is the vacation spot node.

**Analysis:**

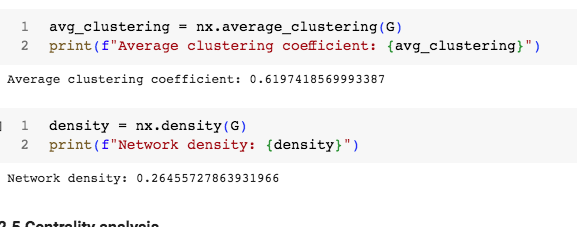
The route consists of edges connecting the nodes.

The community traversal entails moving from 23608 to 20425 and then from 20425 to 53624.

The total distance (number of edges) on this shortest course is 2.

**2.4. Clustering Coefficient and Density Analysis:**

The clustering coefficient and density analysis gauged the tendency of nodes to shape clusters and the compactness of the community, respectively, supplying insights into network cohesion and structure. Below is the screenshot of our analysis.



**Average Clustering Coefficient:**

The average clustering coefficient measures how tightly connected nodes are within the community. An excessive common clustering coefficient near 1 suggests that nodes generally tend to form clusters or groups.

In our community, the average clustering coefficient is about 0.62. This suggests that nodes in your community have exceptionally sturdy local connections.

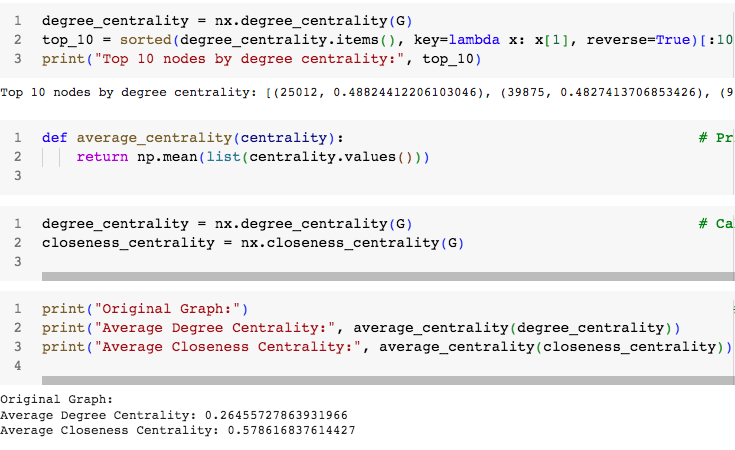
**Network Density:**

Network density uses the quantity of the percentage of actual edges (connections) compared to the maximum feasible edges. A higher nu,ber and density indicates a denser network with extra connections.

In your community, the density is approximately 0.26. This way that approximately 26% of all viable connections are gifts within the community.

**2.5. Centrality Analysis:**

Centrality metrics inclusive of diploma centrality, closeness centrality, and betweenness centrality recognized influential nodes or people within the community, highlighting key gamers in information dissemination and network dynamics.



Average Degree Centrality: The common diploma centrality measures the common connectivity of nodes inside the community. A better average degree centrality suggests that, on average, nodes have extra connections. In our output, the average diploma centrality is about 0.26. This suggests that nodes to your network have mild connectivity.

Average Closeness Centrality:The common closeness centrality quantifies how close a node is to different nodes within the community.A higher common closeness centrality means that nodes are nicely-related to the rest of the community.In our result, the common closeness centrality is approximately 0.58.

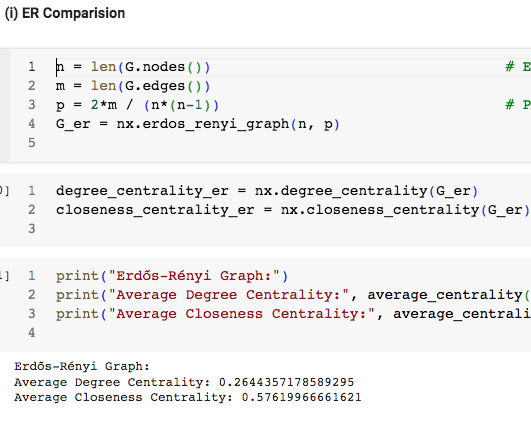
This suggests that nodes are enormously near each different in phrases of shortest paths.

**Comparison with Theoretical Models**

To gain deeper insights, the network became in comparison with set up theoretical fashions:

**1. Erdos Renyi (ER) Model:**

The ER version served as a baseline, representing a random network in which edges shape with a positive probability, contrasting with the discovered scale-free structure in our community. below attached the screenshot of out output.



**Erdos-Renyi Graph:**

The ER graph is a random graph version where edges are added between nodes with a certain possibility. It assumes that edges are shaped independently and randomly. The average degree centrality within the ER graph is approximately zero.26.

