

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Information Technology

By
KUNAL KUMAR
(40120120)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPIAAR NAGAR, RAJIV GANDHI SALAI CHENNAI – 600119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
KUNAL KUMAR (40120120) carried out the project entitled
"SOCIAL NETWORK ANALYSIS" under our supervision from
January 2023 to April 2023.


Internal Guide

Dr. S. Revathy, M.E., Ph.D.


Head of the Department

Dr. R. Subhashini, M.E., Ph.D

Submitted for Viva-voce Examination held on 29/04/2023


Internal Examiner


External Examiner

DECLARATION

I, KUNAL KUMAR (Reg. No – 40120120), hereby declare that the project report entitled "SOCIAL NETWORK ANALYSIS" done by me under the guidance of Dr. S. Revathy, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology.

DATE: 29/04/2023

PLACE: Chennai

A handwritten signature in black ink that reads "Kunal Kumar" with a stylized flourish at the end.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph. D**, Dean, School of Computing, **Dr. R. Subhashini, M.E., Ph.D.**, Heads of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide, **Dr. S. Revathy, M.E., Ph.D.** for her valuable guidance, suggestions, and constant encouragement that paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION



Elite

NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to

KUNAL KUMAR

for successfully completing the course

Advanced Graph Theory

with a consolidated score of **69** %

Online Assignments	16.67/25	Proctored Exam	52.5/75
--------------------	----------	----------------	---------

Total number of candidates certified in this course: **543**

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Jan-Mar 2023

(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur



Indian Institute of Technology Kanpur



Roll No: NPTEL23CS26S44342814

To validate the certificate



No. of credits recommended: 2 or 3

ABSTRACT

The project turns over Social Networks Analysis using Python. Firstly, we have been through the theoretical aspects of graphs. Then, we have applied the social networks analysis methods on different social networks. We have analyzed a Karate Club network and identified influencers and bridges in it. In addition, we have analyzed the members of karate club and predicted the linkage of members within themselves. Keywords: SNA (Social Networks Analysis), Graph Theory, ML (Machine Learning), Python Programming, Social network, detecting communities in social networks, detecting influencers in social networks, detecting bridges between communities in social networks, targeting classes in social networks, Degree, Density, Clustering, Clustering Coefficient, etc.

INDEX

S.No	TITLE	PAGE NO.
	Abstract	i
	List of Figures	ii
1	INTRODUCTION	1
	1.1 Programming Language Used	1
	1.2 Python	2
	1.3 IDE Used	2
	1.4 System Requirements	3
	1.5 Aim	4
	1.6 Scope	4
2	LITERATURE SURVEY	6
	2.1 Introduction	6
	2.2 Literature Review	7
3	METHODOLOGY	11
	3.1 System Architecture	11
	3.2 Network Definition And Vocabulary	13

	3.3 Popular Algorithm	18
	3.4 Working Procedure	19
4	RESULTS AND DISCUSSION	20
5	SUMMARY AND CONCLUSION	27
	5.1 Summary	27
	5.2 Conclusion	27
	5.3 Future Work	28
	APPENDIX	29
	SAMPLE CODE	29
	SCREENSHOTS	34

LIST OF FIGURES

S. No.	FIGURE	PAGE No.
3.1	SNA Architecture	11
3.2	Cluster of Nodes	14
3.3	Open and Closed Triads	14
3.4	Connectedness of a graph	15
4.1	Graphical Plotting of Groups	22
4.2	Degree Distribution Histogram	24
4.3	Local Clustering Coefficient Distribution	25
4.4	Graphical Plot of Communities	26

CHAPTER 1

INTRODUCTION

This is a project titled “Social Network Analysis”. From the above mentioned title it is very clear what our project is about and what problem it aims to solve.

A Social Network is the social structure which facilitates communication between a group of actors (individuals or organizations) that are related somehow (i.e. by common interests, shared values, financial exchanges, friendship, dislike, etc). For instance, your friends and you form a social network. But, social networks operate on many more levels, from family relations and disease spreading up to the level of company strategies, social movements or even nations. Furthermore, research in many scientific areas has shown that social networks are important when we study the way problems are solved, diseases are spreaded, organizations are run, and the degree to which individuals succeed in achieving their goals.

1.1 PROGRAMMING LANGUAGE USED

Social Network Analysis is a very useful mini-project which is implemented here in Python Programming Language which is one of the most popular and powerful programming languages in the world.

It dives into Python code with NetworkX constructing and implying social networks from real datasets.

1.2 PYTHON

- i. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- ii. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
- iii. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
- iv. Python supports modules and packages, which encourages program modularity and code reuse
- v. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

1.3 IDE USED

- i. Colaboratory, or “Colab” for short, is a product from Google Research
- ii. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

- iii. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.
- iv. Document your code that supports mathematical equations.
- v. Import/Save notebooks from/to Google Drive and Github
- vi. Integrate PyTorch, TensorFlow, Keras, OpenCV
- vii. Free Cloud service with free GPU

1.4 SYSTEM REQUIREMENTS

- i. Operating System: Windows XP/Vista/7/8/8.1/10
- ii. Memory (RAM): 2 GB of RAM required.
- iii. Hard Disk Space: 500 MB of free space required.
- iv. Processor: 1.0 GHz Intel Pentium processor or later.
- v. NET 4.5 Framework Required

1.5 AIM

This Software Requirements Specification provides a complete description of all the functions and specifications of the program “Social Networks Visualizer”. The project aims to find the set of people coming to a karate club who can be socially connected among themselves.

1.6 SCOPE

Social Networks Visualizer (SocNetV) is a flexible and user-friendly tool for the analysis and visualization of Social Networks. It lets you construct networks (mathematical graphs) with a few clicks on a virtual canvas or load networks of various formats (GraphViz, GraphML, Adjacency, Pajek, UCINET, etc) and modify them to suit your needs. SocNetV also offers a built-in web crawler, allowing you to automatically create networks from all links found in a given initial URL.

The application can compute basic network properties, such as density, diameter and distances (shortest path lengths), as well as more advanced structural statistics, such as node and network centralities (i.e. closeness, betweenness, graph), clustering coefficient, etc. Various layout algorithms (i.e. Spring-embedder, radial and layered according to node centralization) are supported for meaningful visualizations of your networks. Furthermore, random networks (Erdos-Renyi, Watts-Strogatz, ring lattice, etc) can be created with a few clicks. SocNetV is a work in progress and is being developed in C++ and Qt. Our primary target platform is Linux, but you can compile and run SocNetV on OS X and Windows as well, as long as you have Qt4 libraries installed.

