Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme	: B.Tech
Course Title	: Social and Information Networks
Course Code	: CSE3021
Slot	: C1+TC1
Title	: Trust Network Analysis
Team Members	: RS Jyothish / 20BCE1040
	Nilavan I / 20BCE1080

Faculty: Dr Punitha K

Sign: [illegible]

Date: 10.04.2023

A project report on

Trust Network Analysis

'

Submitted in partial fulfillment for the course

Social and Information Networks

by

RS Jyothish (20BCE1040)

Nilavan I (20BCE1080)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DECLARATION

We hereby declare that the thesis entitled "Trust Network Analysis " submitted by us, for the completion of the course, Social and Information Networks(CSE3021) is a record of bonafide work carried out by us under the supervision of Dr Punitha K, our course instructor. Wefurther declare that the work reported in this document has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

[illegible]

Place: Chennai
Date: 10.04.23

RS Jyothish (20BCE1040)

I. Nilavan

Nilavan I (20BCE1080)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"Trust Network Analysis"** is prepared and submitted by **RS Jyothish(20BCE1040)** and **Nilavan I(20BCE1080)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the course, **Social and Information Networks (CSE3021),** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for any other course and the same is certified.

[illegible]

Name: Dr. Punitha K

Signature of the Faculty

Date: 10.04.23

ABSTRACT

This study analyzes the social network structure of Epinions, a general consumer review site. The site allowed users to create free accounts and earn money based on the usefulness of their reviews. However, there were attempts to manipulate the system and lower the trustworthiness of the reviews, prompting the introduction of a trust system. The trust system allowed users to designate trusted sources and block those whose reviews were deemed unhelpful or offensive.

This study analyzes the structure of the who-trust-whom network on the site and interprets the properties of structural balance. This work found that the trust system improved the quality of reviews on the site and provided a mechanism for users to identify trustworthy sources. It also demonstrates the usefulness of structural analysis in understanding social networks and their properties.

Overall, this work highlights the importance of trust and reputation systems in online communities and the potential for structural analysis to inform the design and improvement of such systems. By understanding the properties of social networks, we can design systems that promote trust and collaboration while minimizing the negative effects of manipulation and dishonesty.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

SCC	Strongly Connected Components
WCC	Weakly Connected Components

Chapter 1

Introduction

1.1 INTRODUCTION

Social networks are a representation of the relationships between individuals, groups, or organizations. These relationships can be visualized in the form of a graph, with the nodes representing the entities and the edges representing the connections between them. While social networks like Facebook and Twitter are well-known, there are other types of networks that can be analyzed as well.

One such network is the trust network, which is based on the concept of trust between individuals. In the case of the Epinions website, users could review products and earn money based on the usefulness of their reviews. However, the system was vulnerable to manipulation, leading to the implementation of a trust system. Users could designate trusted sources and block reviewers whose contributions were deemed unhelpful or offensive.

This trust system created a trust network, which can be analyzed to understand the behavior and interactions of the individuals in the network. The analysis of this network can provide insights into the properties of the network, such as the adoption of opinions and the formation of communities. This type of analysis can be useful in understanding how individuals make decisions and interact with each other in a trust-based environment.

1.2 OVERVIEW OF TRUST NETWORK

A trust network is a type of social network where the relationships between nodes (or individuals) are based on trust. In a trust network, individuals may establish trust relationships with others based on personal experiences, recommendations from trusted sources, or other factors. These trust relationships can be used to evaluate the trustworthiness of other individuals in the network, and to determine who to trust when making decisions.

Trust networks are often used in online communities to improve the quality of user-generated content, such as product reviews or recommendations. By allowing users to designate trusted sources and block those whose contributions are deemed unhelpful or offensive, trust networks can help to reduce the impact of manipulation and bias. Trust networks can also be used in other contexts, such as in business or political networks, to facilitate decision-making and collaboration based on trust relationships.

1.3 CHALLENGES IN EPINIONS

The Epinions website provided a platform for users to review products and services, and they could earn money based on the usefulness of their reviews. However, this system was found to be vulnerable to manipulation, as some users were incentivized to post false or misleading reviews to boost their earnings. This practice not only undermined the credibility of the reviews but also made it difficult for other users to make informed decisions based on them.

To address this issue, a trust system was implemented, which allowed users to designate other users as trusted sources based on the usefulness and reliability of their reviews. Conversely, users could also create a blocklist of reviewers whose reviews were consistently unhelpful or even harmful to the community.

Despite the implementation of the trust system, the website still faced challenges in maintaining the integrity of its reviews. Nonetheless, this case highlights the importance of trust and credibility in online communities, and the need for effective measures to combat manipulation and fraud.

1.4 PROJECT STATEMENT

This project aims to investigate the effectiveness of a trust system implemented on the Epinions website to combat manipulation and fraud in user-generated reviews. The study will analyze the resulting trust network to gain insights into the behavior and interactions of individuals within the network. By examining properties such as opinion adoption and community formation, the project will provide valuable information on how individuals make decisions and interact in a trust-based online environment. The findings of this project will contribute to the development of effective measures to maintain the integrity of online communities and user-generated content.

1.5 OBJECTIVES

1. To analyze the trust network created by the implementation of a trust system on the Epinions website.
2. To identify the properties of the network, such as the adoption of opinions and the formation of communities.
3. To gain insights into how individuals make decisions and interact with each other in a trust-based environment.
4. To understand the challenges faced in maintaining the integrity of reviews and combating manipulation and fraud in online communities.
5. To highlight the importance of trust and credibility in online communities and the need for effective measures to maintain them.

1.6 SCOPE OF THE PROJECT

The scope of the project would involve analyzing the trust network created by the trust system implemented on the Epinions website. This would include identifying the properties of the network, such as the adoption of opinions and formation of communities, and understanding how individuals make decisions and interact with each other in a trust-based environment. The analysis could be conducted using network analysis techniques and algorithms, such as the Louvain algorithm, to detect communities and measure properties like reciprocity. The project could also include exploring the limitations and effectiveness of the trust system in combating manipulation and fraud on the website.

Chapter 2

Background

2.1 INTRODUCTION

The Epinions website provided a platform for users to review products and services, and they could earn money based on the usefulness of their reviews. However, this system was found to be vulnerable to manipulation, as some users were incentivized to post false or misleading reviews to boost their earnings. This practice not only undermined the credibility of the reviews but also made it difficult for other users to make informed decisions based on them.

To address this issue, a trust system was implemented, which allowed users to designate other users as trusted sources based on the usefulness and reliability of their reviews. Conversely, users could also create a blocklist of reviewers whose reviews were consistently unhelpful or even harmful to the community.

Despite the implementation of the trust system, the website still faced challenges in maintaining the integrity of its reviews. Nonetheless, this case highlights the importance of trust and credibility in online communities, and the need for effective measures to combat manipulation and fraud.

This trust system created a trust network, which can be analyzed to understand the behavior and interactions of the individuals in the network. The analysis of this network can provide insights into the properties of the network, such as the adoption of opinions and the formation of communities. This type of analysis can be useful in understanding how individuals make decisions and interact with each other in a trust-based environment.

2.2 SURVEY ON TRUST NETWORK

[1] Leskovec, Huttenlocher, and Kleinberg's (2010) paper explores the concept of signed social networks, where the polarity of relationships between individuals is considered. The authors analyze data from two social networking platforms, Epinions and Slashdot, to investigate the implications of positive and negative interactions on social networks. They found that individuals tend to be connected to others who share similar opinions, and negative relationships are more prevalent than positive relationships. Negative relationships can play a significant role in shaping the network structure and can lead to the formation of distinct sub-communities. The paper provides a novel approach for understanding the dynamics of social interactions in online communities, with important implications for understanding the formation and evolution of social networks and the impact of positive and negative interactions on user behavior.

[3] The paper examines the efficiency of searching in power-law networks, which are characterized by a few highly connected nodes (known as "hubs") and many poorly connected nodes. The authors propose a new algorithm called "preferential walk," which takes advantage of the power-law distribution of links in the network to improve search efficiency. They compare the performance of preferential walk with other search algorithms, such as random walks and breadth-first search, and show that preferential walk outperforms them in terms of both speed and accuracy. The authors also discuss the implications of their findings for the design of search engines and other network-based applications.

[5] The paper investigates the origins of the power-law degree distribution in social networks, which is the observation that the number of connections (degree) of nodes follows a power-law distribution. The authors argue that the heterogeneity of human activity, specifically the tendency for some individuals to be more active and have more connections than others, is a key factor that contributes to this distribution. They use empirical data from a large online social network and a model that incorporates heterogeneity in node activity to demonstrate that the power-law degree distribution emerges naturally from the dynamics of the network. They also show that the heterogeneity in node activity is related to factors such as age, gender, and geographic location. The authors conclude that understanding the origins of the power-law degree distribution is important for predicting and managing the behavior of social networks.

[7] The paper examines the concept of the "rich-club" in weighted networks, which refers to a group of highly connected nodes that are also highly weighted (i.e., have a large amount of resources flowing through them). The authors propose a weighted version of the rich-club coefficient, which measures the extent to which the rich-club nodes are more interconnected than would be expected based on their degree and weight alone. They find that the weighted rich-club effect is more pronounced in real-world networks than in randomized models, and that it is especially strong in networks related to transportation and information flow. The authors also investigate the relationship between the weighted rich-club effect and the control of the network, showing that the rich-club nodes play a crucial role in controlling the overall flow of resources in the network. The paper concludes that the weighted rich-club effect provides insights into the organization and function of complex networks, and has important implications for network design and management.

[9] The paper investigates the organization of the human connectome, which is the network of connections between different brain regions. The authors analyze the degree and strength distributions of the connectome and find that it exhibits a rich-club organization, meaning that highly connected and highly weighted nodes are more likely to be connected to each other than would be expected by chance. They also show that the rich-club nodes are located in areas of the brain associated with higher-order cognitive functions, such as attention and memory. The authors use a simulation to demonstrate that the rich-club organization is not simply a consequence of the degree and weight distributions, but rather reflects a fundamental property of the network. They conclude that the rich-club organization of the human connectome plays a crucial role in its function, and that understanding this organization is important for advancing our understanding of the brain's complex information processing capabilities.

[11] The paper explores the adoption behavior of big data platforms among groups using social network analysis. The authors propose a model that incorporates factors such as network structure, individual characteristics, and external influences to predict the adoption behavior of groups. They test the model using data from a large social network platform and find that it performs well in predicting the adoption behavior of different groups. The authors also use the model to investigate the impact of various factors on adoption behavior, such as the size and density of the network and the influence of opinion leaders.

[14] The paper examines the Technology Acceptance Model (TAM) in the context of social

network adoption. The TAM is a widely used framework for understanding the adoption of technology, which posits that perceived usefulness and perceived ease of use are the key factors driving adoption behavior. The authors conduct a survey to test the TAM in the context of social network adoption and find that both perceived usefulness and perceived ease of use are significant predictors of adoption behavior. They also find that other factors, such as social influence and trust, have a significant impact on adoption behavior, highlighting the importance of considering social factors in technology adoption. The authors conclude that the TAM is a useful framework for understanding social network adoption, but that it should be supplemented with other factors that are specific to the social context of the technology.

[15] The paper provides an overview of community detection in social networks, which is the process of identifying groups of nodes that are densely connected within the network. The authors review various methods for community detection, including hierarchical clustering, modularity optimization, and label propagation. They also discuss different types of networks, such as directed, weighted, and signed networks, and how they can impact community detection. The paper highlights the importance of community detection in various applications, such as social media analysis, recommendation systems, and targeted advertising. The authors also identify some of the challenges and limitations of community detection, such as the trade-off between accuracy and scalability, the sensitivity to network structure, and the lack of a clear definition of what constitutes a community. The paper concludes with a discussion of some future research directions in community detection, such as incorporating dynamic networks and considering multiple levels of community structure.

[17] The paper presents a comprehensive benchmarking study of community detection methods in social networks using a procedure-oriented framework. The authors compare 14 state-of-the-art community detection methods using a variety of metrics, such as modularity, coverage, and conductance, on both synthetic and real-world datasets. They also investigate the impact of various factors, such as network size, density, and community structure, on the performance of the methods. The paper highlights some of the key findings of the study, such as the effectiveness of spectral clustering and modularity optimization methods, the importance of tuning parameters for each method, and the sensitivity of the methods to the network structure and community size. The authors conclude that the procedure-oriented framework provides a useful tool for evaluating and comparing community detection methods and that their study can serve as a reference for future research in this area.

[19] The paper presents a new framework for understanding tie strength in social networks based on the local "bow tie" structure of the network. The authors propose that the strength of a tie between two nodes can be inferred from the size and shape of the "bow tie" structure that surrounds them in the network. The paper provides empirical evidence for this framework by analyzing two real-world social networks: a network of co-authorships in computer science and a network of mobile phone contacts. The authors demonstrate that tie strength is positively correlated with the size of the "in" and "out" components of the bow tie structure, which represent the nodes that are reachable from and can reach the tie pair, respectively. They also show that tie strength is negatively correlated with the size of the "tendrils" and "tubes" components of the bow tie structure, which represent the nodes that are reachable from one of the tie pair but not from the other. The paper concludes that the bow tie framework provides a simple and intuitive way to understand tie strength in social networks and has the potential to inform the design of more effective social network interventions and applications.

[21] This paper highlights some methodological issues related to the measurement of perceived assortativity in social networks, and suggests some solutions to address these problems. The authors argue that traditional measures of assortativity, such as Pearson's correlation coefficient or Newman's assortativity coefficient, may not be appropriate for measuring the perceived assortativity of social networks, as they rely on the assumption that the degree distribution of the network is known or can be estimated accurately. However, in many cases, the degree distribution of a social network may not be known, especially in the case of egocentric networks or online social networks where network data is often incomplete or inaccurate. The authors propose an alternative measure of perceived assortativity based on the comparison of the observed degree distribution of the network with a null model that randomly assigns edges to nodes, and show that this measure can be used to estimate the

perceived degree correlation between pairs of nodes in a network. They also discuss some other methodological issues related to the measurement of perceived assortativity, such as the use of different types of tie strength and the influence of homophily and social influence on network structure. The paper concludes by emphasizing the importance of carefully considering these methodological issues when studying social networks and making inferences about their structure and dynamics.

[24] This paper describes NetworkX, a Python software package for the creation, manipulation, and study of complex networks. The authors outline the basic principles of NetworkX and provide examples of how it can be used to explore network structure, dynamics, and function. They discuss the different types of networks that can be modeled using NetworkX, including directed and undirected graphs, bipartite graphs, and multigraphs, as well as different types of node and edge attributes. They also describe some of the built-in algorithms and functions that are available in NetworkX for studying network properties such as degree distribution, centrality, and clustering. Finally, they discuss some applications of NetworkX in various fields, including biology, social science, and engineering, and highlight some of the challenges and future directions for network analysis. Overall, the paper provides a comprehensive introduction to the capabilities of NetworkX and its potential for analyzing complex networks.

[25] The paper describes the NetworkX Python package, which provides a framework for creating, manipulating, and analyzing complex networks or graphs. The package includes various algorithms for graph analysis, such as centrality measures, community detection, and path finding. The authors highlight the versatility and efficiency of NetworkX, and provide examples of its use in a range of applications, including social network analysis, transportation planning, and neuroscience. The paper also discusses the package's compatibility with other Python libraries, its open-source development, and its user-friendly documentation.