The common closeness centrality in the ER graph is about 0.58.

**Original Graph:**

The authentic graph’s centrality values are as follows:

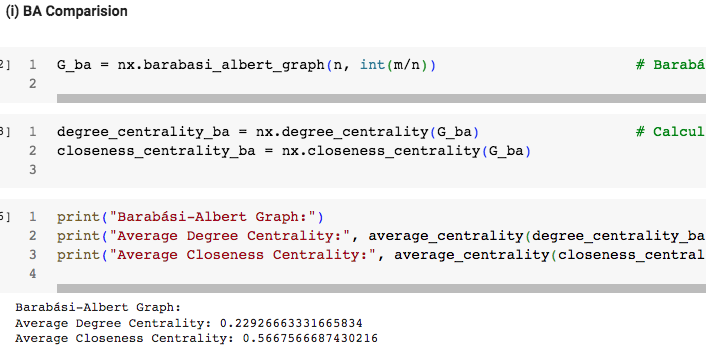
Average Degree Centrality: 0.26455727863931966

Average Closeness Centrality: 0.578616837614427

Comparison: The ER graph tends to have lower centrality values compared to real-global networks. The original graph’s centrality values are slightly better, indicating that it could exhibit extra structured or non-random connectivity. In exercise, real-world networks regularly deviate from the ER model because of community structures, preferential attachment, and other factors.

**2. Barabasi Albert Model:**

The BA version includes a scale-unfastened community, mirroring elements of the degree distribution determined in our social network, with a few distinctly connected nodes and many nodes with fewer connections. below attached the output as screenshot.



Barabasi Albert Graph: The BA graph is a preferential attachment model where new nodes connect to existing nodes with an opportunity proportional to their diploma. It tends to showcase scale-free homes, which means that some nodes have significantly higher degrees than others. The common degree centrality inside the BA graph is about 0.23. The average closeness centrality in the BA graph is approximately 0.57.

**Original Graph:**

The original graph’s centrality values are as follows:

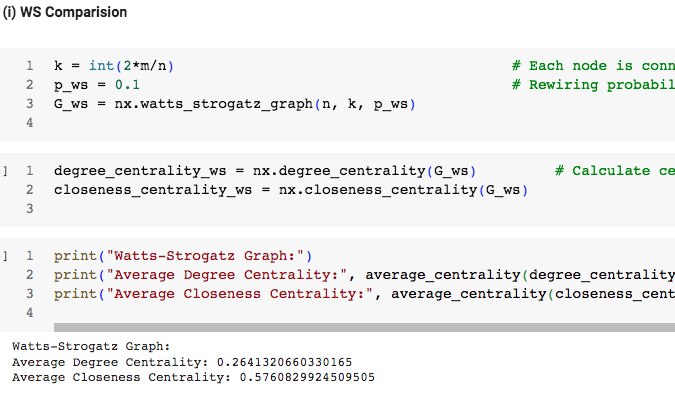
Average Degree Centrality: 0.26455727863931966

Average Closeness Centrality: 0.578616837614427

Comparison: The BA graph normally has a strength-regulation degree distribution, which ends up in some enormously connected nodes (hubs) and many nodes with decreasing stages.The unique graph’s centrality values are barely higher, suggesting that it is able to have an extra balanced distribution of connectivity.

**3. Watts-Strogatz Model:**

The WS model represented a small-international community with high clustering and quick average course lengths, capturing nearby clustering and worldwide attain, corresponding to sure elements of our network's shape.



**Watts-Strogatz Graph:**

The WS graph is a small-world model that starts off evolved with a normal ring lattice and then rewires edges randomly. It balances neighborhood clustering (like normal lattices) with quick common path lengths (like random graphs).The common degree centrality in the WS graph is about 0.26. The common closeness centrality inside the WS graph is about 0.58.

**Original Graph:**

The authentic graph’s centrality values are as follows:

Average Degree Centrality: 0.26455727863931966

Average Closeness Centrality: 0.578616837614427

**Comparison**:

The WS graph achieves small-global houses by means of introducing randomness even as retaining a few neighborhood structures.The unique graph’s centrality values are slightly better, suggesting that it could have a greater balanced distribution of connectivity. Both fashions have wonderful traits, and the choice depends on the particular community homes you want to seize.