Social Network Analysis is now involving because of the development of the technology. In fact, the world is getting more and more connected which increased the quantity of data and the volume of networks (Facebook, LinkedIn, WeChat,

YouTube, ...). We are in front of a huge puzzle to decrypt. Social network analysis can be applied in many fields, it can be applied in chemistry to analyze connections between atoms in the matter. SNA can be applied for marketing purposes. In addition, it can be applied in politics, psychology, sociology and economy.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Social Networks Visualizer (SocNetV) is a project to build a flexible and user-friendly, cross-platform tool for the analysis and visualization of social networks, targeting primarily the researcher. SocNetV lets you construct social networks with a few clicks on a virtual canvas or load networks of various formats (GraphML, GraphViz, Adjacency (Sociomatrix), Pajek, UCINET, etc.) and modify them to suit your needs. The application can compute all the basic network properties, such as graph diameter, and distances (shortest path lengths), as well as more advanced structural statistics, such as node and network centralities (i.e. Closeness, Betweenness, Graph, etc.), clustering coefficient, etc. Various layout algorithms (i.e. Energy-based, in circles and in levels according to various centrality indexes) are supported for meaningful visualizations of your networks. Furthermore, random networks (Erdos-Renyi, Watts-Strogatz, ring lattice, etc.) can be created with a few clicks. SocNetV is a work in progress and is being developed in C++ and Qt, an open-source GUI development toolkit from Nokia. Its primary target platform is Linux, but it can be compiled and run on OS X and Windows as well, as long as the Qt4 libraries are installed. The study of social networks is a new but quickly widening multidisciplinary area involving social, mathematical, statistical, and computer sciences (see Burt, Minor, & Associates, 1983, for application in diverse social environments; in the latter sciences.

2.2 LITERATURE REVIEW

2.2.1 *Networks and the relational perspective for studying behaviour*

A network is a set of nodes (vertices or entities) and a set of ties (edges or links) indicating connections or relations between the nodes. ¹ In a social network, the network nodes are typically people and the network ties are relations between people. However, social network analysis is also used to study the behaviour of organizations and groups. A visual representation of a social network is called a sociogram, often informally referred to as a network map.

There are two types of attributes of nodes: graph-theoretic attributes are derived from the network (e.g., number of connections, or 'degree'), while non-graphtheoretic attributes are intrinsic to the node (e.g., gender, race, age).

Discrete or categorical node attributes (e.g., race or gender) can be represented in sociograms by the colour or shape of the node, while numerical attributes (e.g., age, degree) can be represented by the size of the node.

There are several types of social networks. An egocentric network (also known as an egonet, or personal network) consists of a focal node/person ('ego') and the people he or she is connected to ('alters'). A 1.0 degree egonet doesn't show connections between alters, while a 1.5 degree egonet does show connections between alters. A 2.0 degree egonet also includes people who are 2 degrees of separation from ego i.e. friends of the alters who are not also friends of the ego. In contrast, a complete network consists of a set of nodes where all possible ties between the nodes are indicated.

The type of network can also be distinguished on the basis of mode. A unimodal network contains only one type of node or vertex, while a multimodal network contains more than one type of node. A bimodal network contains exactly two types of vertices, and an example of a bimodal network is an affiliation network consisting of people and the wiki articles they have edited. In this bimodal network,

people don't connect directly with people and wiki pages don't connect directly with wiki pages. However, a bimodal affiliation network can be transformed into two separate unimodal networks e.g., wiki editor-to-wiki editor and article-to-article.

Finally, multiplex networks have multiple types of edges. Indeed, most social networks are multiplex, that is, actors share more than one type of tie. For example, two people working in a firm may have a tie that describes their working relationship (e.g., person A reports to person B) but they might also be friends outside of work and hence have another type of tie that represents their affective friendship. With regards to online networks, there are also examples of multiplex networks. With Twitter we can have three types of directed edges:

following relationships, 'reply to' relationships, and 'mention' relationships. Often multiplex ties are reduced to a simplex tie e.g., a tie is deemed to exist if any of the multiplex ties exists.

While sociograms are useful for providing an intuitive overview of a social network, social network researchers have developed a wide range of network metrics that are used to quantitatively describe a given social network and the actors within the network.

2.2.2 Relational Social Science

Social network analysis (SNA) differs markedly from other social scientific approaches to studying human behaviour and outcomes. These differences are neatly captured in five fundamental and inter-related principles proposed by Wellman (1988), which together constitute the 'network perspective'.

First, the network perspective puts emphasis on the structure of relations, rather than the attributes of individual actors, in determining behaviour and outcomes. This can be seen clearly in the context of understanding the factors that contribute to labour market success, for example. The economic approach emphasizes human capital (e.g., experience, qualifications) as being important in determining

whether a person gets a job. In contrast, the network perspective regards the person's social network as the main driver of labour market success, since networks can provide opportunities (and also impose constraints) that impact on behaviour and outcomes. In particular, Granovetter (1973) emphasizes the importance of 'weak ties' (connections with people with whom you do not share any or many friends) as potential sources of innovative and useful information in the context of labour market outcomes.

Second, the network perspective focuses on pairs of actors (known as dyads) as the unit of analysis, rather than the actors themselves. A hallmark empirical technique used in approaches that do not take a network perspective is ordinary least squares (OLS) regression. In the labour market example above the human

capital model is tested by regressing log wages on explanatory variables measuring the human capital of each individual, and in such an approach the unit of analysis is the individual. In contrast, a social networks approach to studying labour market outcomes might employ an empirical technique such as exponential random graph modelling (ERGM) where the unit of analysis is the dyad (see, e.g., Daraganova and Pattison, 2013).

2.2.3 Examples and Tools

While almost all websites of user-generated content (UGC) are arguably online communities/networks of some sort, it is both informative and desirable for us to have a conceptual typology to delineate unique structural features for various online networks so that we can focus on 'networking' issues rather than content or functions of the websites. For that purpose, we employ a 2-by-2 scheme involving two dimensions of network ties – directionality and manifestation – to help organize the rich and fast growing body of studies on online networks.