Chapter 3

Methodology

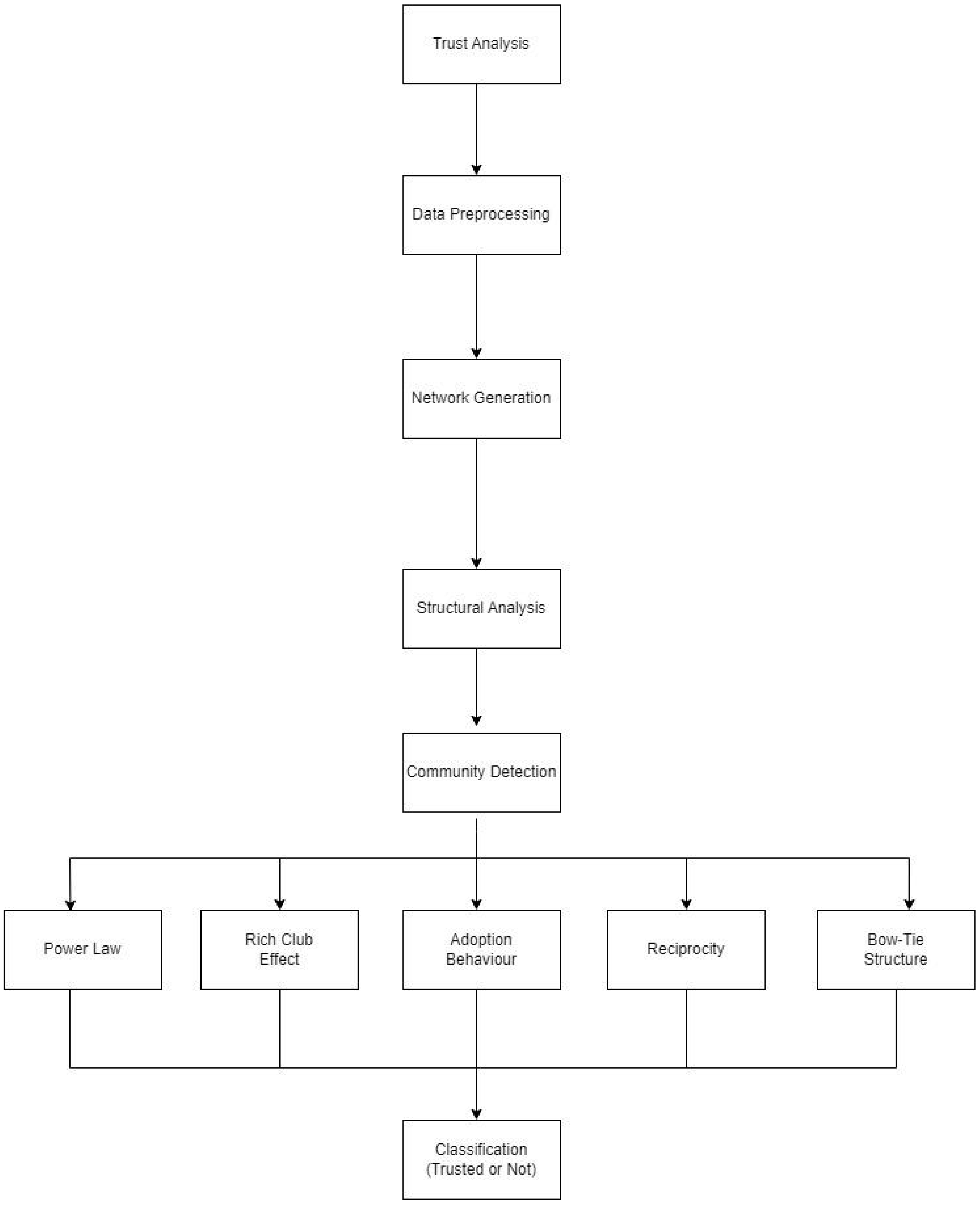


Figure 1: Workflow diagram

3.1 STRUCTURAL ANALYSIS

The given network is represented as a directed weighted graph, where edges with a weight of one indicate trusting relationships, while edges with a weight of negative one indicate distrusting relationships. This is a sparsely connected network with a high clustering coefficient, which is typical of social networks. The average clustering coefficient is greater than what would be expected in a random network, due to transitivity. Additionally, the diameter of the network is relatively small when compared to its size. It is worth noting that the majority of the edges in the network (about 85%) are positive.

Table 1: Structure of the network

Number of nodes	131828
Number of edges	841372
Average Clustering	0.09561744905322256
Transitivity	0.07428166527700729
Density	4.841456374018419e-05

3.2 POWER LAW

Power law refers to a mathematical relationship between two variables where one variable's value is proportional to the power of the other variable. In other words, it is a relationship of the form y = kx^a, where x and y are variables, k is a constant, and a is the exponent that determines the degree of the power law relationship.

In social networks, power law is often used to describe the distribution of connections (degrees) among nodes. In a power-law network, a few nodes have a large number of connections, while the majority of nodes have only a few connections. This pattern is also known as a "scale-free" network because it does not have a characteristic scale (or typical size), unlike regular networks where all nodes have a similar number of connections.

3.3 RICH CLUB EFFECT

The rich club effect is a phenomenon observed in network analysis, where a group of well-connected nodes, known as the "rich club," forms a densely interconnected subgraph within a larger network. In other words, it refers to the tendency of highly connected nodes to preferentially connect with other highly connected nodes.

This effect can be observed in a variety of real-world networks, including social networks, biological networks, and the internet. The rich club effect has implications for the structure and function of networks, as it can influence the flow of information and resources through the network. It can also impact the resilience of the network to node or edge failures.

3.4 RECIPROCITY

Reciprocity in a network refers to the tendency for two nodes to form a mutual connection or link. In other words, if node A is connected to node B, and node B is also connected to node A, then there is reciprocity in the network. Reciprocity is a measure of the extent to which links in a network are mutual or bidirectional, as opposed to being unidirectional.

Reciprocity is commonly used as a metric to study the structural properties of social networks, where it can indicate the presence of certain social phenomena such as friendship or trust. For example, if there is high reciprocity in a social network, it may indicate that people in the network have close relationships and trust each other. On the other hand, if there is low reciprocity, it may indicate that the relationships in the network are more hierarchical or one-sided.

3.5 SMALL WORLD PHENOMENA

The small-world phenomenon refers to the observation that in many networks, including social networks, it is possible to reach any node from any other node through a relatively small number of intermediate steps. Small-world networks typically exhibit high clustering and short path lengths. This combination of high clustering and short path lengths is characteristic of networks that are somewhere between a regular lattice and a random graph.

Small-world networks play an important role in facilitating the spread of information and influence through a network, as well as in promoting cooperation and coordination among networked individuals.

3.6 BOW-TIE STRUCTURE

The bow-tie structure can be visualized as a bow-tie shape, with a central core of strongly connected components or SCC surrounded by two wings of nodes. The first wing, known as the in component, contains nodes that can reach the core but cannot be reached from it. The second wing, known as the out component, contains nodes that can be reached from the core but cannot reach it.

The core of the bow-tie is typically the most densely interconnected part of the network and represents a hub or backbone of the system. The in and out components are typically less densely interconnected and represent "entry points" and "exit points" to and from the core. It can help to explain how information and influence can spread through a network.

3.7 BEHAVIOUR ADOPTION

Adoption rate in a social network refers to the rate at which individuals within the network adopt a new behavior or idea. This can refer to a wide range of behaviors or ideas, such as using a new technology, following a new trend, or adopting a new social norm. The adoption rate is often influenced by various factors, including the network structure, the perceived benefits and costs of adoption, and social influence from other members of the network. The study of adoption rates in social networks is an important area of research for understanding how ideas behaviors spread through communities.

3.8 COMMUNITY DETECTION

Community detection in network analysis refers to the process of identifying groups of nodes in a network that are more densely connected to each other than to the rest of the network. These groups are often referred to as "communities" or "clusters" and can represent meaningful substructures within the larger network.

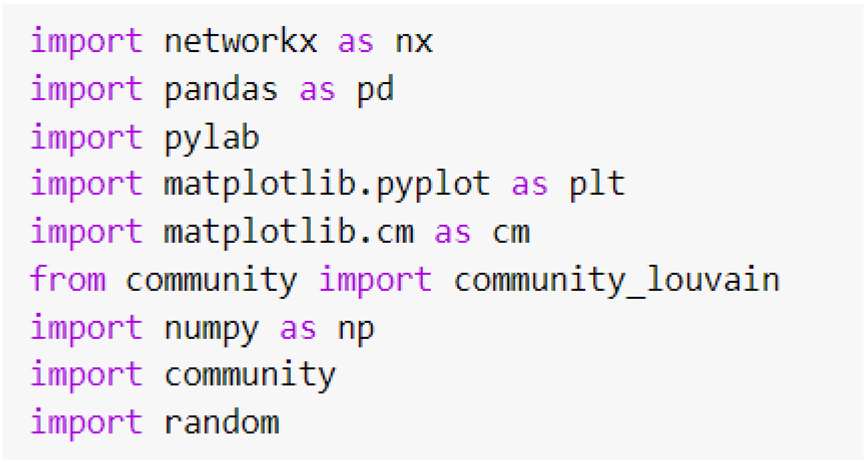
The goal of community detection is to uncover these substructures and to use them to gain insight into the overall organization and function of the network. This can involve identifying groups of nodes that are more likely to share similar attributes or perform similar functions, or it can help to identify important nodes or connections within the network that might otherwise be difficult to detect.

There are many different methods and algorithms for community detection, ranging from simple approaches based on measures of network density or connectivity, to more complex methods that use statistical models or machine learning techniques to identify and characterize communities. Some common methods for community detection include modularity-based methods, hierarchical clustering, spectral clustering, and network partitioning.

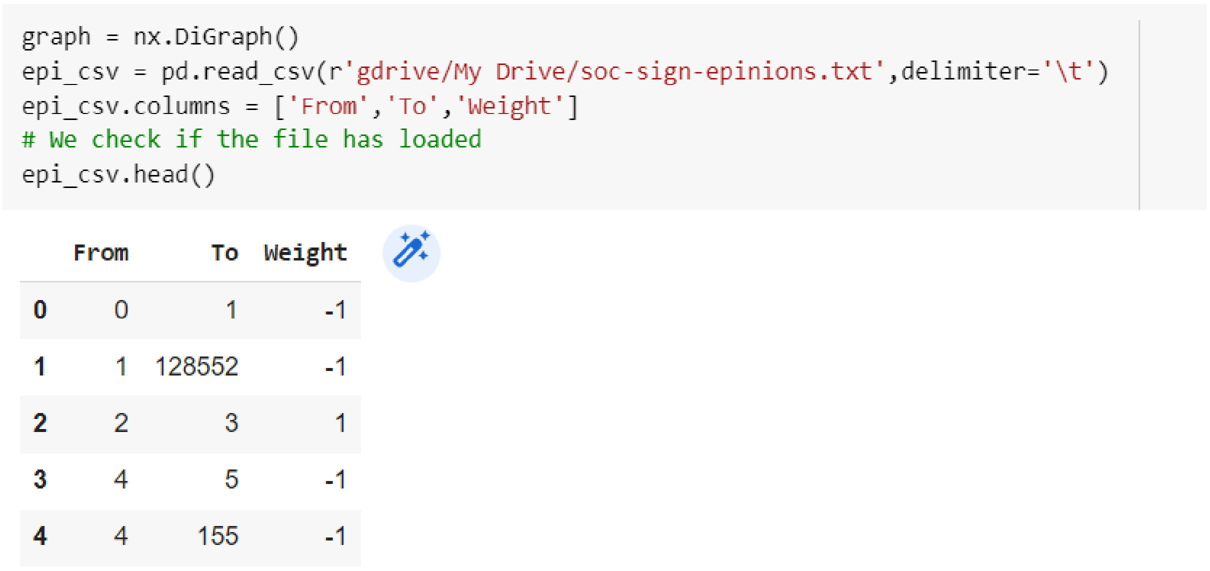
Chapter 4

Results and Discussions

4.1 CODE AND OUTPUTS:



```python
import networkx as nx
import pandas as pd
import pylab
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from community import community_louvain
import numpy as np
import community
import random
```



```python
graph = nx.DiGraph()
epi_csv = pd.read_csv(r'gdrive/My Drive/soc-sign-epinions.txt',delimiter='\t')
epi_csv.columns = ['From','To','Weight']
# We check if the file has loaded
epi_csv.head()
```

	From	To	Weight
0	0	1	-1
1	1	128552	-1
2	2	3	1
3	4	5	-1
4	4	155	-1

```python
print('Row count is:',len(epi_csv.axes[0]))
print('Column count is:',len(epi_csv.axes[1]))
```

```
Row count is: 841372
Column count is: 3
```

```python
# Adding the edges.
for index,row in epi_csv.iterrows():
    graph.add_edges_from([(row[0],row[1])],weight = row[2])
```

```python
# Positive and negative edges
Stats = pd.DataFrame(graph.out_degree(),columns=['From','Outdegree']).sort_values('From')
Stats_2 =  pd.DataFrame(graph.in_degree(),columns=['To','Indegree']).sort_values('To')
b= epi_csv.groupby('From',as_index=False)[['Weight']].sum().sort_values('From')
c= epi_csv.groupby('To',as_index=False)[['Weight']].sum().sort_values('To')
Stats = Stats.merge(b,on='From',how='left')
Stats_2 = Stats_2.merge(c,on='To',how='left')
Stats['Pos_out'] = (  Stats['Outdegree'] + Stats['Weight']  )/2
Stats['Neg_out'] = (  Stats['Outdegree'] - Stats['Weight']  )/2
Stats_2['Pos_in']= (Stats_2['Indegree']  + Stats_2['Weight'])/2
Stats_2['Neg_in']= (Stats_2['Indegree']  - Stats_2['Weight'])/2
Stats = pd.merge(Stats,Stats_2,left_on='From', right_on='To').drop('To', axis=1)
Stats = Stats.drop(['Weight_x','Weight_y'],axis=1)
Stats.fillna(0,inplace=True)
Stats.head()
```

	From	Outdegree	Pos_out	Neg_out	Indegree	Pos_in	Neg_in
0	0	1	0.0	1.0	0	0.0	0.0
1	1	1	0.0	1.0	2	1.0	1.0
2	2	1	1.0	0.0	1	1.0	0.0
3	3	0	0.0	0.0	4	4.0	0.0
4	4	14	9.0	5.0	1	1.0	0.0

```
[ ]  Stats.describe()
     print(Stats.max(axis=0))
```

```
From          131827.0
Outdegree       2070.0
Pos_out         2070.0
Neg_out         1562.0
Indegree        3478.0
Pos_in          3338.0
Neg_in           540.0
dtype: float64
```

Basic stats

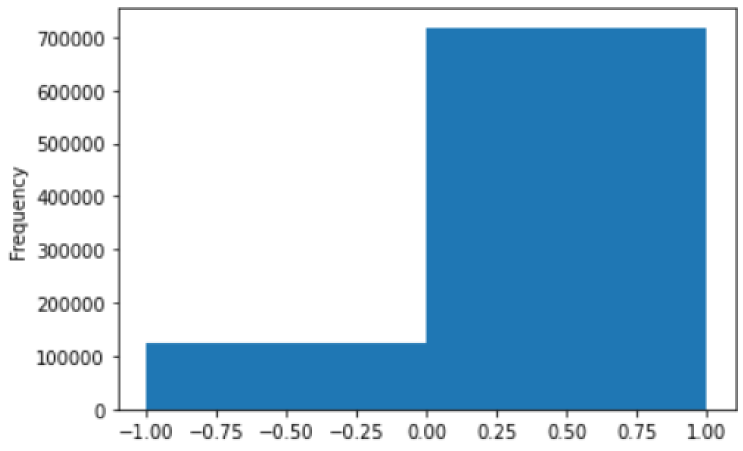
```
[ ]  print('Number of nodes:',graph.number_of_nodes())
     print('Number of edges:',graph.number_of_edges())
     print('Average Clustering:',nx.average_clustering(graph))
     print('Transitivity:',nx.transitivity(graph))
     print('Density:',nx.density(graph))
```

```
Number of nodes: 131828
Number of edges: 841372
Average Clustering: 0.09561744905322256
Transitivity: 0.07428166527700729
Density: 4.841456374018419e-05
```

Edge weight distribution

```
[ ]  epi_csv['Weight'].plot.hist(bins=2, alpha=1)
```

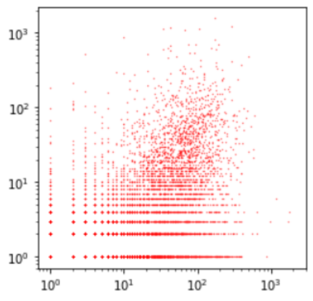
```
<AxesSubplot:ylabel='Frequency'>
```