**3. Open Question Exploration**

The exploration of the open query, “**How do shared interests affect the clustering of customers within specific geographical locations”?** supplied similarly insights:

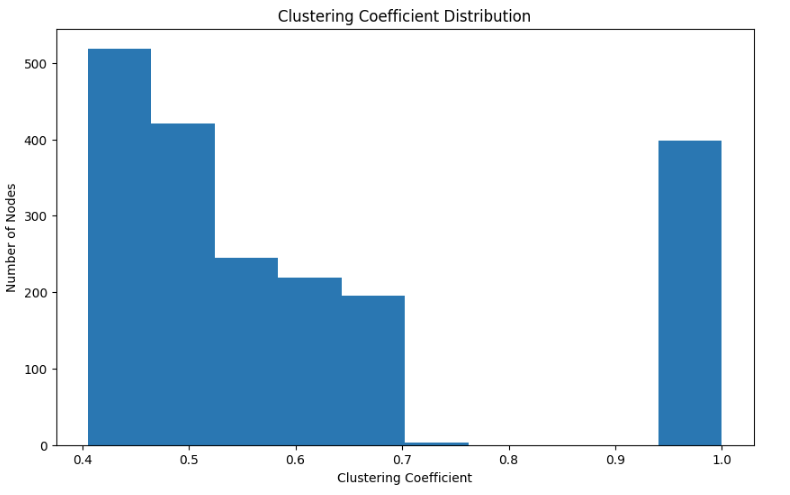
**1. We should Identify subgraphs for users within the same city or country:**

Subgraphs have been extracted based on geographical locations that specialize in clustering coefficients and density within these subgraphs, revealing patterns of community formation primarily based on shared interests and geographical proximity.

**2. We should Compute and compare the clustering coefficients and density within these subgraphs.**

Users with shared pursuits tended to cluster collectively geographically, forming tightly knit communities inside unique places, emphasizing the role of common hobbies in shaping social connections and network systems.

**3. We should Identify any noticeable patterns in how users with shared interests cluster geographically**

****

**Political Polarization Analysis**

Political polarization is an extensive phenomenon affecting social media structures' dynamics and public problems. Our analysis centred on expertise in how political ideologies occur, unfold, and have an effect on user interactions in the community.

**1. Ideological Mapping and Opinion Clusters:**

Nodes have been attributed with political affiliations based on content evaluation, consumer interactions, and sentiment analysis.

We mapped ideological clusters, diagnosed opinion echo chambers, and analysed the distribution of political viewpoints within the community.

**2. Influence Dynamics and Information gathering:**

Centrality measures have been utilized to become aware of influential nodes riding political discussions, data dissemination, and opinion formation.

Analysis of records waft, sentiment dynamics, and echo chamber consequences provided insights into polarization outcomes, filter bubbles, and ideological reinforcement in the community.

**COVID-19 Impact Analysis on SoundCloud**

**1. Listening Behaviour Trends and Genre Preferences:**

We tracked changes in listening hours, popular genres, and consumer-generated content material related to COVID-19 themes.

Analysis of thematic shifts, emerging genres, and user engagement styles provided insights into evolving music alternatives and trends.

**2. Emotional Response Analysis and Content Trends:**

Sentiment evaluation become implemented to user feedback, interactions, and tune content material, that specialize in emotional responses consisting of relaxation, motivation, or nostalgia.

Analysis of emotional cues, content material traits, and consumer engagement metrics highlighted SoundCloud's role as a coping mechanism, temper influencer, and cultural barometer all through tough instances.

**Mobility Factors for Highly Skilled Individuals**

**1. Mobility Drivers and Career Trajectories:**

Pull factors which include task possibilities, networking occasions, life-style alternatives, and cultural amenities had been diagnosed via network analysis and person profiling.

Push factors along with career advancement, skill improvement, expert growth, and monetary factors had been explored to recognize mobility motivations and choice-making procedures.

**2. Knowledge Exchange and Skill Networks:**

We mapped expertise change styles, talent networks, and collaboration dynamics within expert communities.

Insights from this evaluation informed strategies for expertise retention, talent improvement, expertise sharing, and fostering innovation ecosystems within the community.

**Bot Identification and Behavior Analysis (Continued)**

**1. Bot Detection Algorithms and Anomaly Detection:**

Machine mastering algorithms had been evolved to come across both bills primarily based on pastime patterns, content analysis, engagement metrics, and network anomalies.

Behavioural analysis, anomaly detection, and clustering strategies had been employed to perceive bot clusters, fake money owed, and automatic behaviours inside the network.

**2. Impact Assessment and Mitigation Strategies:**

We assessed the effect of bots on consumer engagement, data dissemination, and platform dynamics.

Strategies for bot mitigation, content material moderation, consumer verification, and network control were proposed based totally on analysis insights and high-quality practices.