Directionality of ties simply refers to the nature of the relations (i.e., directed versus undirected) between any pair of nodes. Manifestation of ties refers to the substantiality of the relations, with explicit ties formed by active acts (e.g.,

invitation, acceptance, reference, etc.) between the nodes whereas implicit ties created from inferences (e.g., co-occurrence or interactions). As shown in Table 10.1, the scheme

results in four distinct types of online networks, including i) explicitly undirected ties, ii) explicitly directed ties, iii) implicitly undirected ties, and iv) implicitly directed ties.

How much empirical research has each type of online network attracted? To gain a quick overview of the state of the art in online network research, we searched Web of Science (WoS) for studies that involve any of the 20 most popular social networking sites.³ A total of 4,260 relevant articles are found between 2004 and 2013. Four discernible patterns emerge from the studies. First, an overwhelming majority (70 percent) of the studies focused on two ‘newcomers’ of the top 20 list, i.e., Facebook and Twitter. Second, most (72 percent) of the studies appeared in the last three years

(2011–2013), with an increasingly accelerated growth rate (Figure 10.3). Third, more than 120 research areas (as defined by WoS) were involved, ranging from social sciences, business, and humanities to science, engineering and medicine, suggesting that online social networks have become a truly multidisciplinary field of studies. Four, areas of science and technology led the studies, as shown in the top 8 most active areas (each with a share of more than 5 percent) including computer science (43 percent), engineering (16 percent), education (10 percent), psychology (9 percent), information and library science (8 percent), business and economics (8 percent), communication (7 percent) and telecommunications (6 percent). In comparison, only 2 percent of the studies came from sociology (ranked the 13th), the traditional intellectual home of SNA.

CHAPTER 3

METHODOLOGY

3.1 SYSTEM ARCHITECTURE

The Social Network Analysis architecture consists of several interacting components. These are shown in the following figure. The various phases and modes of analytics are introduced afterwards.

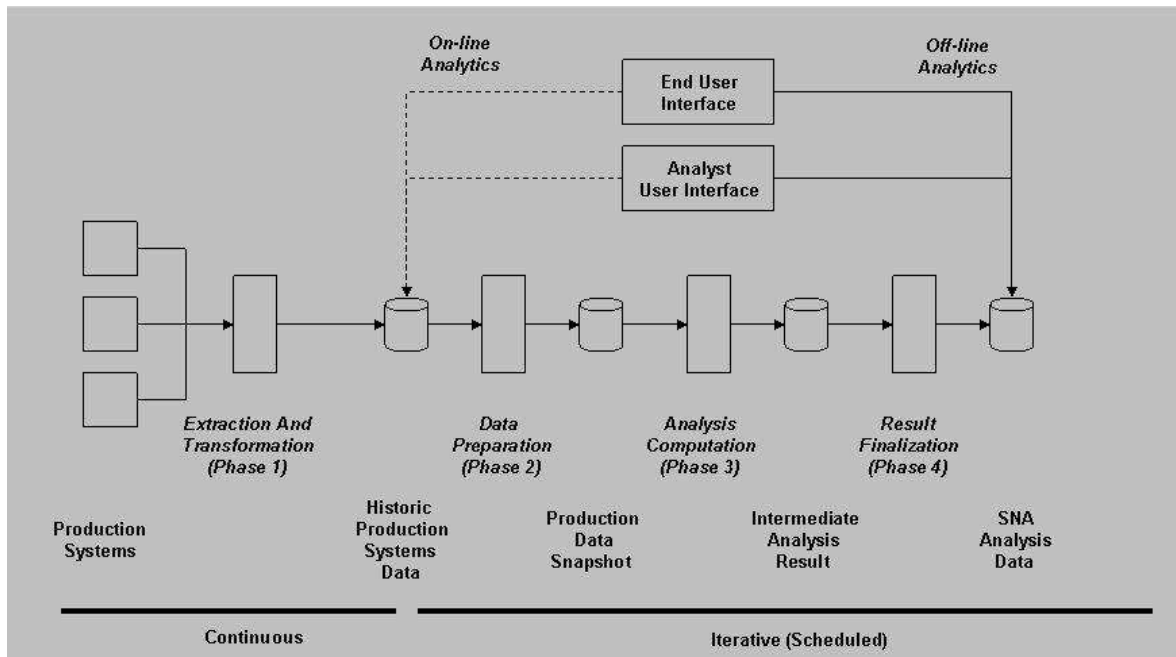


Fig. 3.1: SNA Architecture

The overall execution life cycle of Social Network Analysis (SNA) starts with collecting the data updates coming from production systems:

- i. **Phase 1: Data Collection** - This phase is an ongoing phase in the sense that the stream of data from the production systems is continuous

and the SNA system needs to ensure that all incoming data are reliably stored (independent of the arrival rate).

- ii. **Phase 2: Data Preparation** - This phase prepares a snapshot of the collected production data. The snapshot is the basis for a complete execution of the various analytics queries. Since all analytics queries are run on the same snapshot, the outcome of each of the analysis is consistent with respect to each other.
- iii. **Phase 3: Analysis Execution** - As soon as the snapshot is made available, the analysis starts and executes all analytics queries completely.
- iv. **Phase 4: Result Finalization** - After the analysis execution phase the results are made available to the user interface for end users and analysts to examine the analysis results.

Phase 2, 3 and 4 are iterative and are repeated many times. One possible strategy is to create snapshots by the data preparation phase on a fixed schedule (like once a day), or on a "as soon as possible" strategy. In principle, as soon as a snapshot is prepared, another one can be prepared in parallel, and consequently the analysis execution phase can be run in parallel, too. The schedule and degree of parallelization is determined by the needs of the company. However, the available space for data and the available computational power must be increased significantly in case of parallelism.

The final SNA Analysis Data produced by Phase 4 is being replaced with the result of each subsequent scheduled execution so that end users and analysts always operate on the latest data set.

3.2 NETWORK DEFINITION AND VOCABULARY

Network (or Graph): A representation of connections among a set of items.

- a. Items are called nodes (or vertices)
- b. Connections are called edges (or link or ties)
- c. Directed Graph: edges have direction
- d. Undirected Graph: edges have no direction

Weighted network: a network where edges are assigned a (typically numerical) weight. Edges can have many labels or attributes other than weights. Nodes can also have attributes.