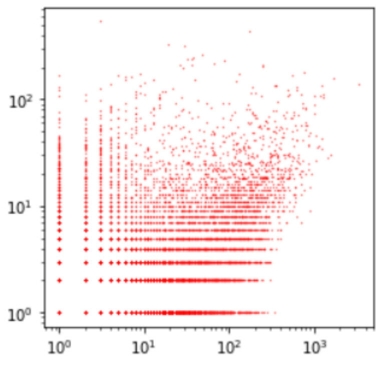
Log-Log plots of degrees

Out degree and indegree relationship, relationship between positive and negative weighted outgoing edges in the network and relationship between positive and negative ingoing edges.

```
#Positive and negative Out degree
plt.figure(figsize=(4,4))
plt.plot(Stats['Pos_out'],Stats['Neg_out'],'ro',markersize=0.25)
plt.xscale('log')
plt.yscale('log')
```

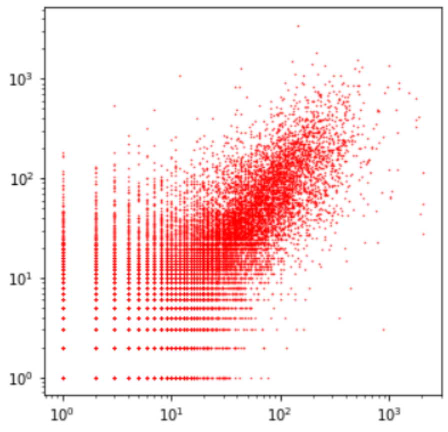


```
#Positive and negative In degree
plt.figure(figsize=(4,4))
plt.plot(Stats['Pos_in'],Stats['Neg_in'],'ro',markersize=0.25)
plt.xscale('log')
plt.yscale('log')
```



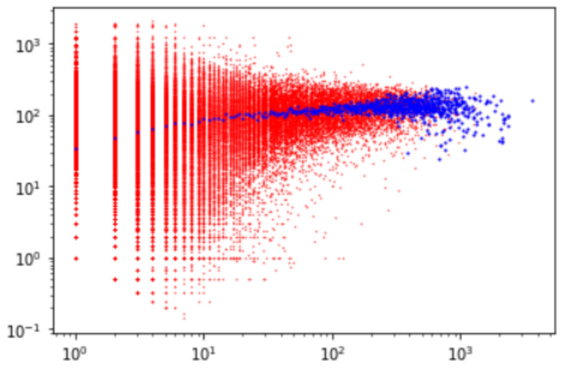
Out degree and indegree relationship.

```
#Out degree and In degree relationship
plt.figure(figsize=(5,5))
plt.plot(Stats['Outdegree'],Stats['Indegree'],'ro',markersize=0.4)
plt.xscale('log')
plt.yscale('log')
```

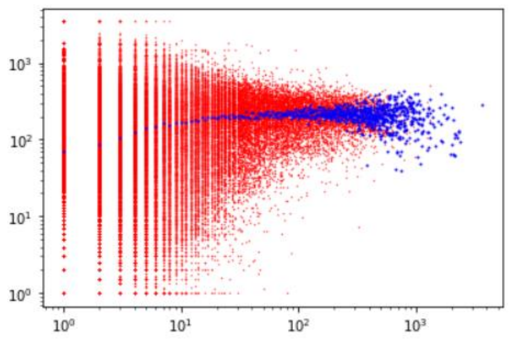


Plotting the assortativity coefficients of the nodes.

```
Deg_cor = nx.average_neighbor_degree(graph,target='out')
dict_list = []
for key, value in Deg_cor.items():
    temp = [key,value]
    temp[0] = graph.degree(key,'out')
    dict_list.append(temp)
dfa1 = pd.DataFrame(dict_list,columns =['Outdegree','Average neighbors outdegree'])
dfa2 = dfa1.groupby('Outdegree',as_index=False)['Average neighbors outdegree'].mean()
plt.plot(dfa1['Outdegree'],dfa1['Average neighbors outdegree'],'ro',markersize=0.3)
plt.plot(dfa2['Outdegree'],dfa2['Average neighbors outdegree'],'bo',markersize=1)
plt.xscale('log')
plt.yscale('log')
```



```
[ ] Deg_cor = nx.average_neighbor_degree(graph,target='in')
    dict_list = []
    for key, value in Deg_cor.items():
        temp = [key,value]
        temp[0] = graph.degree(key,'in')
        dict_list.append(temp)
    dfa1 = pd.DataFrame(dict_list,columns =['Indegree','Average neighbors indegree'])
    dfa2 = dfa1.groupby('Indegree',as_index=False)['Average neighbors indegree'].mean()
    plt.plot(dfa1['Indegree'],dfa1['Average neighbors indegree'],'ro',markersize=0.3)
    plt.plot(dfa2['Indegree'],dfa2['Average neighbors indegree'],'bo',markersize=1)

    plt.xscale('log')
    plt.yscale('log')
```



Checking assortativity

```
[ ] print(nx.degree_assortativity_coefficient(graph,'in','in'))
    print(nx.degree_assortativity_coefficient(graph,'out','out'))
```

```
0.005121469550551876
-0.030516704380786377
```

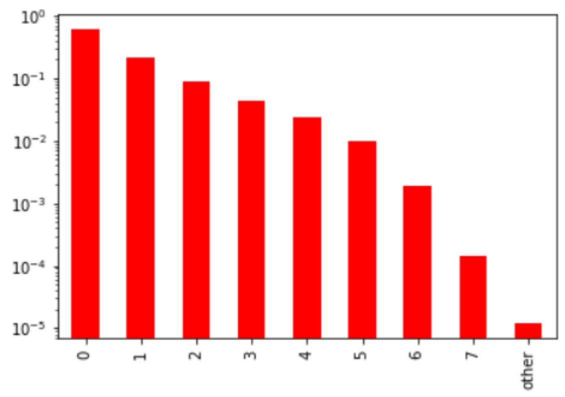
```
[ ] nx.degree_assortativity_coefficient(graph)
```

```
-0.06430169476171486
```

Checking the power law for In-degree

```
import math
pos = Stats['Indegree'][Stats['Indegree'] != 0]
pos = pos.transform(lambda x: math.floor(math.log(x)) )

prob = pos.value_counts(normalize=True)
threshold = 0.0001
mask = prob > threshold
tail_prob = prob.loc[~mask].sum()
prob = prob.loc[mask]
prob['other'] = tail_prob
prob.plot(kind='bar',log=True,color='r')
```

```
<AxesSubplot:>
```



Finding the power law coefficient of the indegree

```
import math
import igraph

pos = Stats['Indegree'][Stats['Indegree'] != 0]
igraph.power_law_fit(pos)
```

```
FittedPowerLaw(continuous=False, alpha=1.7049436066757273, xmin=1.0, L=-197839.7497898456, D=0.02086119220655669, p=0.0)
```

Size of components for bowtie structure

```
[ ]  SCC = max(nx.strongly_connected_components(graph), key=len)
     print('Size of maximal strongly connected component is ' + str(len(SCC)))
     WCC = max(nx.weakly_connected_components(graph), key=len)
     print('Size of maximal weakly connected component is ' + str(len(WCC)))
     DIF = list(set(WCC)-set(SCC))
     OUT_C = []
     IN_C = []
     IN = 0
     OUT = 0
     for index, node1 in enumerate(DIF):
         for index2, node2 in enumerate(SCC):

             if graph.has_edge(node1,node2):
                 OUT+=1
                 OUT_C.append(node2)
                 break
             elif graph.has_edge(node2,node1):
                 IN+=1
                 IN_C.append(node1)
                 break


     TND = len(DIF)-IN-OUT
     print(IN,OUT,TND)
```

```
     Size of maximal strongly connected component is 41441
     Size of maximal weakly connected component is 119130
     29732 37682 10275
```

Community detection in graph

```
[ ]  SCC = max(nx.strongly_connected_components(graph), key=len)
     scc_com = graph.subgraph(SCC).copy()
     scc2 = nx.Graph(scc_com)

     # Removing the weights.

     for u,v,d in scc2.edges(data=True):
         d['weight']=1

     communities =community_louvain.best_partition(scc2,random_state=5)
```

```
[ ]  print(set(communities.values()))
```

```
     {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
[ ]  ind_graph = community_louvain.induced_graph(communities, scc2)

     inv_map = dict()
     for key, value in communities.items():
         inv_map.setdefault(value, list()).append(key)

     # Removing the small communities

     inv_map2 = {key:val for key, val in inv_map.items() if len(val) >= 100}
     sizes = np.array([len(inv_map2[k]) for k in list(inv_map2.keys())])

     # Removing edges from the graph:

     inv_keys = [key for key in inv_map2]
     ind_nodes = list(ind_graph.nodes())
     for node in ind_nodes:
         if node not in inv_keys:
             ind_graph.remove_node(node)
```

```
[ ]  community_louvain.modularity(communities,scc2)
```

```
     0.4328499720766367
```

Getting the sizes of the communities

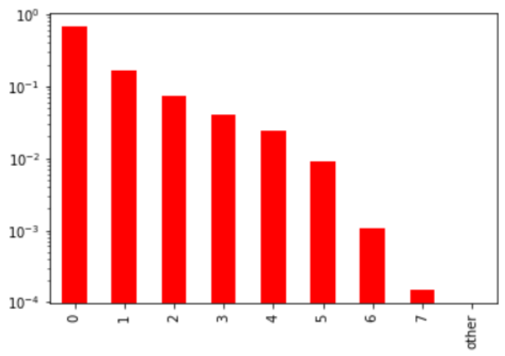
```
[ ]  print(sizes)
```

```
     [10032 10540   453 11587  4309   221   143   945   311   111   124]
```

Checking power law for outdegree

```
import math
pos = Stats['Outdegree'][Stats['Outdegree'] != 0]
pos = pos.transform(lambda x: math.floor(math.log(x)) )

prob = pos.value_counts(normalize=True)
threshold = 0.0001
mask = prob > threshold
tail_prob = prob.loc[~mask].sum()
prob = prob.loc[mask]
prob['other'] = tail_prob
prob.plot(kind='bar',log=True,color='r')
```

```
<AxesSubplot:>
```

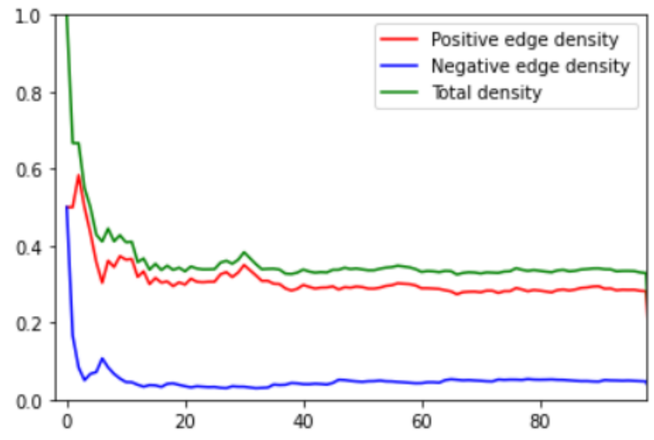


```
[ ] import math
    import igraph
    pos = Stats['Outdegree'][Stats['Outdegree'] != 0]
    igraph.power_law_fit(pos)
```

```
FittedPowerLaw(continuous=False, alpha=1.7290327710653879, xmin=2.0, L=-133428.12747064946, D=0.019071101789395528, p=0.0)
```

Checking the rich club effect

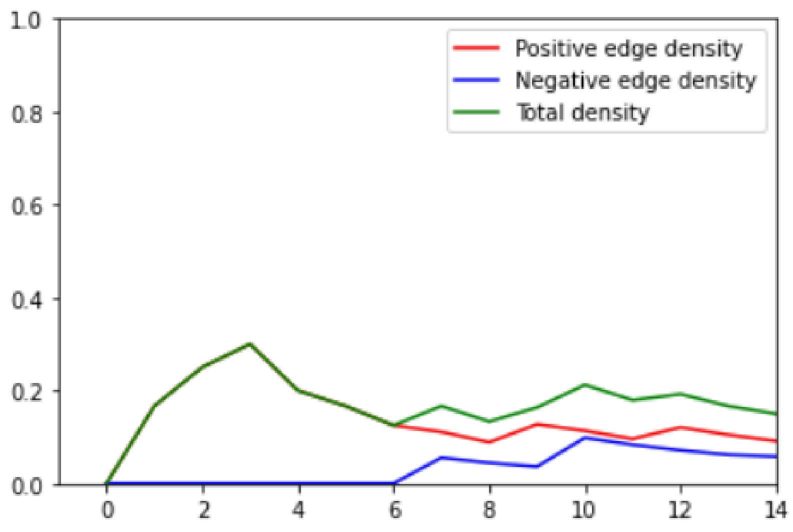
```
[ ] # We want to see the rich club effect in the 100 edges with the highest indegree.
    picked = 100
    rce1 = Stats[['From','Indegree']].sort_values(by=['Indegree'], ascending=False)[0:picked]
    rich = list(rce1.From.values)
    rece_graph = graph.subgraph(rich).copy()
    # Dropping first value

    densities = np.zeros((3,picked))
    pos_edges = 0
    neg_edges = 0
    for rank, node1 in enumerate(rich[1::]):

        for rank2, node2 in enumerate(rich[:rank+1]):
            if graph.has_edge(node1, node2):
                if graph[node1][node2]['weight'] == 1:
                    pos_edges += 1
                else:
                    neg_edges += 1
            if graph.has_edge(node2, node1):
                if graph[node2][node1]['weight'] == 1:
                    pos_edges += 1
                else:
                    neg_edges += 1

        densities[0][rank] = pos_edges/(rank+2)/(rank+1)
        densities[1][rank] = neg_edges/(rank+2)/(rank+1)
        densities[2][rank] = (pos_edges + neg_edges)/(rank+2)/(rank+1)


    plt.plot( [i for i in range(0,picked)],densities[0,],'r-',label='Positive edge density')
    plt.plot( [i for i in range(0,picked)],densities[1,],'b-',label='Negative edge density')
    plt.plot( [i for i in range(0,picked)],densities[2,],'g-',label='Total density')
    plt.legend(loc="upper right")
    plt.ylim((0,1))
    plt.xlim((-2,picked-2))
    plt.show()


    # Checking the one sidedness
    nx.reciprocity(rece_graph)
```



```
0.6515615431720759
```

Negative rich club

```
[ ] # We want to see the rich club effect in the 100 edges with the highest negative_degree.
    picked = 16
    rce2 = Stats[['From','Neg_in']].sort_values(by=['Neg_in'], ascending=False)[4:picked+4]
    rich = list(rce2.From.values)
    # Dropping first value

    densities = np.zeros((3,picked))
    pos_edges = 0
    neg_edges = 0

    for rank, node1 in enumerate(rich[1::]):

        for rank2, node2 in enumerate(rich[:rank+1]):
            if graph.has_edge(node1, node2):
                if graph[node1][node2]['weight'] == 1:
                    pos_edges += 1
                else:
                    neg_edges += 1
            if graph.has_edge(node2, node1):
                if graph[node2][node1]['weight'] == 1:
                    pos_edges += 1
                else:
                    neg_edges += 1

        densities[0][rank] = pos_edges/(rank+2)/(rank+1)
        densities[1][rank] = neg_edges/(rank+2)/(rank+1)
        densities[2][rank] = (pos_edges + neg_edges)/(rank+2)/(rank+1)


    plt.plot( [i for i in range(0,picked)],densities[0,],'r-',label='Positive edge density')
    plt.plot( [i for i in range(0,picked)],densities[1,],'b-',label='Negative edge density')
    plt.plot( [i for i in range(0,picked)],densities[2,],'g-',label='Total density')
    plt.legend(loc="upper right")
    plt.ylim((0,1))
    plt.xlim((-1,picked-2))
    plt.show()
```