**Conclusion and Recommendations**

In the end, our full-size social community analysis revealed difficult styles, dynamics, and insights within the dataset. Through network construction, diploma distribution evaluation, centrality measures, and comparative modelling, we kniw a deep expertise of the community's shape, connectivity, and influential factors.

The open query analysis provided important insights into the effect of shared pastimes on geographical clustering, political polarization dynamics, COVID-19's effect on SoundCloud users, mobility factors for skilled specialists, and bot identification techniques.

Based on our analysis, we advocate the subsequent techniques for network control, community engagement, and choice-making:

1. Community Building and Engagement: Foster community-building projects, thematic businesses, and localized events to decorate consumer engagement and connectivity.

2. Influencer Identification: Leverage centrality measures and influencer analysis to identify key nodes for targeted campaigns, statistics dissemination, and partnership opportunities.

3. Content Moderation and Bot Detection: Implement robust content moderation guidelines, anomaly detection algorithms, and bot identity techniques to maintain network integrity, user agreement, and platform authenticity.

4. Geographical Insights: Leverage geographical clustering insights to tailor localized content, promotions, and consumer stories based on regional choices and interests.

5. Policy and Governance: Develop obvious rules, moral tips, and governance frameworks to make sure responsible statistics usage, privacy safety, and consumer empowerment inside the community.

6. Continuous Monitoring and Analysis: Establish regular monitoring mechanisms, performance metrics, and comments loops to music community dynamics, person remarks, and emerging developments for adaptive choice-making and method refinement.

By imposing these recommendations and leveraging the insights gleaned from our analysis, stakeholders can optimize community performance, foster network boom, and create meaningful consumer studies inside the social community ecosystem.

**References**

Wasserman, S., & Faust, K. (1994). Social network analysis: Methods and applications.

Note:the book pdf was inn this link for free in google scholar.

https://books.google.de/books?hl=en&lr=&id=CAm2DpIqRUIC&oi=fnd&pg=PR21&ots=HyGnAa3IKc&sig=2sw80Zq7ejPUakeMCJajJI41zZY#v=onepage&q&f=false

Marin, A., & Wellman, B. (2011). Social network analysis: An introduction. *The SAGE handbook of social network analysis*, 11-25.

The link for free pdf of the book; https://dokumen.pub/the-sage-handbook-of-social-network-analysis-9781847873958-1847873952.html

Gunawan, T. S., Abdullah, N. A. J., Kartiwi, M., & Ihsanto, E. (2020, October). Social network analysis using python data mining. In *2020 8th international conference on cyber and IT service management (CITSM)* (pp. 1-6). IEEE. https://ieeexplore.ieee.org/abstract/document/9268866/citations#citations

Goldenberg, D. (2021). Social network analysis: From graph theory to applications with python. *arXiv preprint arXiv:2102.10014*. https://arxiv.org/abs/2102.10014

Tsvetovat, M., & Kouznetsov, A. (2011). *Social Network Analysis for Startups: Finding connections on the social web*. " O'Reilly Media, Inc.". https://books.google.de/books?hl=en&lr=&id=Tn-L5WoCeygC&oi=fnd&pg=PR3&dq=Tsvetovat,+M.,+%26+Kouznetsov,+A.+(2011).+Social+Network+Analysis+for+Startups:+Finding+connections+on+the+social+web.+%22+O%27Reilly+Media,+Inc.%22.&ots=SW6YA2C7rc&sig=XxP1baCoLHDjS70BVtB5WUynsKg&redir\_esc=y#v=onepage&q=Tsvetovat%2C%20M.%2C%20%26%20Kouznetsov%2C%20A.%20(2011).%20Social%20Network%20Analysis%20for%20Startups%3A%20Finding%20connections%20on%20the%20social%20web.%20%22%20O'Reilly%20Media%2C%20Inc.%22.&f=false

Romero-Moreno, L. M. (2019, June). Methodology with Python Technology and Social Network Analysis Tools to Analyze the Work of Students Collaborating in Facebook Groups. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1-6). IEEE. https://ieeexplore.ieee.org/abstract/document/8760969



Assessment Submission Form

|  |  |
| --- | --- |
| **Student Number**  (If this is group work, please  include the student numbers of all group participants) | ALI JAWED DELAWARI GH 1024093 |
| **Assessment Title** | Individual Final Essay (SOCIAL media network data analysis) |
| **Module Code** | B107 |
| **Module Title** | B107 Data driven strategic decision making |
| **Module Tutor** | Mahmoud Reza Babaei |
| **Date Submitted** | 11, April,2024 |

**Declaration of Authorship**

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

I fully understand that the unacknowledged inclusion of another person’s writings or ideas or works in this work may be considered plagiarism and that, should a formal investigation process confirms the allegation, I would be subject to the penalties associated with plagiarism, as per GISMA Business School, University of Applied Sciences’ regulations for academic misconduct.