Signed network: a network where edges are assigned positive or negative sign.

Multigraph: A network where multiple edges can connect the same nodes (parallel edges).

Bipartite Graph: a graph whose nodes can be split into two sets L and R and every edge connects a node in L with a node in R.

Clustering coefficients

Local clustering coefficient of a node

coefficient= # of pairs of C's friends who are friends / # of pairs of C's friends

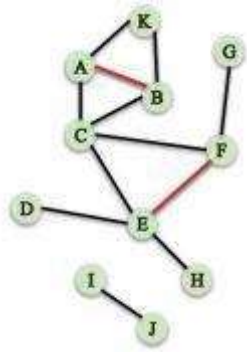


Fig. 3.2: Cluster of Nodes

Global clustering coefficient

First approach is to calculate the average of all the local clustering coefficients of all the nodes.

Second approach is to calculate the transitivity.

Transitivity: Ratio of number of triangles and number of “open triads” in a network.

Transitivity = $3 * \text{Number of closed triads} / \text{Number of open triads}$

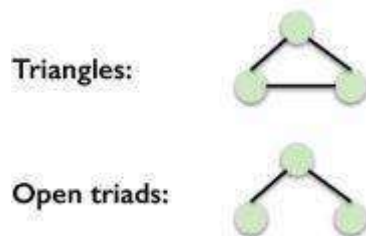


Fig. 3.3: Open and Closed Triads

Distances in the graph

Distance between two nodes: length of the shortest path between them.

Eccentricity of a node n is the largest distance between n and all other nodes.

Average distance between every pair of nodes.

Diameter: maximum distance between any pair of nodes,

Radius: the minimum eccentricity in the graph.

Identifying central and peripheral nodes:

The Periphery is the set of nodes with eccentricity = diameter. The center is the set of nodes with eccentricity = radius.

Connectedness of the graph

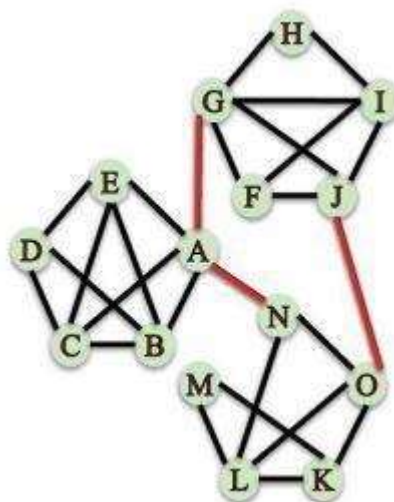


Fig. 3.4: Connectedness of a graph

An undirected graph is connected if, for every pair node, there is a path between them.

However, if we remove edges A—G, A—N, and J—O, the graph becomes disconnected.

A directed graph is weakly connected if replacing all directed edges with undirected edges produces a connected undirected graph.

A directed graph is Strongly connected if for every pair node, there is a directed path between them.

Network robustness: the ability of a network to maintain its general structural properties when it faces failures or attacks.

Node connectivity: Minimum number of nodes needed to disconnect a graph or pair of nodes.

Edge connectivity: Minimum number of edges needed to disconnect a graph or pair of nodes.

Node Importance

Degree Centrality

We assume that important nodes have many connections. The most basic measure of centrality: number of neighbors. Undirected networks: use degree.

Directed networks: use in-degree or out-degree

$$C_{deg}(v) = \frac{d_v}{|N| - 1} \quad \text{—— eqn (i)}$$

Where N number of nodes and d_v is the degree of the node v

Closeness centrality

We assume that important nodes are close to other nodes.

$$C_{close}(v) = \frac{|N| - 1}{\sum_{u \in N \setminus \{v\}} d(v, u)} \quad \text{—— eqn (ii)}$$

Where $\mathbf{v}$ is the set of nodes in the network, and $d(v,u)$ =length of shortest path from $\mathbf{u}$ to $\mathbf{v}$.

Betweenness centrality

We assume that important nodes connect other nodes.

$$C_{btw}(v) = \sum_{s,t \in N} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \quad \text{—— eqn (iii)}$$

$\sigma_{s,t}(v)$ is the number of shortest paths between nodes s and t .

$\sigma_{s,t}$ is the number shortest paths between nodes s and t that pass through central node.

3.3 POPULAR ALGORITHM

PageRank

Developed by Google founders to measure the importance of webpages from the hyperlink network structure. PageRank assigns a score of importance to each node.

Important nodes are those with many in-links from important pages. It can be used for any type of network, but it is mainly useful for directed networks.

The Algorithm:

1. Assign all nodes a PageRank of 1/
2. Perform the Basic PageRank Update Rule k times.

Basic PageRank Update Rule: Each node gives an equal share of its current PageRank to all the nodes it links to. The new PageRank of each node is the sum of all the PageRank it received from other nodes.

HITS Algorithm

Computing iterations of the HITS algorithm to assign an authority score and hub score to each node.

1. Assign each node an authority and hub score of 1.
2. Apply the Authority Update Rule: each node's authority score is the sum of hub scores of each node that points to it.
3. Apply the Hub Update Rule: each node's hub score is the sum of authority scores of each node that it points to.
4. Normalize Authority and Hub scores

5. Repeat k times.

3.4 WORKING PROCEDURE

The network shows the relationship between 34 people in a karate club. The link or edges are people who know each other outside of the club. Each node is a person. There is a dispute between two members of the club, John A represented by Node 33 and Mr. Higgins represented by Node 0. The fight led to division of Karate club into two groups. Each member's choice of which side to join would depend on how they get along with the other people in the club. Network analysis will give us power to predict which group each person will join. We will firstly import sys module. sys module in python provides various functions and variables that are used to manipulate different parts of python runtime environment. It allows operating on the interpreter as provides access to the variables and functions that interact strongly with the interpreter. After that import the networkx as nx. networkx is a python package for the creation, manipulation and study of the structure, dynamics and functions of complex networks. After that import matplotlib.pyplot as plt. It is an API for python matplotlib that effectively make it viable open source alternative to matlab. Matplotlib is a library for gated visualisation typically in the form of plots. Import ZKC_graph already defined in networkx package. Specify nodes of two group leaders. Create circular position using circular_layout(). Then draw the network. We have used node colour as green for Mr_Hi's group and red for John A's group. Now we will find density of the network which is comparison between existing number of edges and maximum possible number of edges. Density 1 means every possible edge exist while density 0 means no connections between nodes. Here we will calculate the degree of the network. Node with maximum degree or with maximum centrality has a high chance of playing the central role. Now we will find the clustering coefficient. Cluster is like a group inside the network who are more tightly connected. Now we will check communities by firstly importing the required packages. The output shows 3 which means there are 3 communities here.