Simulating the adoption of a behaviour.

```
[ ] # Let top 200 out of 300 people adopt a certain opinion
    import random
    random.seed(5)
    x=len(Stats)
    op1 = Stats[['From','Pos_in']].sort_values(by=['Pos_in'], ascending=False)[0:300]
    op2 = Stats[['From','Neg_in']].sort_values(by=['Neg_in'], ascending=False)[0:100]
    adopters =  random.sample(list(op1['From']), k=200) + random.sample(list(op2['From']), k=50)
    Stats['Adopt'] = 0
    Stats['Heard'] = 0
    Stats['Adopted'] = 0
    for node in adopters :
        Stats.loc[node-1,'Adopt'] = 1

    Total_adopters = 250
    c = []
    change = True
    threshold = 0.4
    j=0
    while change:
        new_adopters = 0
        for node in graph.nodes :
            if Stats.loc[x-1 if node-1==-1 else node-1,'Heard']== 0:
                Stats.at[x-1 if node-1==-1 else node-1,'Heard'] = j
            if Stats.loc[x-1 if node-1==-1 else node-1,'Adopt'] == 0 and graph.out_degree(node)!= 0:
                trust_ch = 0
                no_out = graph.out_degree(node)
                for outgoing in graph.successors(node):
                    if Stats.loc[outgoing-1,'Adopt'] == 1:
                        trust_ch += graph.get_edge_data(node,outgoing).get('weight')
                if trust_ch / no_out > threshold:
                    Stats.at[x-1 if node-1==-1 else node-1,'Adopt'] = 1
                    Stats.at[x-1 if node-1==-1 else node-1,'Adopted'] = j
                    new_adopters += 1
        c.append(new_adopters)
        j+=1
        print('The number of new adopters at step ' + str(j) + ' is ' + str(new_adopters))
        if new_adopters == 0:
            change = False
```

```
The number of new adopters at step 1 is 8782
The number of new adopters at step 2 is 665
The number of new adopters at step 3 is 229
The number of new adopters at step 4 is 96
The number of new adopters at step 5 is 45
The number of new adopters at step 6 is 94
The number of new adopters at step 7 is 329
The number of new adopters at step 8 is 529
The number of new adopters at step 9 is 1362
The number of new adopters at step 10 is 1548
The number of new adopters at step 11 is 961
The number of new adopters at step 12 is 845
The number of new adopters at step 13 is 906
The number of new adopters at step 14 is 1384
The number of new adopters at step 15 is 1327
The number of new adopters at step 16 is 1283
The number of new adopters at step 17 is 1093
The number of new adopters at step 18 is 1789
The number of new adopters at step 19 is 3048
The number of new adopters at step 20 is 3577
The number of new adopters at step 21 is 3463
The number of new adopters at step 22 is 4769
The number of new adopters at step 23 is 4623
The number of new adopters at step 24 is 4589
The number of new adopters at step 25 is 3474
The number of new adopters at step 26 is 1781
The number of new adopters at step 27 is 1010
The number of new adopters at step 28 is 414
The number of new adopters at step 29 is 130
The number of new adopters at step 30 is 27
The number of new adopters at step 31 is 2
The number of new adopters at step 32 is 0
```

Average time taken to adopt an opinion

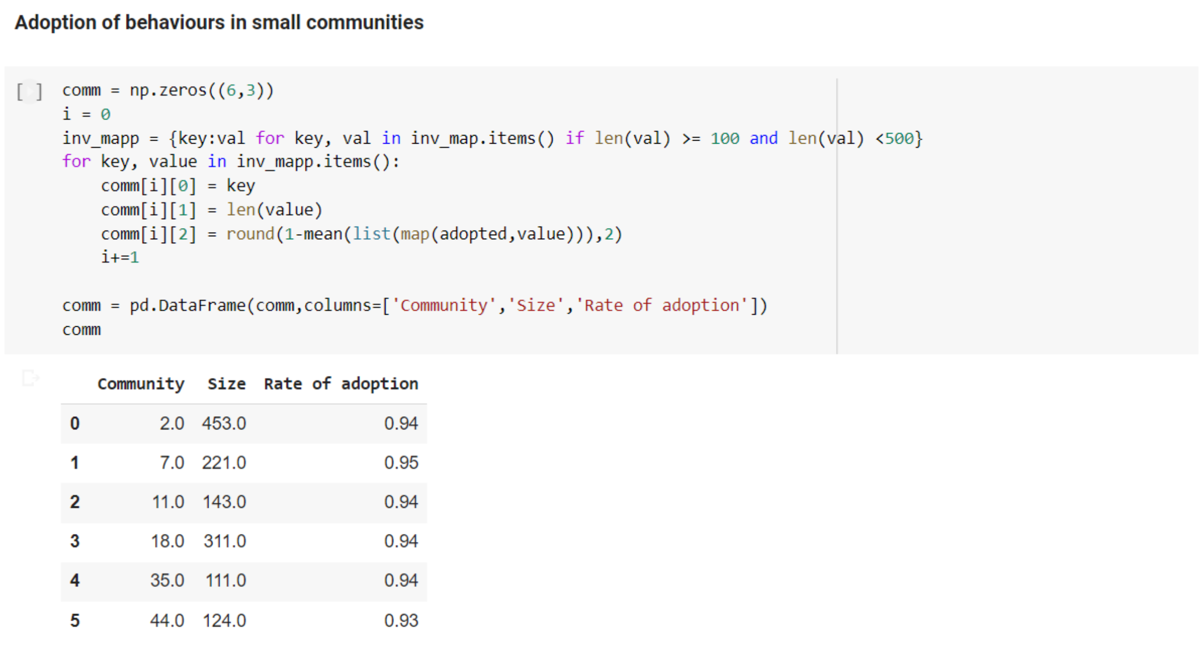
```
[ ]  Times = Stats.loc[Stats['Heard'] != 0]
     Times = Times.loc[Times['Adopted'] != 0]

     print(Times['Adopted'].mean()-Times['Heard'].mean())
```

```
17.403463165315475
```

Adoption of behaviour in the out component

```
[ ]  from statistics import mean

     def adopted(node):
         return int(Stats.loc[node-1,'Adopt'] == 1)

     print(round(1-mean(list(map(adopted,OUT_C))),5))
```

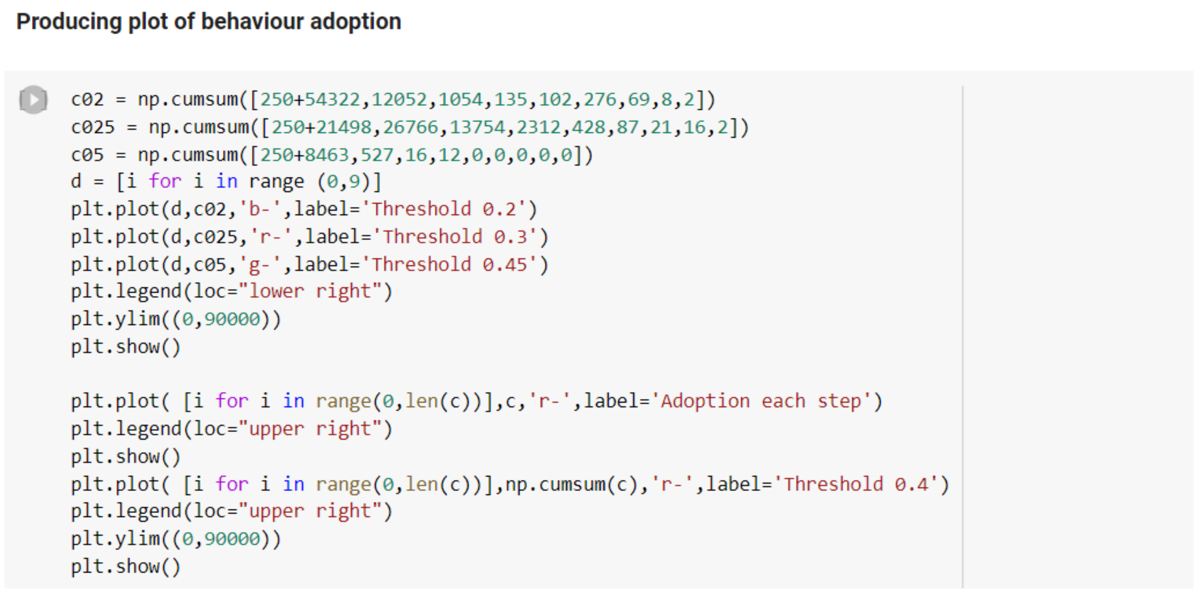
```
0.2718
```



Adoption of behaviours in small communities

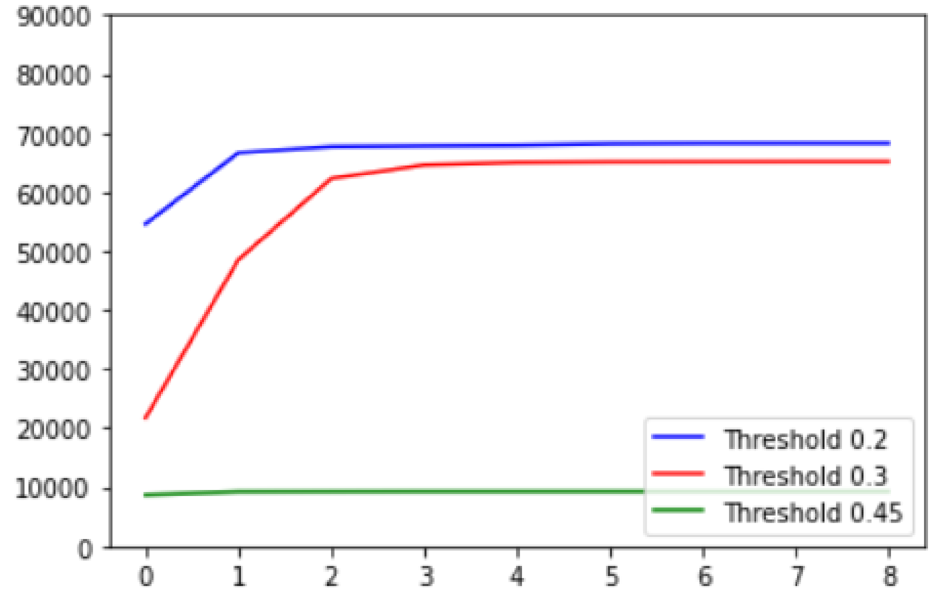
```
[ ]  comm = np.zeros((6,3))
     i = 0
     inv_mapp = {key:val for key, val in inv_map.items() if len(val) >= 100 and len(val) <500}
     for key, value in inv_mapp.items():
         comm[i][0] = key
         comm[i][1] = len(value)
         comm[i][2] = round(1-mean(list(map(adopted,value))),2)
         i+=1

     comm = pd.DataFrame(comm,columns=['Community','Size','Rate of adoption'])
     comm
```

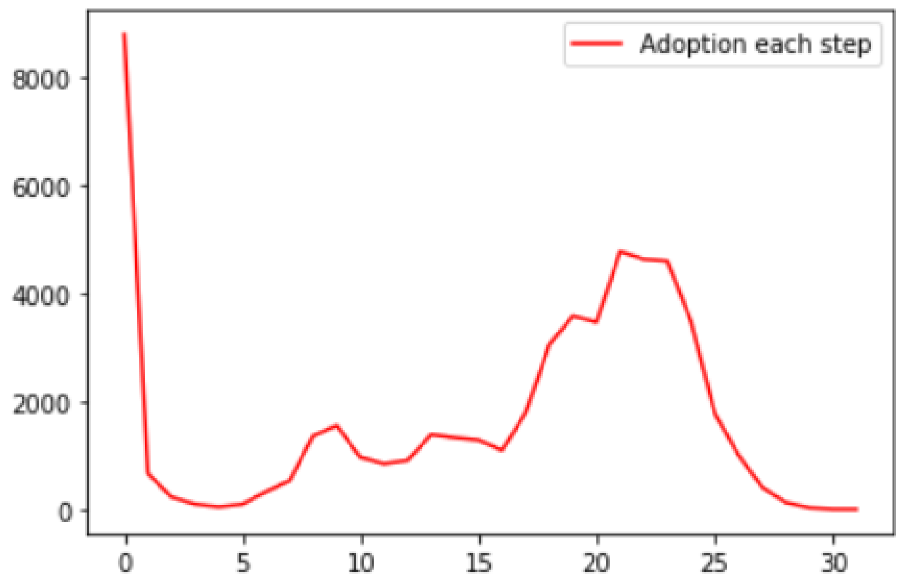
	Community	Size	Rate of adoption
0	2.0	453.0	0.94
1	7.0	221.0	0.95
2	11.0	143.0	0.94
3	18.0	311.0	0.94
4	35.0	111.0	0.94
5	44.0	124.0	0.93