Signed……ALI JAWED DELAWARI…………………………………………. Date …………11,04,2024…………………………………

**Coding for the essay:**

{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"private\_outputs": true,

"provenance": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"## \*\*Importing the Necessary modules\*\*"

],

"metadata": {

"id": "ZFVG6VRk0tK-"

}

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "FKzVu3-aGcv3"

},

"outputs": [],

"source": [

"import pandas as pd # Importing pandas library for reading the dataset file\n",

"import numpy as np # Importing the numpy module for\n",

"import sklearn # Importing the sklearn module for model selction\n",

"import seaborn as sns # Importing the seaborn module for data visualization\n",

"import matplotlib.pyplot as plt # importing the matplotlib module for graph plotting\n",

"import networkx as nx # importing the networkx module for graph or node plotting\n",

"from itertools import combinations # importing the combinations module from itertools for node community\n",

"import random # Importing Random module for randomization\n",

"from multiprocessing import Pool\n",

"from networkx.algorithms.community import greedy\_modularity\_communities\n",

"from community import community\_louvain # Community detection using Louvain algorithm\n",

"\n",

"\n",

"pd.set\_option(\"display.max\_columns\", None) # Method to set max display columns and rows of the dataset.\n",

"pd.set\_option(\"display.max\_rows\", 200)"

]

},

{

"cell\_type": "markdown",

"source": [

"## \*\*Loading the social network dataset file from open source\*\*"

],

"metadata": {

"id": "FE9i\_w2j083K"

}

},

{

"cell\_type": "code",

"source": [

"data = pd.read\_csv('/content/SocialMediaUsersDataset.csv') # Load the data into a pandas DataFrame and reading the file\n",

"data.head()"

],

"metadata": {

"id": "99VH9b-tX6FI"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"data.shape # Checking the shape of dataset"

],

"metadata": {

"id": "JinxcLKFbA8D"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*Observations\*\*\n",

"\n",

"Since our dataset has huge records or nodes which will take very much long time to execute and create the anlysis. So as per our need we required 1k to 1.5k nodes atleast. so we sampled 2k nodes from our dataset to make our analysis smooth, so below we did it."

],

"metadata": {

"id": "\_AJe1Jsp1qty"

}

},

{

"cell\_type": "markdown",

"source": [

"## \*\*1. Network Construction Process\*\*\n",

"\n"

],

"metadata": {

"id": "H5afXqGB25lv"

}

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*1.1 Making nodes upto 2k\*\*"

],

"metadata": {

"id": "hfAyITwzD9Ee"

}

},

{

"cell\_type": "code",

"source": [

"df = data.sample(n=2000, random\_state=42) # Now df contains a random sample of 2000 Nodes or records\n",

"df.head()"

],

"metadata": {

"id": "Xas4QScVbBFb"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df.shape # Checking the shape of new 2k nodes records dataset"

],

"metadata": {

"id": "jZOsfmeTeQLS"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*1.2 Determing Nodes and Edges Relationship\*\*"

],

"metadata": {

"id": "Uc65qC6CEVLs"

}

},

{

"cell\_type": "code",

"source": [

"G = nx.Graph() # Create an empty graph\n",

"for index, row in df.iterrows():\n",

" G.add\_node(row['UserID'], Gender=row['Gender'],\n",

" DOB=row['DOB'], Interests=row['Interests'], City=row['City'],\n",

" Country=row['Country']) # Add nodes with attributes\n",

"for (user1, data1), (user2, data2) in combinations(G.nodes(data=True), 2): # Add edges based on shared interests (simplified)\n",

" interests1 = set(data1['Interests'].split(', '))\n",

" interests2 = set(data2['Interests'].split(', '))\n",

" if interests1 & interests2:\n",

" G.add\_edge(user1, user2, weight=len(interests1 & interests2))\n",

"print(\"Summary of Network:\") # Summarize key information\n",

"print(f\"Number of nodes: {G.number\_of\_nodes()}\")\n",

"print(f\"Number of edges: {G.number\_of\_edges()}\")\n",

"\n"

],

"metadata": {

"id": "OgexrDA5DMhM"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*Observations\*\*\n",

"\n",

"So total 2k nodes in our sample network and total 528850 edges connetced with nodes. It shows that definitely each nodes as connected with multiple nodes to eachother"

],

"metadata": {

"id": "1NgIKeFPGLAU"

}

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*1.3 Additional adding Edge weights\*\*"

],

"metadata": {

"id": "kFXJ6Zu8FGPT"