CHAPTER 4

RESULTS AND DISCUSSION

This project aims to detect community in a network using python. We will do network analysis in Python using the networkx package, which is one of the most popular package for network analysis.

The network shows the relationship between 34 people in a karate club. The link or edges are people who know each other outside of the club. Each node is a person.

There is a dispute between two members of the club, John A represented by Node 33 and Mr.Higgins represented by Node 0. The fight led to division of Karate club into two groups. Each member's choice of which side to join would depend on how they get along with the other people in the club. Network analysis will give us power to predict which group each person will join.

```
[3] import sys
import networkx as nx
import matplotlib.pyplot as plt
```

We will firstly import sys module.sys module in python provides various functions and variables that are used to manipulate different parts of python runtime environment.It allows operating on the interpreter as provides access to the variables and functions that interact strongly with the interpreter.

After that import the networkx as nx. networkx is a python package for the creation, manipulation and study of the structure, dynamics and functions of complex networks.

After that import matplotlib.pyplot as plt. It is an API for python matplotlib that effectively make it viable open source alternative to matlab. Matplotlib is a library for gated visualisation typically in the form of plots.

```
[4] ZKC_graph = nx.karate_club_graph()
    Mr_Hi=0
    John_A=33
    club_labels=nx.get_node_attributes(ZKC_graph,'club')
    print({key:club_labels[key] for key in range(10,16)})
```

Import ZKC_graph already defined in networkx package. Specify nodes of two group leaders.

```
[5] circ_pos=nx.circular_layout(ZKC_graph)
    nx.draw(ZKC_graph,circ_pos)
    nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=[Mr_Hi],node_color='g',alpha=1)
    nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=[John_A],node_color='r',alpha=1)
```

Create circular position using circular_layout(). Then draw the network. We have used node colour as green for Mr_Hi's group and red for John A's group.

<matplotlib.collections.PathCollection at 0x7fdee9cda070>

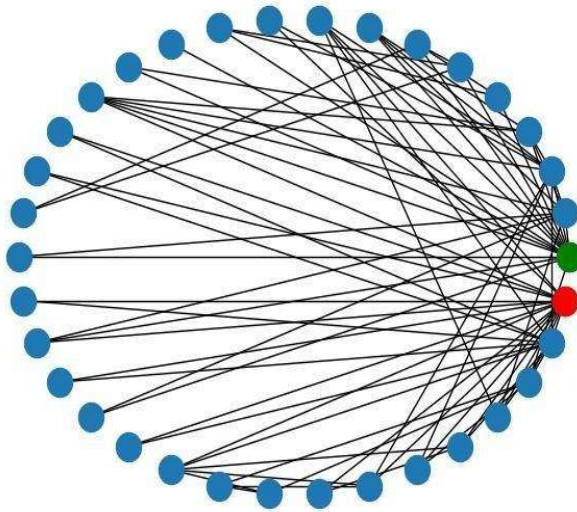


Fig 4.1 Graphical Plotting of Groups

```
[6] density=nx.density(ZKC_graph)
    print("The edge density is: "+str(density))
```

```
The edge density is: 0.13903743315508021
```

Now we will find density of the network which is comparison between existing number of edges and maximum possible number of edges. Density 1 means every possible edge exist while density 0 means no connections between nodes.

```
[7] degree=ZKC_graph.degree()
    degree_list=[]

    for (n,d) in degree:
        degree_list.append(d)

    av_degree=sum(degree_list)/len(degree_list)
    print("The average degree is "+str(av_degree))

The average degree is 4.588235294117647
```

Here we will calculate the degree of the network. Node with maximum degree or with maximum centrality has a high chance of playing the central role.

```
[8] plt.hist(degree_list,label='Degree Distribution')
    plt.axvline(av_degree,color='r',linestyle='dashed',label='Average Degree')
    plt.legend()
    plt.ylabel('Number of Nodes')
    plt.title('Karate Club: Node Degree')
```

Plot the degree distribution histogram.

```
Text(0.5, 1.0, 'Karate Club: Node Degree')
```

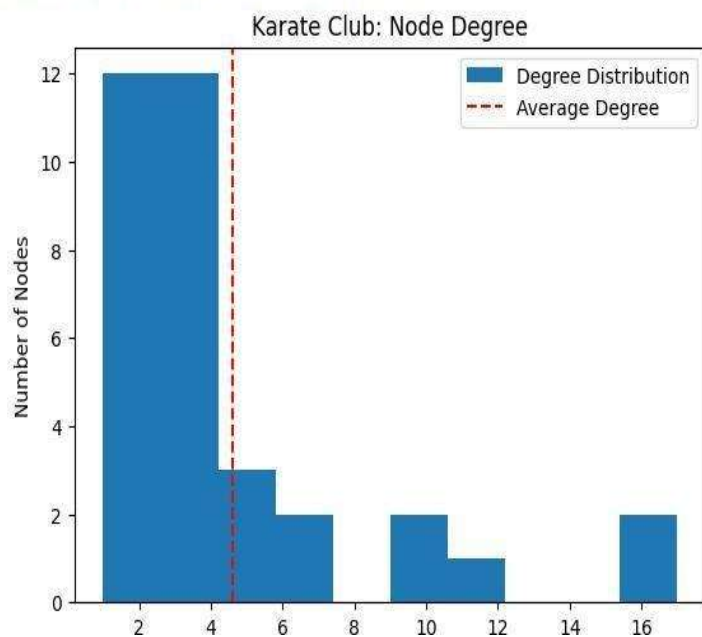


Fig 4.2 Degree Distribution Histogram

```
[9] local_clustering_coefficient=nx.algorithms.cluster.clustering(ZKC_graph)
    av_local_clustering_coefficient=sum(local_clustering_coefficient.values())/len(local_clustering_coefficient)

    plt.hist(local_clustering_coefficient.values(),label='Local Clustering Coefficient Distribution')
    plt.axvline(av_local_clustering_coefficient,color='r',linestyle='dashed',label='Average Local Clustering Coefficient')
    plt.legend()
    plt.ylabel('Number of Nodes')
    plt.title('Local Clustering Coefficient')
    plt.show()
```

Now we will find the clustering coefficient. Cluster is like a group inside the network who are more tightly connected.