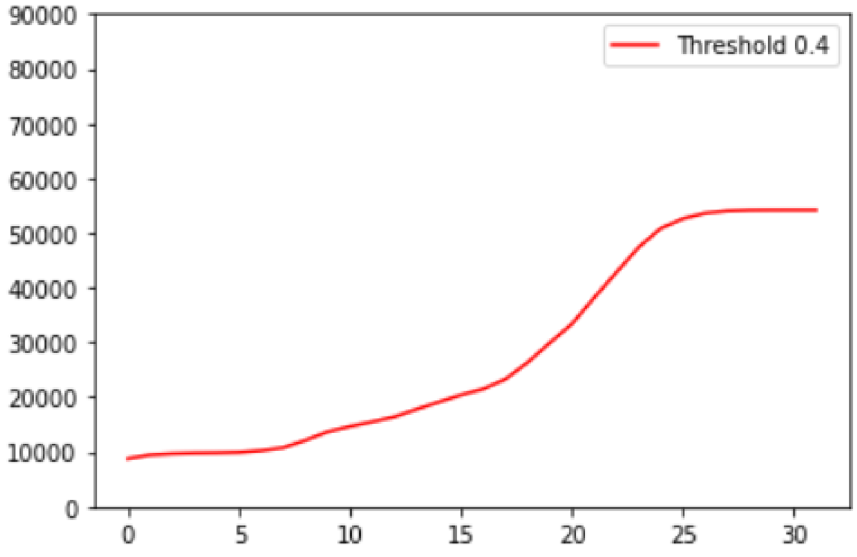


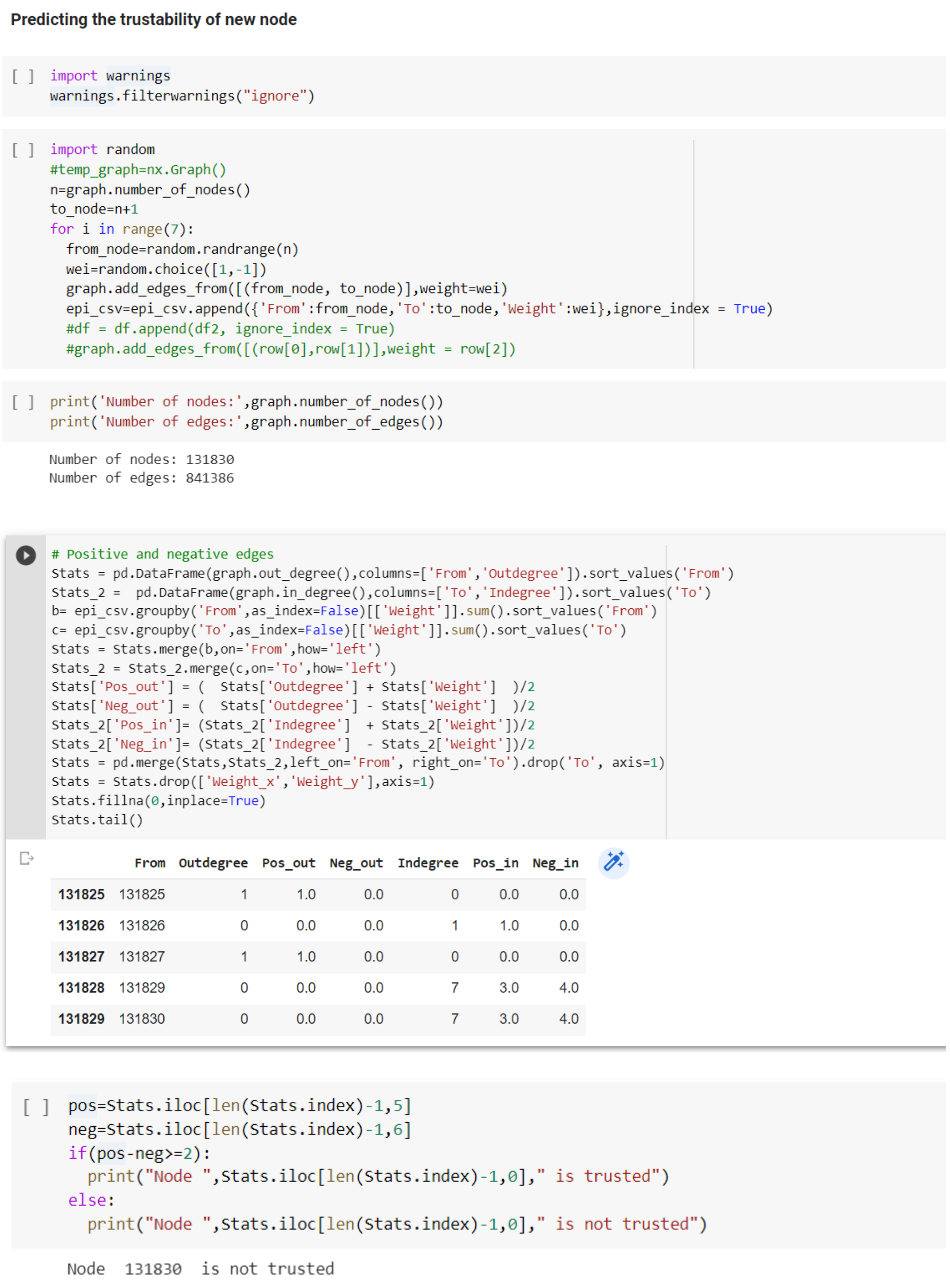
Producing plot of behaviour adoption

```
c02 = np.cumsum([250+54322,12052,1054,135,102,276,69,8,2])
c025 = np.cumsum([250+21498,26766,13754,2312,428,87,21,16,2])
c05 = np.cumsum([250+8463,527,16,12,0,0,0,0,0])
d = [i for i in range (0,9)]
plt.plot(d,c02,'b-',label='Threshold 0.2')
plt.plot(d,c025,'r-',label='Threshold 0.3')
plt.plot(d,c05,'g-',label='Threshold 0.45')
plt.legend(loc="lower right")
plt.ylim((0,90000))
plt.show()

plt.plot( [i for i in range(0,len(c))],c,'r-',label='Adoption each step')
plt.legend(loc="upper right")
plt.show()
plt.plot( [i for i in range(0,len(c))],np.cumsum(c),'r-',label='Threshold 0.4')
plt.legend(loc="upper right")
plt.ylim((0,90000))
plt.show()
```


Threshold 0.2
Threshold 0.3
Threshold 0.45


Adoption each step


Threshold 0.4



Predicting the trustability of new node

```
[ ] import warnings
    warnings.filterwarnings("ignore")
```

```
[ ] import random
    #temp_graph=nx.Graph()
    n=graph.number_of_nodes()
    to_node=n+1
    for i in range(7):
      from_node=random.randrange(n)
      wei=random.choice([1,-1])
      graph.add_edges_from([(from_node, to_node)],weight=wei)
      epi_csv=epi_csv.append({'From':from_node,'To':to_node,'Weight':wei},ignore_index = True)
      #df = df.append(df2, ignore_index = True)
      #graph.add_edges_from([(row[0],row[1])],weight = row[2])
```

```
[ ] print('Number of nodes:',graph.number_of_nodes())
    print('Number of edges:',graph.number_of_edges())
```

```
Number of nodes: 131830
Number of edges: 841386
```

```
# Positive and negative edges
Stats = pd.DataFrame(graph.out_degree(),columns=['From','Outdegree']).sort_values('From')
Stats_2 =  pd.DataFrame(graph.in_degree(),columns=['To','Indegree']).sort_values('To')
b= epi_csv.groupby('From',as_index=False)[['Weight']].sum().sort_values('From')
c= epi_csv.groupby('To',as_index=False)[['Weight']].sum().sort_values('To')
Stats = Stats.merge(b,on='From',how='left')
Stats_2 = Stats_2.merge(c,on='To',how='left')
Stats['Pos_out'] = (  Stats['Outdegree'] + Stats['Weight']  )/2
Stats['Neg_out'] = (  Stats['Outdegree'] - Stats['Weight']  )/2
Stats_2['Pos_in']= (Stats_2['Indegree']  + Stats_2['Weight'])/2
Stats_2['Neg_in']= (Stats_2['Indegree']  - Stats_2['Weight'])/2
Stats = pd.merge(Stats,Stats_2,left_on='From', right_on='To').drop('To', axis=1)
Stats = Stats.drop(['Weight_x','Weight_y'],axis=1)
Stats.fillna(0,inplace=True)
Stats.tail()
```

	From	Outdegree	Pos_out	Neg_out	Indegree	Pos_in	Neg_in
131825	131825	1	1.0	0.0	0	0.0	0.0
131826	131826	0	0.0	0.0	1	1.0	0.0
131827	131827	1	1.0	0.0	0	0.0	0.0
131828	131829	0	0.0	0.0	7	3.0	4.0
131829	131830	0	0.0	0.0	7	3.0	4.0

```
[ ] pos=Stats.iloc[len(Stats.index)-1,5]
    neg=Stats.iloc[len(Stats.index)-1,6]
    if(pos-neg>=2):
      print("Node ",Stats.iloc[len(Stats.index)-1,0]," is trusted")
    else:
      print("Node ",Stats.iloc[len(Stats.index)-1,0]," is not trusted")
```

```
Node  131830  is not trusted
```

4.2 DISCUSSIONS:

Power Law:

While analyzing the trust network, the focus was on the distribution of degrees among nodes that have at least one incoming edge, which serves as a proxy for popularity. The assumption was that this distribution would follow a power law, indicating that there are a small number of highly trusted nodes and a large number of nodes with low trust. This was verified by plotting the log-log plot of the degree distribution and observed linearity, indicating that the assumption of a power law was correct. Then a built-in function was used to check whether the degree distribution was in a scale-free regime, and it was found to have a power-law coefficient of 1.70 as shown in **Figure 2**, indicating that the network is not in a scale-free regime. However, it is to be noted that the degree distribution is still heterogeneous, and the mean carries little statistical significance. On calculating the maximum degree values it was found that a few users account for most of the activity on the site. Repeating the same calculation for the density of outdegrees gave similar results, with a power-law coefficient of 1.72 as shown in **Figure 3**.

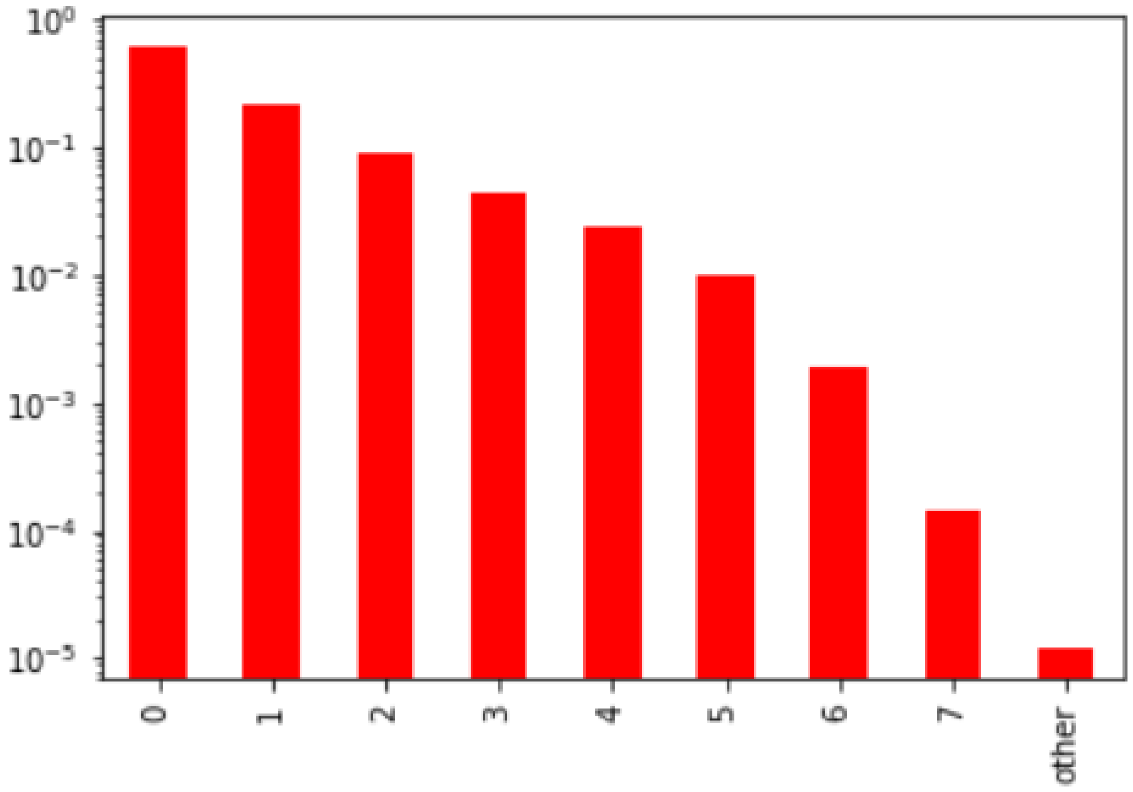


Figure 2 Power Law for In-degree

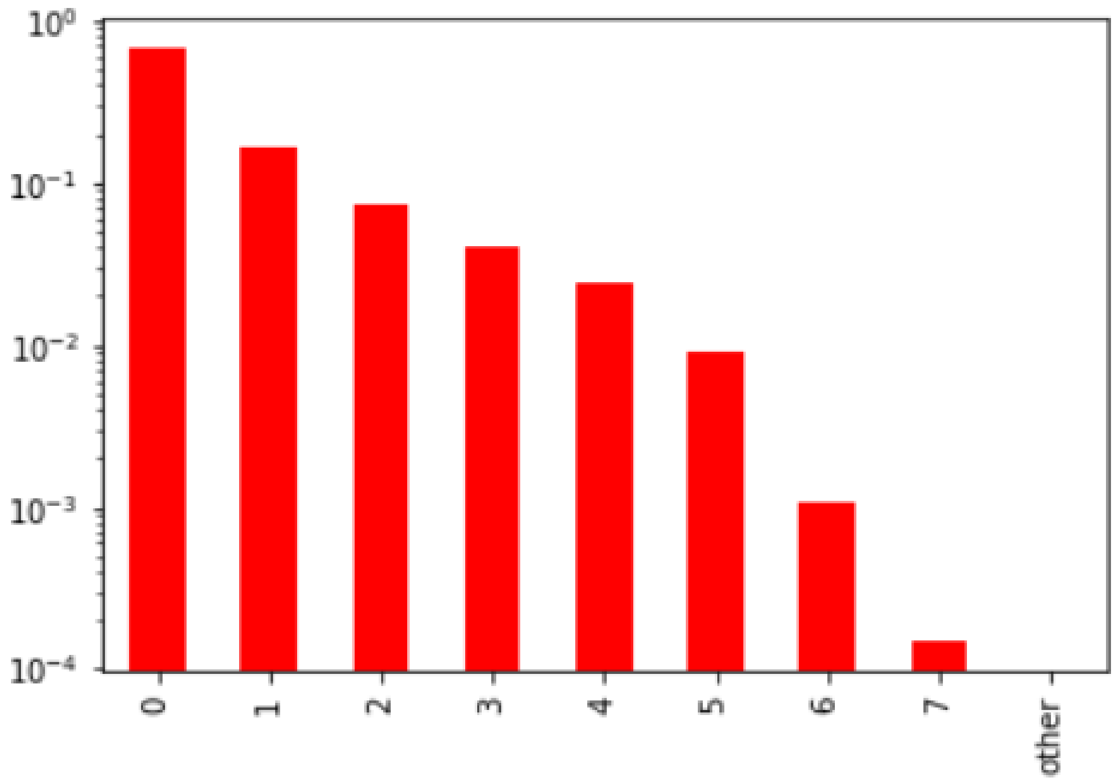


Figure 3 Power Law for Out-degree

Rich Club Effect:

The Objective was to examine the level of interconnectivity among the most trusted edges in the network. Because of the size and the power-law distribution of the network, the analysis was restricted to the top hundred nodes and investigated densities for both positive and negative edges. The results showed no evidence of a rich club effect, as the density remained constant around 0.4 after the top ten edges. Then it was checked whether the connections tended to be one-sided or followed a random pair of nodes, and found that the reciprocity for this subgraph was 63%. Therefore, it can be concluded that popular edges usually aren't connected, but when they are, the following is reciprocal. One of the interpretations could be that these popular nodes are from different groups and therefore don't communicate with each other in the network. It was also observed that some of the most trusted members distrust other highly trusted members.

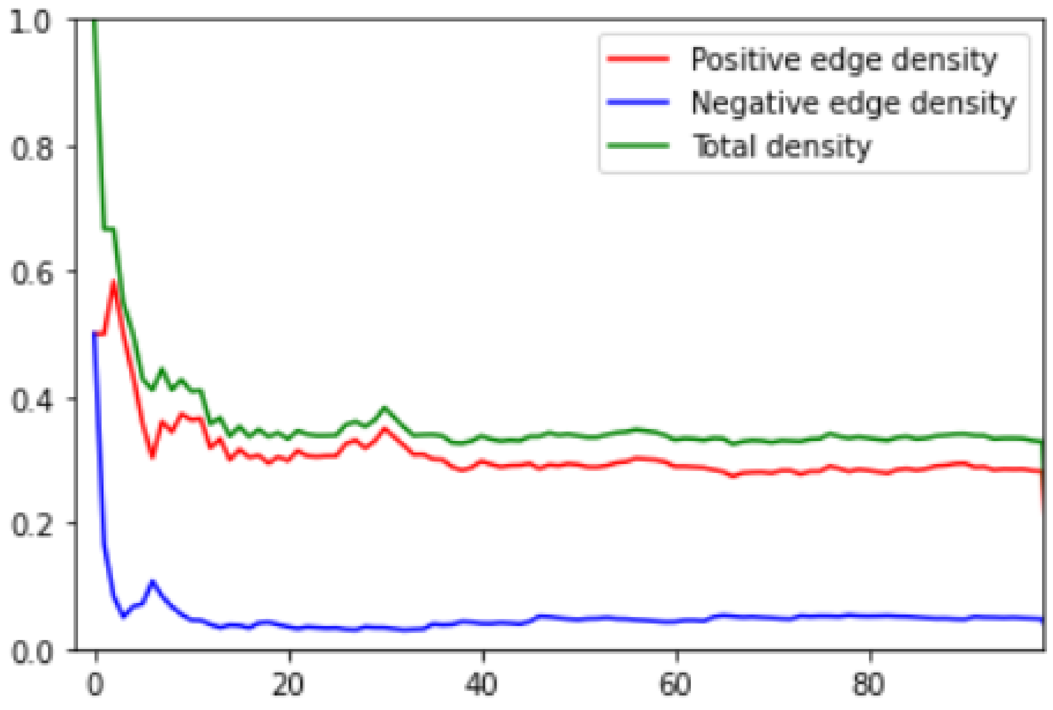


Figure 4: Rich Club Effect

Negative Rich Club Effect:

The possibility of a negative rich club effect was also , which is an uncommon scenario. We looked for interconnections between edges with the highest negative indegree, imagining that we had a group of people with opinions deviating from consensus, who were generally disliked but very popular among their group. By analyzing the rich-club effect among such nodes, we could check the existence of such a group. However, the top four most distrusted vertices were not connected between themselves. After removing them and plotting another graph, we found that there was not a high interconnection between distrusted nodes, and it quickly turned into distrust. Hence, the distrusted members of this site tended to operate like isolated islands.