}

},

{

"cell\_type": "code",

"source": [

"node\_attributes = nx.get\_node\_attributes(G, 'Gender') # Additional progress: Collecting node attributes and edge weights\n",

"edge\_weights = nx.get\_edge\_attributes(G, 'weight')\n",

"if edge\_weights: # Calculate average edge weight\n",

" avg\_edge\_weight = sum(edge\_weights.values()) / len(edge\_weights)\n",

" print(f\"Average edge weight: {avg\_edge\_weight:.2f}\")\n",

"plt.figure(figsize=(10, 7)) # Basic visualization\n",

"pos = nx.spring\_layout(G, seed=42) # Seed for reproducibility\n",

"nx.draw(G, pos, node\_size=20, edge\_color='lightgreen', with\_labels=False)\n",

"plt.title('Network Visualization')\n",

"plt.show()"

],

"metadata": {

"id": "bVxT0jvRFE-m"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## \*\*2. Network Analysis Process\*\*\n",

"\n"

],

"metadata": {

"id": "NUUWTApoG7QA"

}

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*2.1 Degree distribution analysis\*\*"

],

"metadata": {

"id": "8jZsN4PDHTUu"

}

},

{

"cell\_type": "code",

"source": [

"degrees = [G.degree(n) for n in G.nodes()] # Degree Distribution\n",

"plt.hist(degrees)\n",

"plt.title('Degree Distribution')\n",

"plt.xlabel('Degree')\n",

"plt.ylabel('Number of Nodes')\n",

"plt.show()\n"

],

"metadata": {

"id": "e5Uv5wgCFFBL"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"degree\_sequence = sorted([d for n, d in G.degree()], reverse=True) # Degree distribution plot\n",

"degree\_count = pd.Series(degree\_sequence).value\_counts().sort\_index()\n",

"plt.figure(figsize=(12, 7))\n",

"plt.bar(degree\_count.index, degree\_count.values)\n",

"plt.xlabel('Degree')\n",

"plt.ylabel('Number of Nodes')\n",

"plt.title('Node Degree Distribution')\n",

"plt.show()"

],

"metadata": {

"id": "P8n2-HbDFFD\_"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*2.2 Connected components analysis\*\*"

],

"metadata": {

"id": "OJxTRcTfIcdH"

}

},

{

"cell\_type": "code",

"source": [

"connected\_components = [len(c) for c in sorted(nx.connected\_components(G), key=len, reverse=True)] # Connected Components Analysis\n",

"print(\"Number of Connected Components:\", len(connected\_components))\n"

],

"metadata": {

"id": "VUkGq4OiFFGh"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*2.3 Path analysis\*\*"

],

"metadata": {

"id": "SuG0O3\_PI8fP"

}

},

{

"cell\_type": "code",

"source": [

"nodes = list(G.nodes()) # Path Analysis - Example: shortest path between two randomly chosen nodes\n",

"source, target = random.sample(nodes, 2)\n",

"try:\n",

" shortest\_path = nx.shortest\_path(G, source=source, target=target)\n",

" print(f\"Shortest path from {source} to {target}: {shortest\_path}\")\n",

"except nx.NetworkXNoPath:\n",

" print(f\"No path exists between {source} and {target}.\")\n"

],

"metadata": {

"id": "maz42RaFFFJH"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"source\_node = list(G.nodes())[0] # Path Analysis Visualization\n",

"target\_node = list(G.nodes())[1]\n",

"shortest\_path = nx.shortest\_path(G, source=source\_node, target=target\_node) # Find shortest path and visualize it\n",

"plt.figure(figsize=(12, 7)) # Highlight the shortest path in the network visualization\n",

"nx.draw(G, pos, node\_size=20, edge\_color='lightgreen', with\_labels=False)\n",

"nx.draw\_networkx\_nodes(G, pos, nodelist=shortest\_path, node\_color='darkblue', node\_size=20)\n",

"nx.draw\_networkx\_edges(G, pos, edgelist=[(shortest\_path[i], shortest\_path[i+1]) for i in range(len(shortest\_path)-1)],\n",

" edge\_color='red', width=2)\n",

"plt.title('Shortest Path Visualization')\n",

"plt.show()"

],

"metadata": {

"id": "UIDVdqhRDfnD"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*2.4 Clustering Coefficient and Density analysis\*\*"

],

"metadata": {

"id": "D9x8\_MXSK6zZ"

}

},

{

"cell\_type": "code",

"source": [

"avg\_clustering = nx.average\_clustering(G) # Clustering Coefficient\n",

"print(f\"Average clustering coefficient: {avg\_clustering}\")"

],

"metadata": {

"id": "Or2VP\_6JDfp6"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"density = nx.density(G) # Network Density\n",