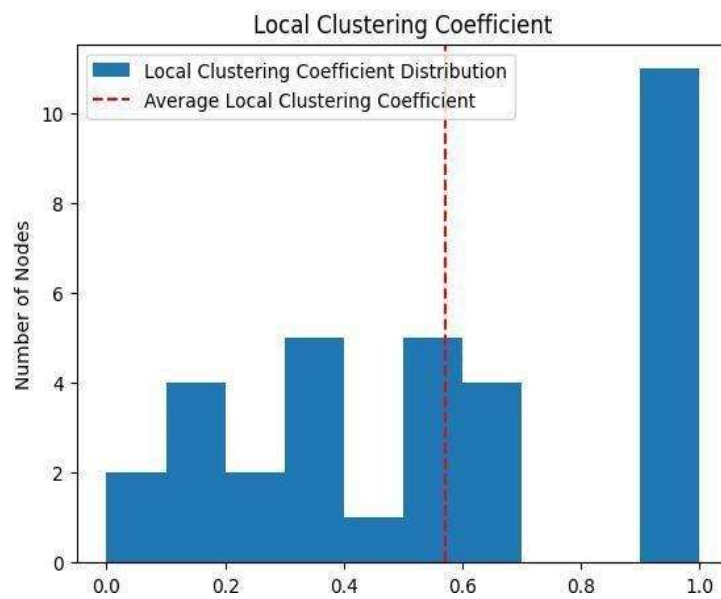


Fig 4.3 Local Clustering Coefficient Distribution

```
[10] from networkx.algorithms.community.modularity_max import greedy_modularity_communities

c=list(greedy_modularity_communities(ZKC_graph))
print(len(c))

3
```

Now we will check communities by firstly importing the required packages. The output shows 3 which means there are 3 communities here.

```
[11] community_0=sorted(c[0])
      community_1=sorted(c[1])
      community_2=sorted(c[2])

      print(community_0)
      print(community_1)
      print(community_2)

      [8, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]
      [1, 2, 3, 7, 9, 12, 13, 17, 21]
      [0, 4, 5, 6, 10, 11, 16, 19]
```

We will print the nodes in the respective communities.

```
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_0,node_color='g',alpha=0.5)
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_1,node_color='r',alpha=0.5)
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_2,node_color='b',alpha=0.5)

nx.draw_networkx_edges(ZKC_graph,circ_pos,width=0.2)
nx.draw_networkx_labels(ZKC_graph,circ_pos,club_labels,font_size=0.9)

plt.show()
```

Now we will check the communities graphically.

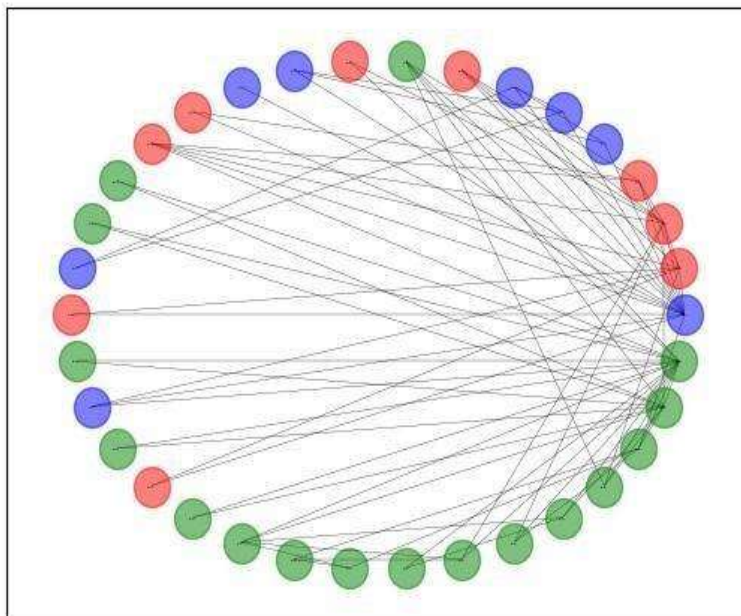


Fig 4.4 Graphical Plot of Communities

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 SUMMARY

Social Networks Visualizer (SocNetV) is a flexible and user-friendly tool for the analysis and visualization of Social Networks. It lets you construct networks (mathematical graphs) with a few clicks on a virtual canvas or load networks of various formats. The application can compute basic network properties, such as density, diameter and distances (shortest path lengths), as well as more advanced structural statistics, such as node and network centralities (i.e. closeness, betweenness, graph), clustering coefficient, etc. Various layout algorithms (i.e. Spring-embedder, radial and layered according to node centralization) are supported for meaningful visualizations of your networks.

5.2 CONCLUSION

Social network analysis is a method of analyzing social structures through the use of networks and graph theory. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them. Social network analysis is used in many fields such as sociology, psychology, economics, and computer science. It can be used to study social networks like Facebook and Twitter.

5.3 FUTURE WORK

Using modern technologies which would evolve in coming years and decade,we can make this implementation even more responsive,simple and we would be able to do many more such operations in a matter of seconds.We can even use now many inbuilt packages of python which is one of the most popular languages in the world to make it more descriptive and we can design involving interfaces. We have lot of things we can add to this miniproject.