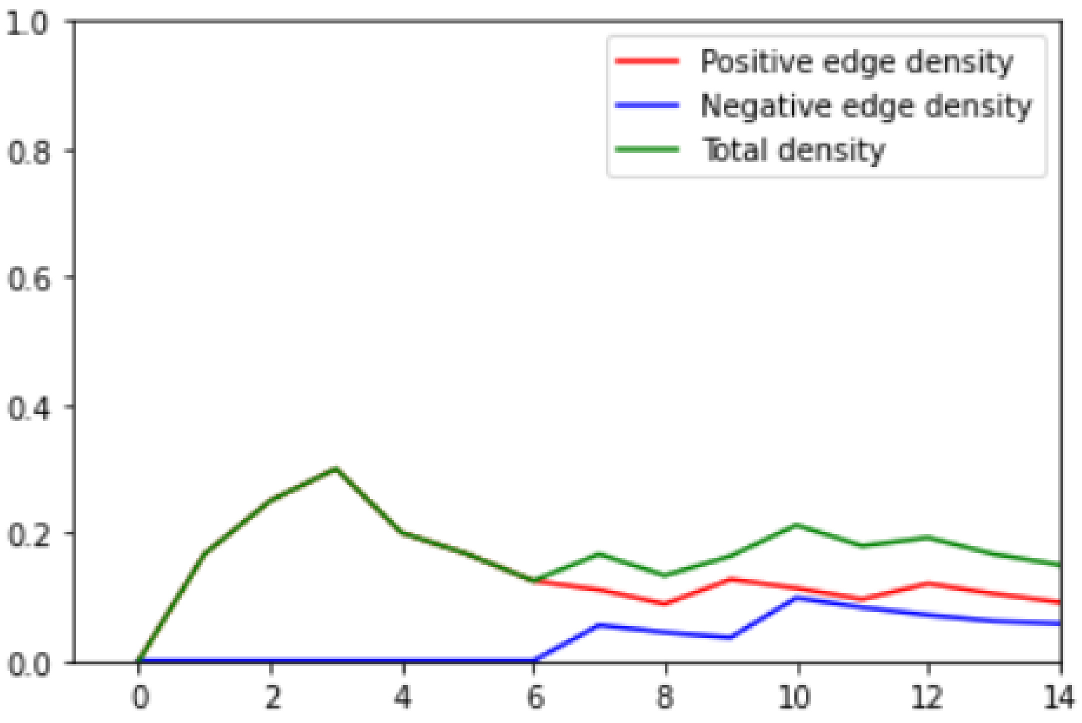


Figure 5: Negative Rich Club Effect

Community Detection:

The graph that was worked with for this study did not contain any prior information regarding the structure of its underlying communities. Therefore, the Louvain algorithm was utilized to identify them. This algorithm is highly efficient, even for large networks. The primary objective was to identify the most central communities, so the underlying undirected graph of the largest SCC was examined. The weights of the edges were disregarded as the distrusted members were still considered part of the community.

The Louvain algorithm yielded a small number of big components and a significant number of small communities. In total, eleven communities were identified that had at least 100 members, with sizes ranging from 10032 to 124. Notably, there were three prominent communities, each occupying around 25% of the largest SCC. Given that this was a vast network, there was a possibility that the algorithm could fail to detect small networks due to the resolution limit. However, the total modularity was 0.43, making the results obtained satisfying.

Table 2: Communities in the Network

Community	**Size**
Community 1	10032
Community 2	10540
Community 3	453
Community 4	11587
Community 5	4309
Community 6	221
Community 7	143
Community 8	945
Community 9	311
Community 10	111
Community 11	124

Bow-Tie Structure:

Examining the connected components of the network provides more information about its structure. It can be seen that the maximum strongly connected component contains around 30% of the total network, with approximately 90% of nodes connected in some way. This component consists of 693,737 edges, representing 0.825% of the total edges. The second-largest weakly connected component has 20 nodes, and the second-largest connected component has 15 nodes.

Table 3: Size of the components

Component	Size
Out Component	37682
In Component	29732
Tendrils	10275

The existence of bow-tie structure in the network was also checked, similar to that of the web. The findings suggest that this is the case, although the proportion of tendrils is much smaller due to the small-world nature of the network.

Reciprocity:

Reciprocity is a significant measure for this network analysis, as it reflects the mutual trust and relationship between the nodes. Reciprocity can be seen as a form of social selection, where individuals tend to trust and interact with those who trust and interact with them. In the context of our graph, the overall reciprocity was found to be 0.31, indicating that a relatively low proportion of nodes reciprocate their connections.

However, when the strongest connected components were examined separately, we found that the reciprocity increased to 0.37. This suggests that the nodes in the central cluster of the network have a higher level of mutual trust and interaction compared to the overall network. The higher reciprocity in the central cluster can be attributed to the fact that these nodes are more likely to be the active and influential members of the network, who are more engaged with each other and have a stronger social tie.

Overall, reciprocity is a crucial measure to understand the dynamics of social networks, and it can provide valuable insights into the social behavior of individuals and groups. By examining the reciprocity level of different parts of the network, we can identify the clusters that have a higher level of mutual trust and interaction, and better understand the factors that shape the social structure of the network.

Degree Pair Plots:

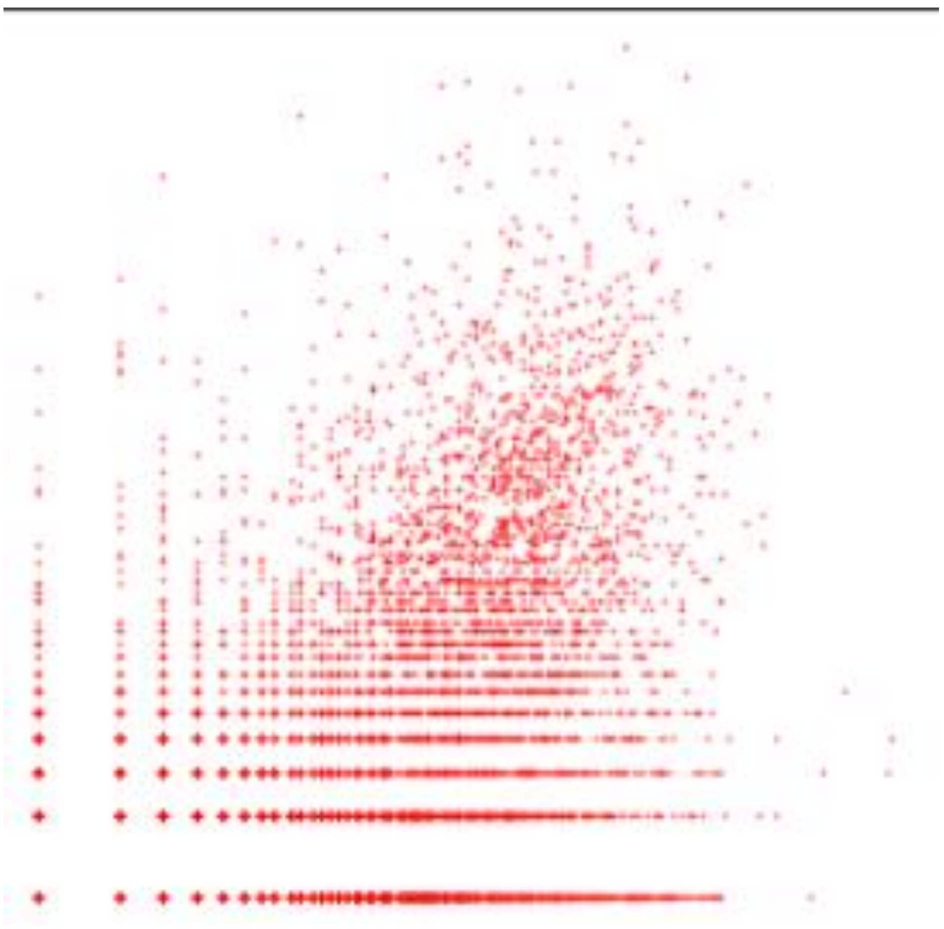


Figure 6: Relationship between positive and negative weighted outgoing edges

It is observed that nodes with higher outdegree tend to have higher indegree on average. However, for nodes with smaller indegrees and outdegrees, there is no noticeable correlation.

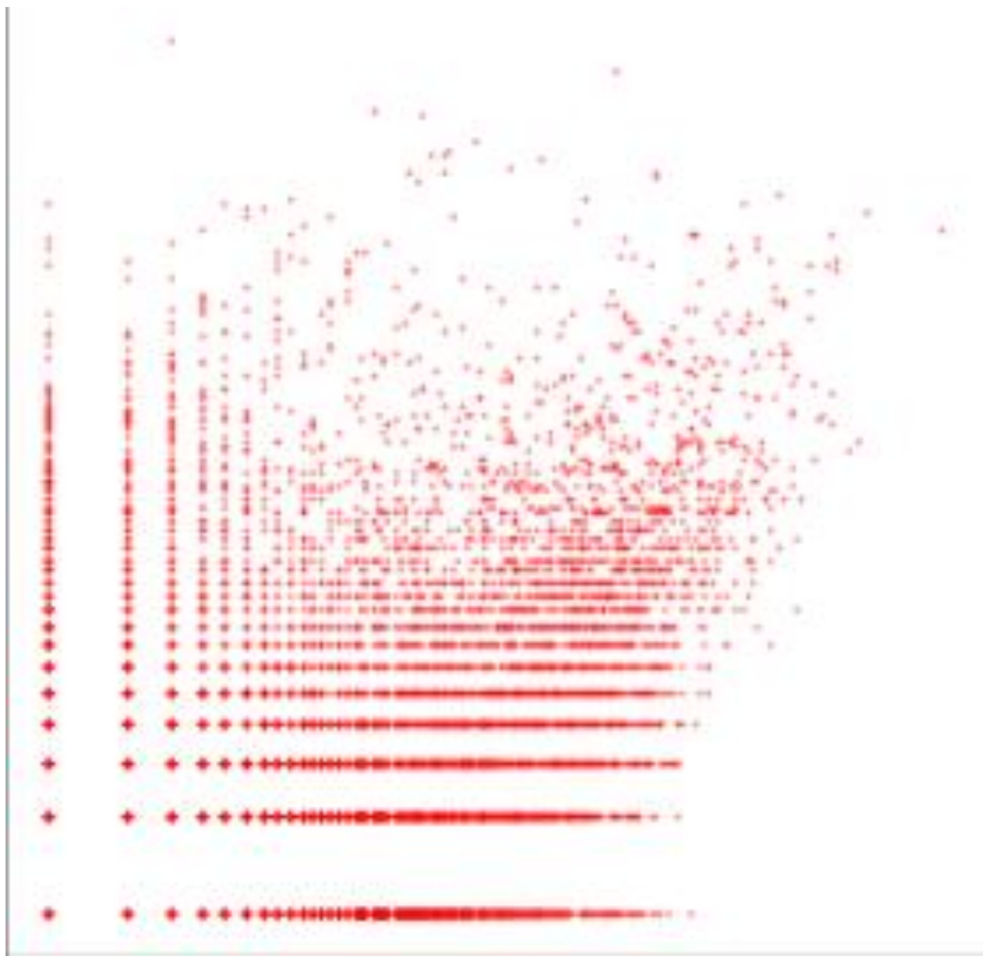


Figure 7: Relationship between positive and negative weighted incoming edges

It can be seen that the most trusted and the most distrusted authors are never the same person. Highly distrusted authors on an average have around 100 ingoing negative edges.

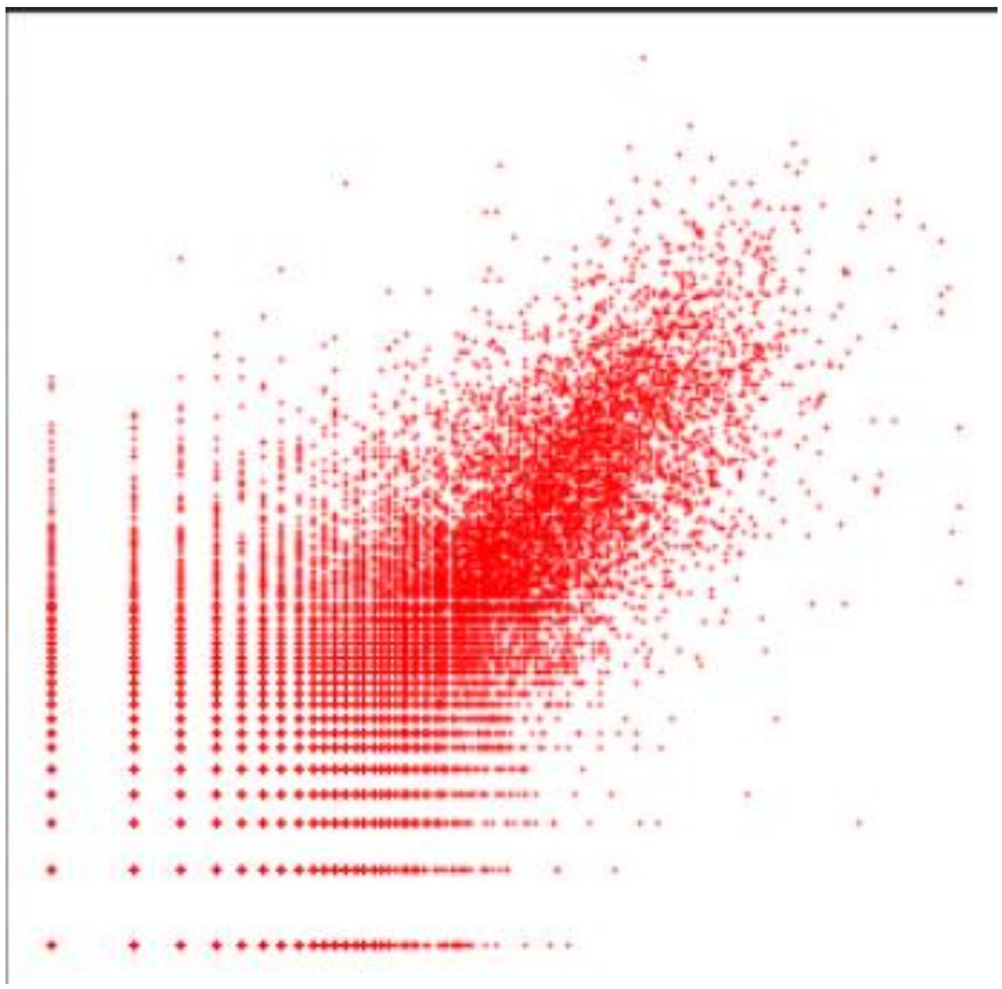


Figure 8: Relationship between incoming and outgoing edges

From the above plot and edge weight distribution it can be noted that a major chunk of the population prefer trusting relations over distrusting ones.

Degree Correlation:

Assortativity refers to the tendency of nodes to connect with other nodes that have a similar level of power or influence, whereas disassortativity refers to the tendency of nodes with higher degrees to connect with nodes that have lower degrees.