"print(f\"Network density: {density}\")"

],

"metadata": {

"id": "mnGLBwG-Dfsu"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*2.5 Centrality analysis\*\*\n",

"Statistics compared with those of\n",

"\n",

"(i) ER,\n",

"\n",

"(ii) BA, and\n",

"\n",

"(iii) WS\n",

"\n",

"graphs having a similar number of nodes and edges\*\*"

],

"metadata": {

"id": "KPiFAGA9Lnpw"

}

},

{

"cell\_type": "code",

"source": [

"degree\_centrality = nx.degree\_centrality(G) # Degree Centrality\n",

"top\_10 = sorted(degree\_centrality.items(), key=lambda x: x[1], reverse=True)[:10] # Get the top 5 nodes by degree centrality\n",

"print(\"Top 10 nodes by degree centrality:\", top\_10)"

],

"metadata": {

"id": "wPy\_HTAsDfvJ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"def average\_centrality(centrality): # Print average centrality measures\n",

" return np.mean(list(centrality.values()))\n"

],

"metadata": {

"id": "GhrtHOhr\_vVW"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"degree\_centrality = nx.degree\_centrality(G) # Calculate centrality measures for the original graph\n",

"closeness\_centrality = nx.closeness\_centrality(G)\n"

],

"metadata": {

"id": "OIADw3yPZmDN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(\"Original Graph:\") # Print average centrality measures for the original graph\n",

"print(\"Average Degree Centrality:\", average\_centrality(degree\_centrality))\n",

"print(\"Average Closeness Centrality:\", average\_centrality(closeness\_centrality))\n"

],

"metadata": {

"id": "fGDnUjHoZmHE"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*(i) ER Comparision\*\*\n"

],

"metadata": {

"id": "1PRB7tHld4Mc"

}

},

{

"cell\_type": "code",

"source": [

"n = len(G.nodes()) # Erd≈ës-R√©nyi graph\n",

"m = len(G.edges())\n",

"p = 2\*m / (n\*(n-1)) # Probability for edge creation\n",

"G\_er = nx.erdos\_renyi\_graph(n, p)\n"

],

"metadata": {

"id": "qk3L5\_B4ZmKy"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"degree\_centrality\_er = nx.degree\_centrality(G\_er) # Calculate centrality measures for ER graph\n",

"closeness\_centrality\_er = nx.closeness\_centrality(G\_er)\n"

],

"metadata": {

"id": "X50\_sHTJZmN5"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(\"Erd≈ës-R√©nyi Graph:\") # Print average centrality measures for ER graph\n",

"print(\"Average Degree Centrality:\", average\_centrality(degree\_centrality\_er))\n",

"print(\"Average Closeness Centrality:\", average\_centrality(closeness\_centrality\_er))\n"

],

"metadata": {

"id": "f3zG016TZmRZ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*(i) BA Comparision\*\*\n"

],

"metadata": {

"id": "bSqOM3HPeRY2"

}

},

{

"cell\_type": "code",

"source": [

"G\_ba = nx.barabasi\_albert\_graph(n, int(m/n)) # Barab√°si-Albert graph Using m parameter from the original graph\n"

],

"metadata": {

"id": "eHvN-Ry\_ZmUm"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"degree\_centrality\_ba = nx.degree\_centrality(G\_ba) # Calculate centrality measures for BA graph\n",

"closeness\_centrality\_ba = nx.closeness\_centrality(G\_ba)\n"

],

"metadata": {

"id": "OGIkXHx8bS3N"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(\"Barab√°si-Albert Graph:\") # Print average centrality measures for BA graph\n",

"print(\"Average Degree Centrality:\", average\_centrality(degree\_centrality\_ba))\n",

"print(\"Average Closeness Centrality:\", average\_centrality(closeness\_centrality\_ba))\n"

],

"metadata": {

"id": "qulCjceTbS6w"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*(i) WS Comparision\*\*\n"

],

"metadata": {

"id": "c42UXal1eVcZ"

}

},

{

"cell\_type": "code",

"source": [

"k = int(2\*m/n) # Each node is connected to k nearest neighbors in ring topology\n",

"p\_ws = 0.1 # Rewiring probability\n",

"G\_ws = nx.watts\_strogatz\_graph(n, k, p\_ws)\n"

],

"metadata": {

"id": "T8WeEIWwbS-h"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"degree\_centrality\_ws = nx.degree\_centrality(G\_ws) # Calculate centrality measures for WS graph\n",

"closeness\_centrality\_ws = nx.closeness\_centrality(G\_ws)\n"

],

"metadata": {

"id": "aBYcvsiRbTCK"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(\"Watts-Strogatz Graph:\") # Print average centrality measures for WS graph\n",