APPENDIX

A. SAMPLE CODE

```
[3] import sys
import networkx as nx
import matplotlib.pyplot as plt
```

```
[4] ZKC_graph = nx.karate_club_graph()
Mr_Hi=0
John_A=33
club_labels=nx.get_node_attributes(ZKC_graph,'club')
print({key:club_labels[key] for key in range(10,16)})
```

```
[5] circ_pos=nx.circular_layout(ZKC_graph)
nx.draw(ZKC_graph,circ_pos)
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=[Mr_Hi],node_color='g',alpha=1)
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=[John_A],node_color='r',alpha=1)
```

```
[6] density=nx.density(ZKC_graph)
print("The edge density is: "+str(density))
```

```
The edge density is: 0.13903743315508021
```

```
[7] degree=ZKC_graph.degree()
    degree_list=[]

    for (n,d) in degree:
        degree_list.append(d)

    av_degree=sum(degree_list)/len(degree_list)
    print("The average degree is "+str(av_degree))
```

The average degree is 4.588235294117647

```
[8] plt.hist(degree_list,label='Degree Distribution')
    plt.axvline(av_degree,color='r',linestyle='dashed',label='Average Degree')
    plt.legend()
    plt.ylabel('Number of Nodes')
    plt.title('Karate Club: Node Degree')
```

```
[9] local_clustering_coefficient=nx.algorithms.cluster.clustering(ZKC_graph)
    av_local_clustering_coefficient=sum(local_clustering_coefficient.values())/len(local_clustering_coefficient)

    plt.hist(local_clustering_coefficient.values(),label='Local Clustering Coefficient Distribution')
    plt.axvline(av_local_clustering_coefficient,color='r',linestyle='dashed',label='Average Local Clustering Coefficient')
    plt.legend()
    plt.ylabel('Number of Nodes')
    plt.title('Local Clustering Coefficient')
    plt.show()
```

```
[10] from networkx.algorithms.community.modularity_max import greedy_modularity_communities
```

```
c=list(greedy_modularity_communities(ZKC_graph))  
print(len(c))
```

3

```
[11] community_0=sorted(c[0])  
community_1=sorted(c[1])  
community_2=sorted(c[2])
```

```
print(community_0)  
print(community_1)  
print(community_2)
```

```
[8, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]  
[1, 2, 3, 7, 9, 12, 13, 17, 21]  
[0, 4, 5, 6, 10, 11, 16, 19]
```

```
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_0,node_color='g',alpha=0.5)  
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_1,node_color='r',alpha=0.5)  
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_2,node_color='b',alpha=0.5)
```

```
nx.draw_networkx_edges(ZKC_graph,circ_pos,width=0.2)  
nx.draw_networkx_labels(ZKC_graph,circ_pos,club_labels,font_size=0.9)
```

```
plt.show()
```

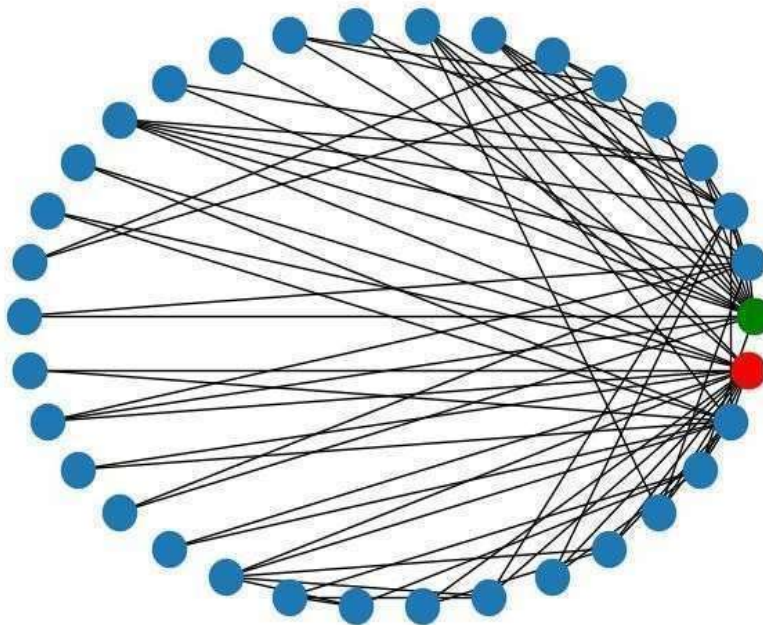
B. SCREENSHOTS

```
[4] ZKC_graph = nx.karate_club_graph()
    Mr_Hi=0
    John_A=33
    club_labels=nx.get_node_attributes(ZKC_graph,'club')
    print({key:club_labels[key] for key in range(10,16)})
```

```
{10: 'Mr. Hi', 11: 'Mr. Hi', 12: 'Mr. Hi', 13: 'Mr. Hi', 14: 'Officer', 15: 'Officer'}
```

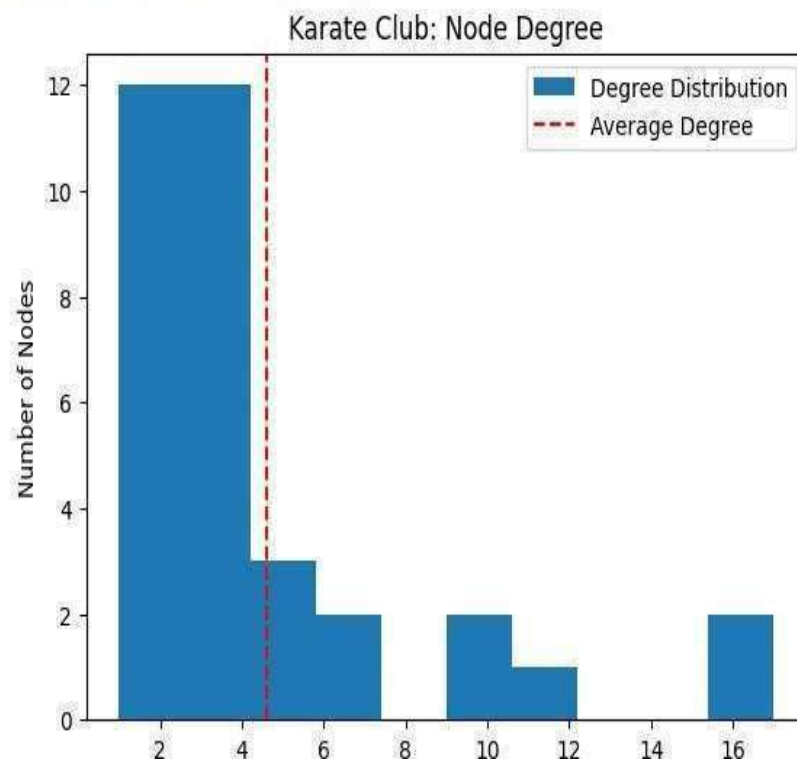
```
[5] circ_pos=nx.circular_layout(ZKC_graph)
    nx.draw(ZKC_graph,circ_pos)
    nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=[Mr_Hi],node_color='g',alpha=1)
    nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=[John_A],node_color='r',alpha=1)
```

```
<matplotlib.collections.PathCollection at 0x7fdee9cda070>
```