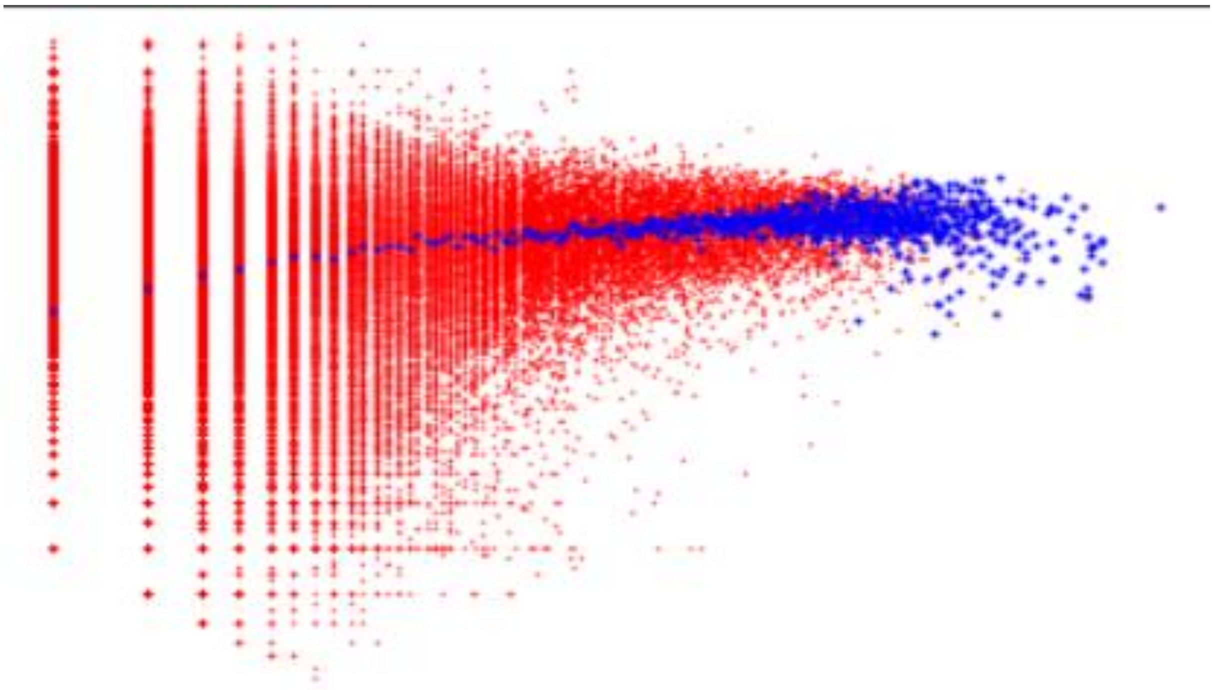


Figure 9: Assortativity coefficient for Outgoing edges

From the above plot, it can be seen that based on outdegree, nodes are mostly assortative.

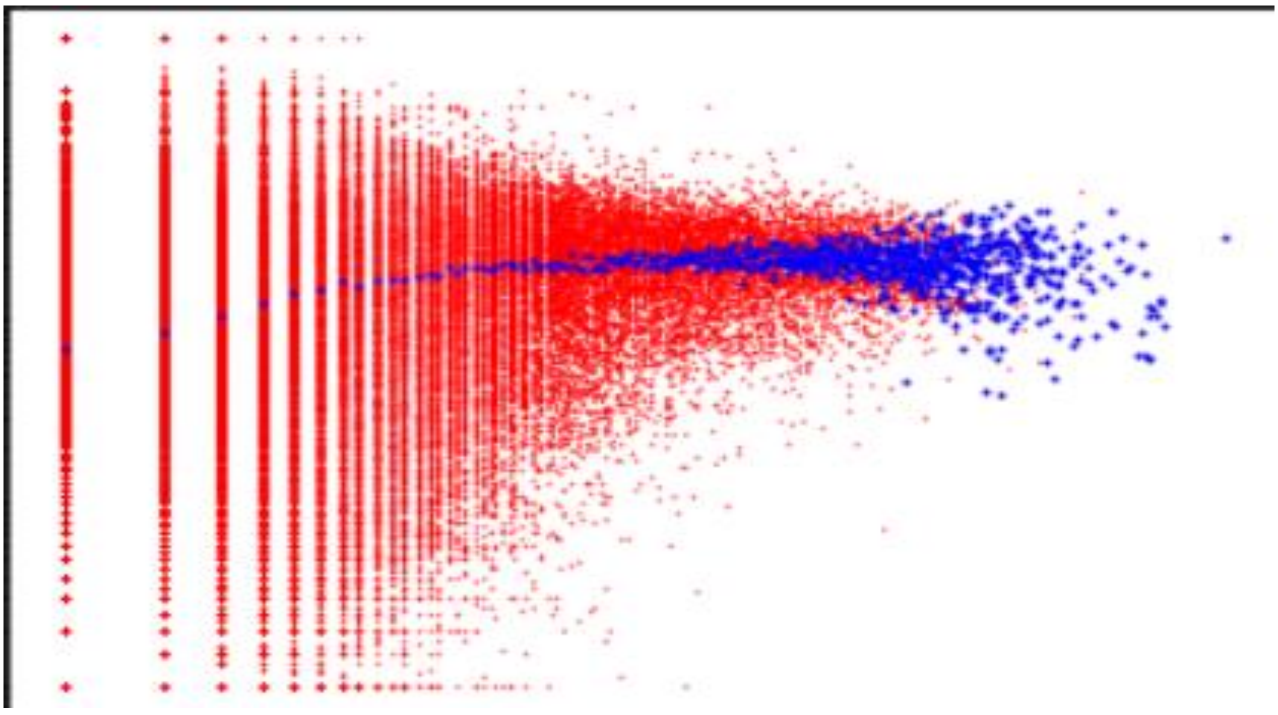


Figure 10: Assortativity coefficient for incoming edges

Nevertheless, it can be observed that the incoming edges exhibit a slight degree of disassortativity. This is expected since nodes with high incoming degrees are more likely to connect to nodes with lower indegrees as there are fewer nodes with similar incoming edges to connect to. The assortativity coefficients for the degree values are 0.0051 and -0.0064, respectively, confirming our initial assumptions.

Adoption Behaviour:

Our focus is on how the adoption of a particular behavior would appear in a network of trusted individuals. As an example, consider a network of doctors who come across a new cure. Initially, a few highly trusted physicians may form a positive opinion about the cure, based on their level of trust and expertise. Similarly, a few less trusted doctors may also adopt the same opinion. However, the network nodes are not aware of each other's popularity. Hence, each node would only adopt a particular viewpoint if a certain percentage of their trusted peers also share the same opinion. Moreover, if a person with a negative reputation adopts a behavior, it would not have the same impact as when a trusted node does. We assume that once a viewpoint is adopted, it is permanent, and there won't be any change. This ensures that the spread of the behavior is always increasing, i.e., monotonic. To achieve our results, we used the same random seed. Our assumptions are that initially, 200 of the 300 most trusted individuals would adopt the opinion, while 50 of the 100 least trusted individuals would also adopt it, even though they may have similar knowledge levels but are distrusted for other reasons.

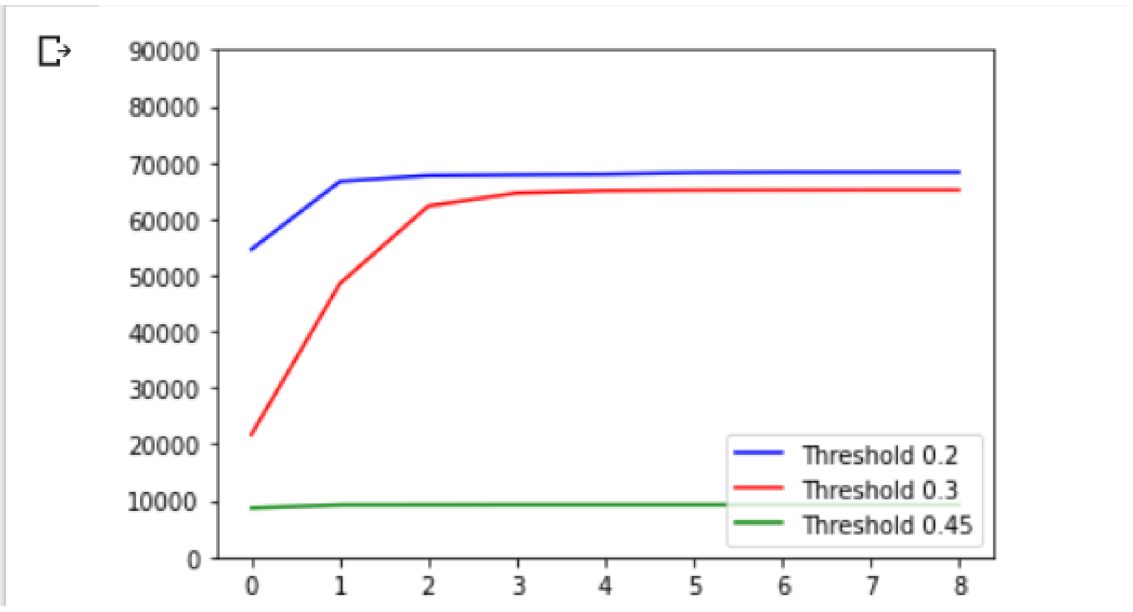


Figure 11: Adoption behaviour for different threshold values

From this plot, it can be seen that the adoption rate of opinion is high for smaller thresholds, while for the larger thresholds, it doesn't have much spread. For more interesting dynamics, by setting the threshold at 0.4 plot was generated.

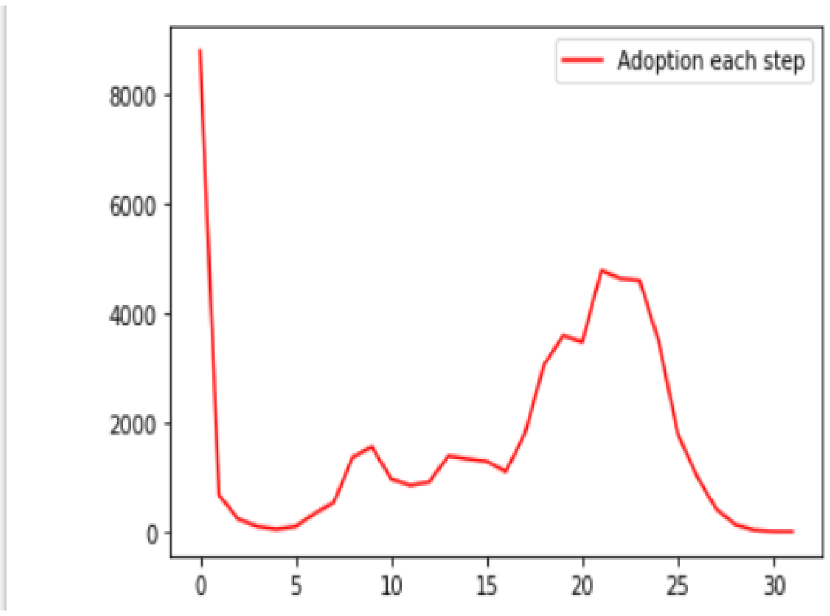


Figure 12: Number of Adopters for each step

It can be observed that after a certain point, there was a slowdown in the adoption of opinions, followed by a subsequent increase. This trend can be attributed to the power-law structure, which facilitates the rapid spread of opinions in the beginning but slows down as the network becomes denser. Once the opinion reached a dense cluster, it took some time to penetrate the border before expanding again. On average, it took 17.12 units of time for the adoption of an opinion, which is half the duration of the entire process. The power-law structure also played a significant role in the initial spread of opinions, as a significant minority of nodes tends to follow only popular nodes, making them more likely to adopt an opinion. In the next step, we will examine the evolution graph.

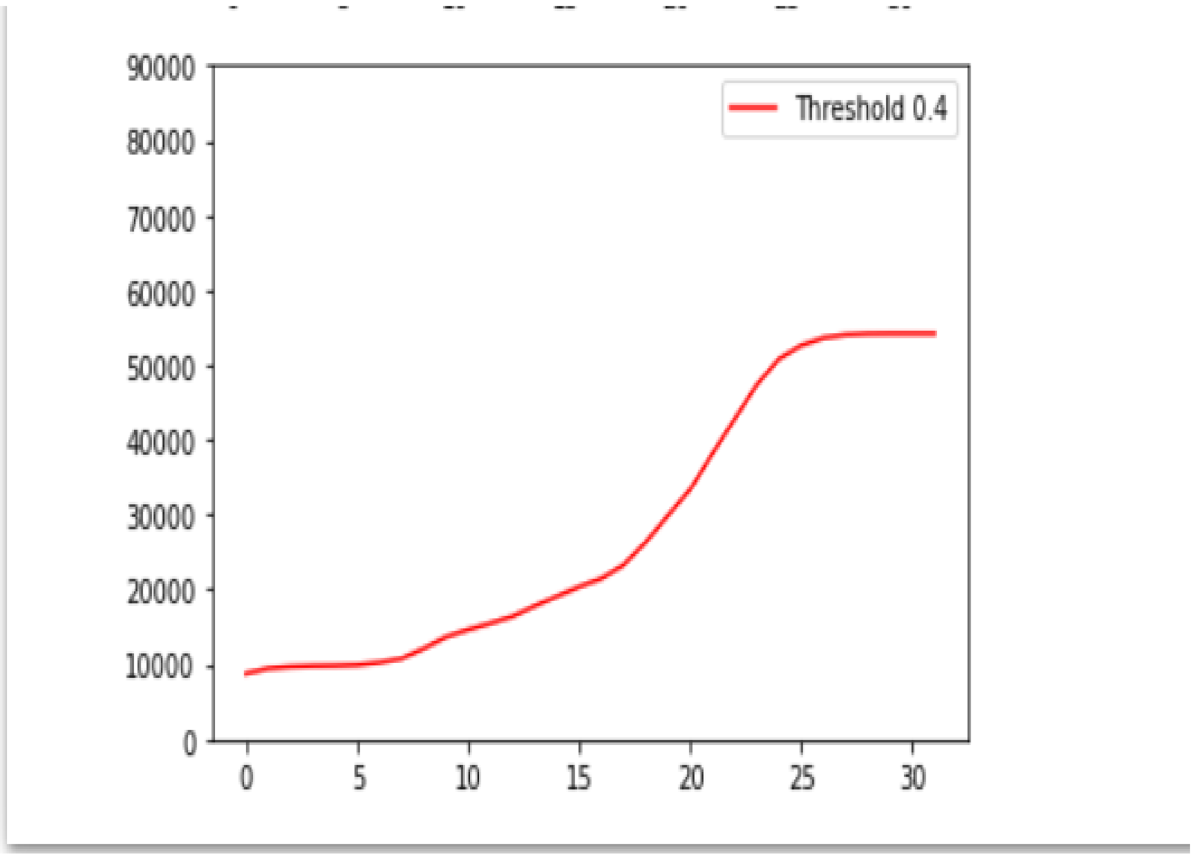


Figure 13: Adoption rate at the threshold value of 0.4

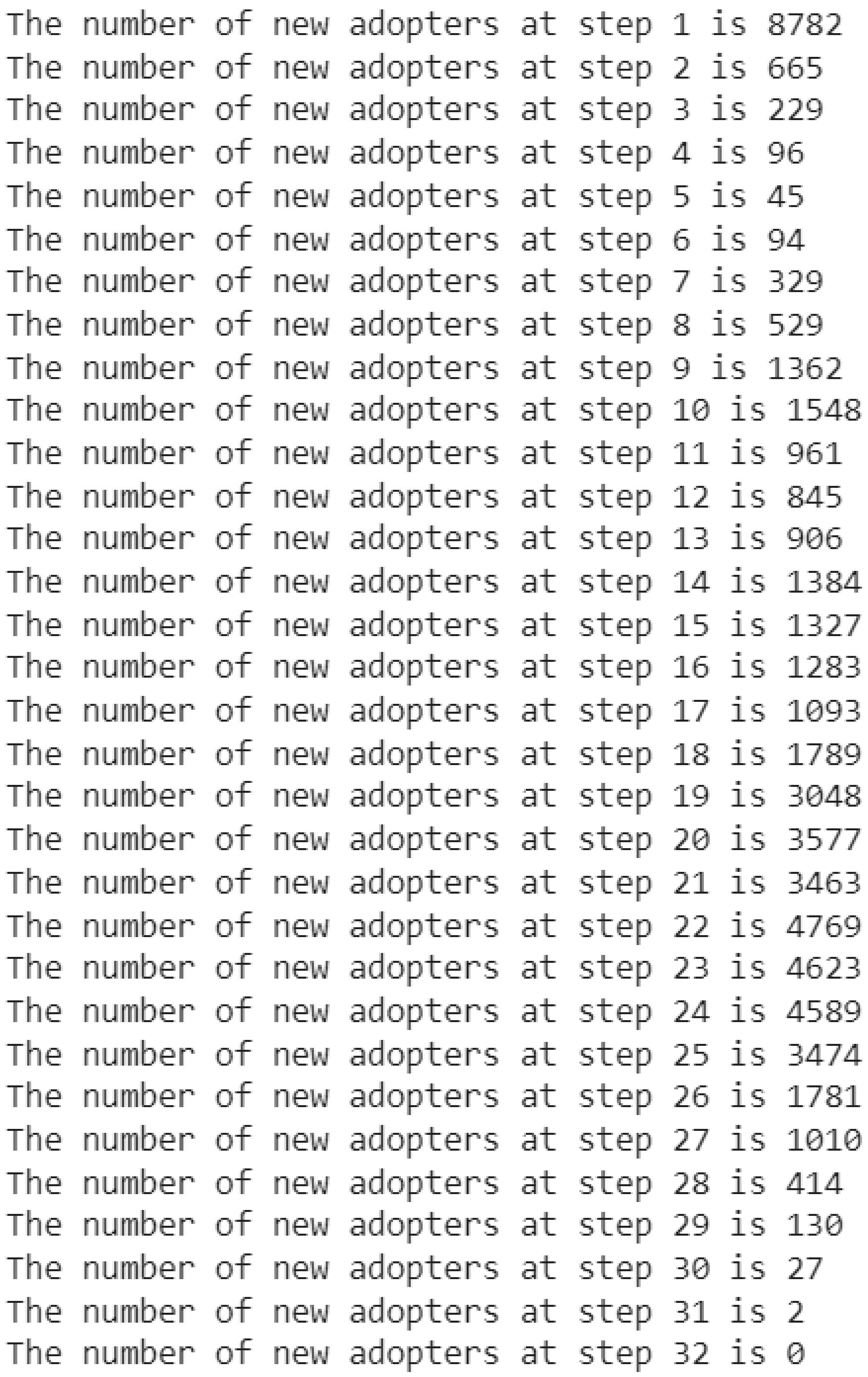
It can be curious to know how other structural properties affect the spread of opinions. It seems reasonable to assume that the opinion would not spread in the OUT component, and the observations confirm this, as only 26% of its members have adopted the new opinion.