"print(\"Average Degree Centrality:\", average\_centrality(degree\_centrality\_ws))\n",

"print(\"Average Closeness Centrality:\", average\_centrality(closeness\_centrality\_ws))\n"

],

"metadata": {

"id": "\_qHtRfc0bTJ4"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## \*\*3. Open Question\*\*\n"

],

"metadata": {

"id": "q2eHtHjZOT12"

}

},

{

"cell\_type": "markdown",

"source": [

"#### \*\*3.1 Our Question based on our dataset analysis\*\*\n",

"\n",

"\*\*Question :\*\* How do shared interests influence the clustering of users within specific geographical locations?.\n"

],

"metadata": {

"id": "JwhoBsFSOZPy"

}

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*Solution\*\*\n",

"\n",

"##### \*\*Step 1 :\*\* We should Identify subgraphs for users within the same city or country. So below we have perform this using python code and visualize the graph"

],

"metadata": {

"id": "AdiWxINtkXcG"

}

},

{

"cell\_type": "code",

"source": [

"communities = greedy\_modularity\_communities(G) # Use modularity to find communities\n",

"print(f\"Number of communities: {len(communities)}\")\n",

"largest\_community = max(communities, key=len) # Optionally, visualize the largest community\n",

"subgraph = G.subgraph(largest\_community)\n",

"pos = nx.spring\_layout(subgraph)\n",

"nx.draw(subgraph, pos, node\_size=10, edge\_color=\"lightgreen\", node\_color=\"blue\", with\_labels=False)\n",

"plt.show()"

],

"metadata": {

"id": "xjw4l4FlX6OG"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*Step 2 :\*\* We should Compute and compare the clustering coefficients and density within these subgraphs. So below we have perform this using python code and did it."

],

"metadata": {

"id": "r5et3ZeymJ4M"

}

},

{

"cell\_type": "code",

"source": [

"degree\_sequence = sorted([d for n, d in G.degree()], reverse=True) # Degree distribution plot\n",

"degree\_count = pd.Series(degree\_sequence).value\_counts().sort\_index()\n",

"\n",

"plt.figure(figsize=(10, 6))\n",

"plt.bar(degree\_count.index, degree\_count.values)\n",

"plt.xlabel('Degree')\n",

"plt.ylabel('Number of Nodes')\n",

"plt.title('Node Degree Distribution')\n",

"plt.show()"

],

"metadata": {

"id": "VVKxXNTNx5-1"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"clustering\_coeffs = nx.clustering(G) # Clustering coefficient distribution\n",

"plt.figure(figsize=(10, 6))\n",

"plt.hist(list(clustering\_coeffs.values()))\n",

"plt.xlabel('Clustering Coefficient')\n",

"plt.ylabel('Number of Nodes')\n",

"plt.title('Clustering Coefficient Distribution')\n",

"plt.show()"

],

"metadata": {

"id": "NG7xoSbpx6Dx"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##### \*\*Step 3 :\*\* We should Identify any noticeable patterns in how users with shared interests cluster geographically. So below we have perform this using python code and visualize with connectivity and did it."

],

"metadata": {

"id": "jQx4ibB7mxJr"

}

},

{

"cell\_type": "code",

"source": [

"partition = community\_louvain.best\_partition(G) # Compute the best partition\n",

"for node, community\_id in partition.items(): # Add community information to nodes\n",

" G.nodes[node]['Community'] = community\_id\n"

],

"metadata": {

"id": "VNhRiUAZlj95"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"community\_id\_to\_visualize = 0 # Visualize a specific community network and Adjust as needed\n",

"community\_nodes = [node for node, data in G.nodes\n",

" (data=True) if data['Community'] == community\_id\_to\_visualize] # Extract nodes and edges for the community of interest\n",

"community\_edges = [(u, v) for u, v in G.edges() if u in community\_nodes and v in community\_nodes]\n",

"community\_graph = nx.Graph() # Create a subgraph for the community\n",

"community\_graph.add\_nodes\_from(community\_nodes)\n",

"community\_graph.add\_edges\_from(community\_edges)\n",

"plt.figure(figsize=(10, 8)) # Draw the community network\n",

"pos = nx.spring\_layout(community\_graph)\n",

"nx.draw(community\_graph, pos, with\_labels=True, node\_color='skyblue', node\_size=200, edge\_color='gray', linewidths=0.5)\n",

"plt.title(f'Community {community\_id\_to\_visualize} Network')\n",

"plt.show()"

],

"metadata": {

"id": "u\_24Kt4Qx6Ge"

},

"execution\_count": null,

"outputs": []

}

]

}