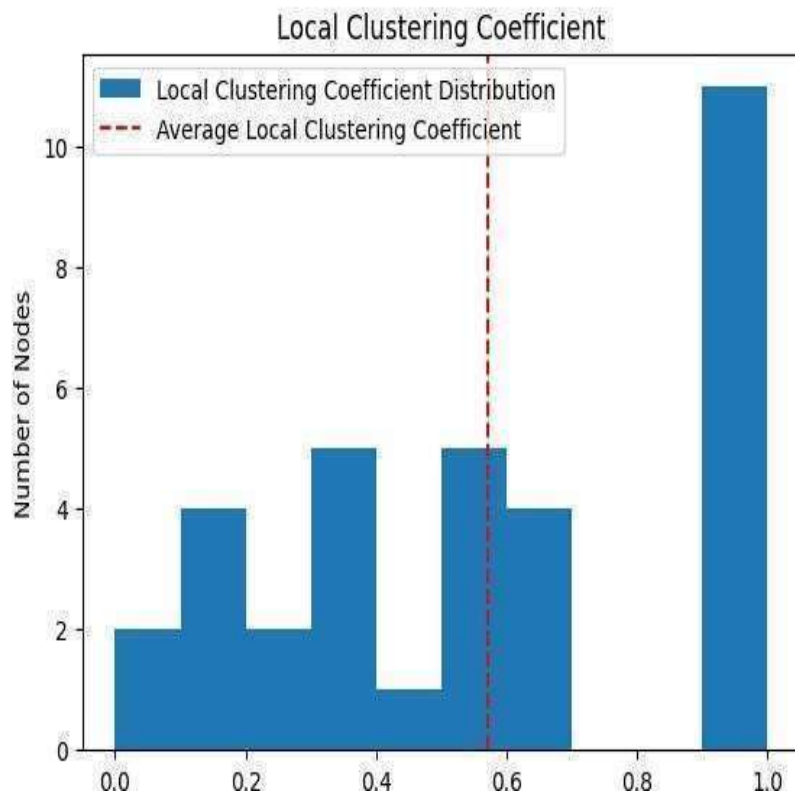
```
[8] plt.hist(degree_list,label='Degree Distribution')
    plt.axvline(av_degree,color='r',linestyle='dashed',label='Average Degree')
    plt.legend()
    plt.ylabel('Number of Nodes')
    plt.title('Karate Club: Node Degree')
```

Text(0.5, 1.0, 'Karate Club: Node Degree')



```
[9] local_clustering_coefficient=nx.algorithms.cluster.clustering(ZKC_graph)
    av_local_clustering_coefficient=sum(local_clustering_coefficient.values())/len(local_clustering_coefficient)

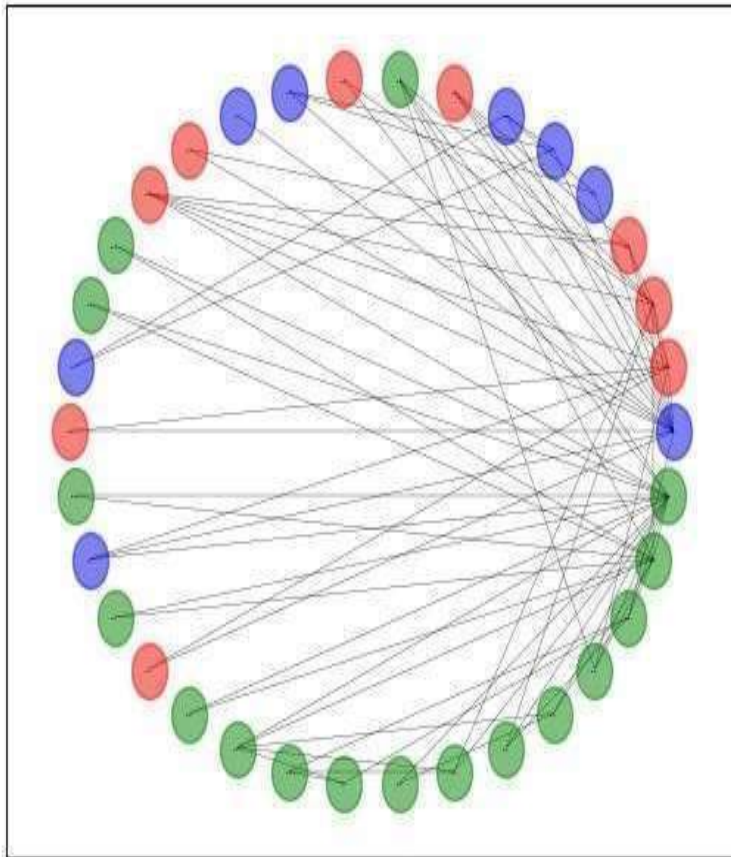
    plt.hist(local_clustering_coefficient.values(),label='Local Clustering Coefficient Distribution')
    plt.axvline(av_local_clustering_coefficient,color='r',linestyle='dashed',label='Average Local Clustering Coefficient')
    plt.legend()
    plt.ylabel('Number of Nodes')
    plt.title('Local Clustering Coefficient')
    plt.show()
```



```
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_0,node_color='g',alpha=0.5)
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_1,node_color='r',alpha=0.5)
nx.draw_networkx_nodes(ZKC_graph,circ_pos,nodelist=community_2,node_color='b',alpha=0.5)

nx.draw_networkx_edges(ZKC_graph,circ_pos,width=0.2)
nx.draw_networkx_labels(ZKC_graph,circ_pos,club_labels,font_size=0.9)

plt.show()
```



REFERENCES

1. <https://www.geeksforgeeks.org/>
2. <https://www.tutorialspoint.com/index.htm>
3. <https://www.w3schools.com/>
4. <https://www.javatpoint.com/>
5. <https://www.tutorialspoint.com/index.htm>
6. S. Kumar, F. Spezzano, V.S. Subrahmanian, C. Faloutsos. Edge Weight Prediction in Weighted Signed Networks. IEEE International Conference on Data Mining (ICDM), 2016.
7. Applied Social Networks Analysis, Michigan University Online Course, Coursera