Next, how the opinion spread in the found communities and to what extent was investigated. We found six of the smallest communities, each containing at least a hundred nodes.

Table 4: Adoption rate for different sizes

Size	**Adoption rate**
453	0.26
311	0.35
221	0.66
143	0.39
124	0.38
111	0.14

Inside these communities, the presence of dense clusters can be noticed, which hindered the adoption of opinions. This finding further suggests that the dataset contains several small, diverse communities, which is one of the reasons why we did not observe any information cascades. Other factors that contributed to this outcome include nodes with no outdegree, rendering them unable to establish a trusted base of users. Nodes outside the largest weakly connected component were also isolated from the main events, while some nodes had a higher tendency to distrust rather than trust, similar to real-life scenarios where some individuals tend to focus solely on negative aspects.

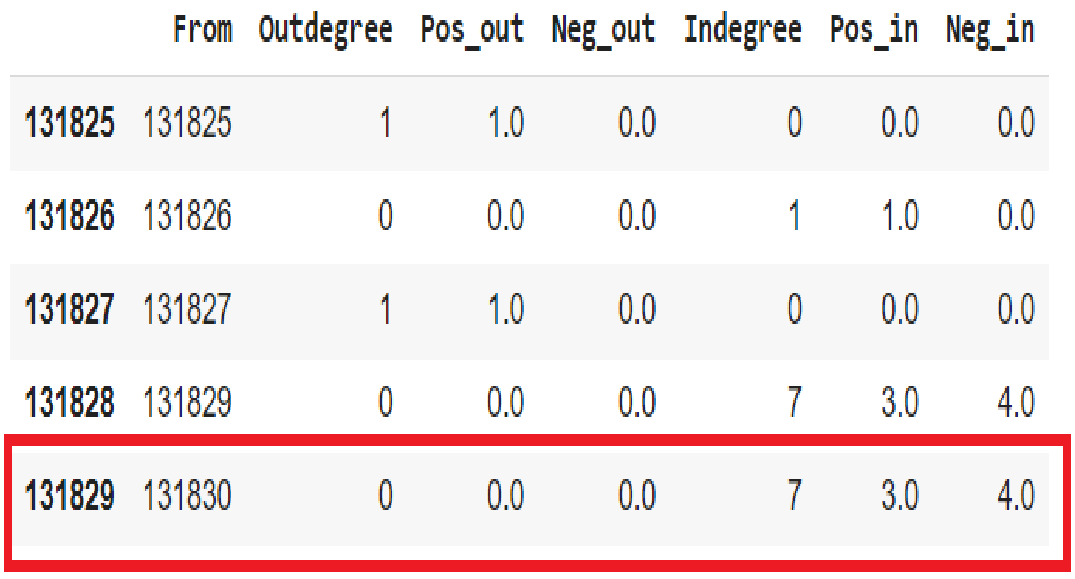


```
The number of new adopters at step 1 is 8782
The number of new adopters at step 2 is 665
The number of new adopters at step 3 is 229
The number of new adopters at step 4 is 96
The number of new adopters at step 5 is 45
The number of new adopters at step 6 is 94
The number of new adopters at step 7 is 329
The number of new adopters at step 8 is 529
The number of new adopters at step 9 is 1362
The number of new adopters at step 10 is 1548
The number of new adopters at step 11 is 961
The number of new adopters at step 12 is 845
The number of new adopters at step 13 is 906
The number of new adopters at step 14 is 1384
The number of new adopters at step 15 is 1327
The number of new adopters at step 16 is 1283
The number of new adopters at step 17 is 1093
The number of new adopters at step 18 is 1789
The number of new adopters at step 19 is 3048
The number of new adopters at step 20 is 3577
The number of new adopters at step 21 is 3463
The number of new adopters at step 22 is 4769
The number of new adopters at step 23 is 4623
The number of new adopters at step 24 is 4589
The number of new adopters at step 25 is 3474
The number of new adopters at step 26 is 1781
The number of new adopters at step 27 is 1010
The number of new adopters at step 28 is 414
The number of new adopters at step 29 is 130
The number of new adopters at step 30 is 27
The number of new adopters at step 31 is 2
The number of new adopters at step 32 is 0
```

Figure 14: Number of Adopters at each step

Predicting the trustability of the new node in the network:

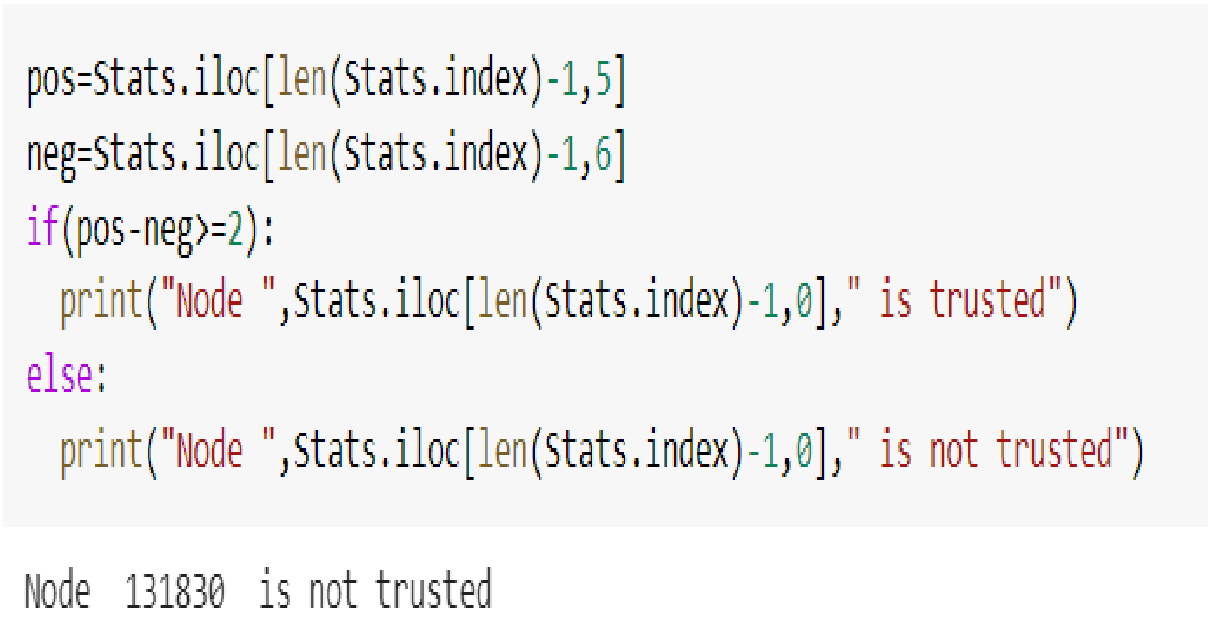
If there is a new node in the network, we need to identify the trustability of the newly arrived node. To classify whether the new node is trusted or not, we need to find the number of positive in-degree and the number of negative in-degree of the node. By using the number of positive in-degree and the number of negative in-degree, we can classify that as trusted node if the difference between the positive and negative in-degree is greater than 2 or else, we will classify that as not trusted node.



	From	Outdegree	Pos_out	Neg_out	Indegree	Pos_in	Neg_in
131825	131825	1	1.0	0.0	0	0.0	0.0
131826	131826	0	0.0	0.0	1	1.0	0.0
131827	131827	1	1.0	0.0	0	0.0	0.0
131828	131829	0	0.0	0.0	7	3.0	4.0
131829	131830	0	0.0	0.0	7	3.0	4.0

Figure 15: In-degree and Out-degree values for new node

From the above Figure, the highlighted part (Node:131830) is the new node in the network. The number of positive in-degree of the node is 3 and the number of negative in-degree of the node is 4. The difference between the positive and negative in-degree is -1 which is clearly less than 2. So, we can classify the new node 131830 as "Not Trusted" which is shown below.



```
pos=Stats.iloc[len(Stats.index)-1,5]
neg=Stats.iloc[len(Stats.index)-1,6]
if(pos-neg>=2):
  print("Node ",Stats.iloc[len(Stats.index)-1,0]," is trusted")
else:
  print("Node ",Stats.iloc[len(Stats.index)-1,0]," is not trusted")
```

```
Node  131830  is not trusted
```

Figure 16: Trustability of the new node

Chapter 5

Conclusion and Future Work

The Epinions network displays typical characteristics of a social network, such as sparsity, power-law structure, and high clustering coefficient. By analyzing pair plots of different degree types and their correlations, we have gained valuable insights. Our analysis has revealed that there is social selection, but we do not observe a rich club effect. We have identified 11 distinct communities and examined behavior adoption within the smallest of these communities. As trust is based on the quality of information received, we have compared the network structure to that of the general web and found them to be similar. Additionally, we have studied adoption behavior within the OUT community and discovered that it is mostly isolated from outside influence. Our investigation of structural balance has shown that this network is approximately balanced and that this balance becomes even stronger in triads with reciprocated edges. Lastly, simulations of opinion adoption have demonstrated how the network reacts to the adoption of opinion with different thresholds and how it spreads through components and communities.

The reviews on Epinions website may contain sentiment that could impact the trust and credibility of the reviewer. Performing sentiment analysis on the reviews could provide additional insights into the behavior of users and how it impacts the trust network. Moreover, the trust network may change over time, and analyzing the temporal dynamics could reveal interesting insights into the evolution of communities and the adoption of opinions. This could involve analyzing the network at different time intervals or using dynamic network analysis techniques.

REFERENCES

[1] Leskovec, J., Huttenlocher, D., & Kleinberg, J. (2010, April). Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 1361-1370).

[2] Brzozowski, M. J., Hogg, T., & Szabo, G. (2008, April). Friends and foes: ideological social networking. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 817-820).

[3] Adamic, L. A., Lukose, R. M., Puniyani, A. R., & Huberman, B. A. (2001). Search in power-law networks. *Physical review E, 64*(4), 046135.

[4] Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review*, *29*(4), 251-262.

[5] Muchnik, L., Pei, S., Parra, L. C., Reis, S. D., Andrade Jr, J. S., Havlin, S., & Makse, H. A. (2013). Origins of power-law degree distribution in the heterogeneity of human activity in social networks. *Scientific reports*, *3*(1), 1783.

[6] Ou, Q., Jin, Y. D., Zhou, T., Wang, B. H., & Yin, B. Q. (2007). Power-law strength-degree correlation from resource-allocation dynamics on weighted networks. *Physical Review E, 75*(2), 021102.

[7] Opsahl, T., Colizza, V., Panzarasa, P., & Ramasco, J. J. (2008). Prominence and control: the weighted rich-club effect. *Physical review letters*, *101*(16), 168702.

[8] Senden, M., Deco, G., De Reus, M. A., Goebel, R., & Van Den Heuvel, M. P. (2014). Rich club organization supports a diverse set of functional network configurations. *Neuroimage*, *96*, 174-182.

[9] Van Den Heuvel, M. P., & Sporns, O. (2011). Rich-club organization of the human connectome. *Journal of Neuroscience, 31*(44), 15775-15786.

[10] Wang, B., Zhan, Q., Yan, T., Imtiaz, S., Xiang, J., Niu, Y., ... & Li, D. (2019). Hemisphere and gender differences in the rich-club organization of structural networks. *Cerebral Cortex, 29*(11), 4889-4901.

[11] Lei, Z., Chen, Y., & Lim, M. K. (2021). Modelling and analysis of big data platform group adoption behaviour based on social network analysis. *Technology in Society*, *65*, 101570.

[12] Serrano Fuentes, N., Rogers, A., & Portillo, M. C. (2019). Social network influences and the adoption of obesity-related behaviours in adults: a critical interpretative synthesis review. *BMC Public Health*, *19*, 1-20.

[13] Shih, Y. Y., & Fang, K. (2006). Effects of network quality attributes on customer adoption intentions of internet banking. *Total Quality Management & Business Excellence*, *17*(1), 61-77.

[14] Carlos Martins Rodrigues Pinho, J., & Soares, A. M. (2011). Examining the technology acceptance model in the adoption of social networks. *Journal of research in Interactive Marketing*, *5*(2/3), 116-129.

[15] Bedi, P., & Sharma, C. (2016). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(3), 115-135.

[16] Xie, J., & Szymanski, B. K. (2012). Towards linear time overlapping community detection in social networks. In *Advances in Knowledge Discovery and Data Mining: 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29–June 1, 2012, Proceedings, Part II 16* (pp. 25-36). Springer Berlin Heidelberg.

[17] Wang, M., Wang, C., Yu, J. X., & Zhang, J. (2015). Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment*, *8*(10), 998-1009.

[18] Fani, H., & Bagheri, E. (2017). Community detection in social networks. *Encyclopedia with semantic computing and robotic intelligence*, *1*(01), 1630001.

[19] Mattie, H., Engø-Monsen, K., Ling, R., & Onnela, J. P. (2018). Understanding tie strength in social networks using a local "bow tie" framework. *Scientific reports*, *8*(1), 1-9.

[20] Yang, R., Zhuhadar, L., & Nasraoui, O. (2011, July). Bow-tie decomposition in directed graphs. In *14th International Conference on Information Fusion* (pp. 1-5). IEEE.

[21] Fisher, D. N., Silk, M. J., & Franks, D. W. (2017). The perceived assortativity of social networks: methodological problems and solutions. *Trends in Social Network Analysis: Information Propagation, User Behavior Modeling, Forecasting, and Vulnerability Assessment*, 1-19.

[22] Newman, M. E., & Park, J. (2003). Why social networks are different from other types of networks. *Physical review E*, *68*(3), 036122.

[23] Shizuka, D., & Farine, D. R. (2016). Measuring the robustness of network community structure using assortativity. *Animal Behaviour*, *112*, 237-246.

[24] Hagberg, A., Swart, P., & S Chult, D. (2008). *Exploring network structure, dynamics, and function using NetworkX* (No. LA-UR-08-05495; LA-UR-08-5495). Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[25] Hagberg, A., & Conway, D. (2020). Networkx: Network analysis with python. *URL: https://networkx. github. io*.