# Topic: Network Analytics

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Vishvash Batch ID:** 23012024

**Topic: Network Analytics**

**Guidelines:**

**1. An assignment submission is considered complete only when correct and executable code(s) are submitted along with the documentation explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered as correct submission.**

**2. Ensure that you submit your assignments correctly and in full. Resubmission is not allowed.**

**3. Post the submission you can evaluate your work by referring to keys provided. (will be available only post the submission).**

**Hints:**

1. **Business Problem**
   1. **What is the business objective?**
   2. **Are there any constraints?**
2. **Work on each feature of the dataset to create a data dictionary as displayed in the below image:**



**Make a table as shown above and provide information about the features such as its data type and its relevance to the model building. And if not relevant, provide reasons and a description of the feature.**

1. **Data Cleaning**
2. **Model Building**
   1. **Perform network analytics on the given datasets.**
   2. **Briefly explain the model output in the documentation.**
3. **Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**Problem Statement: -**

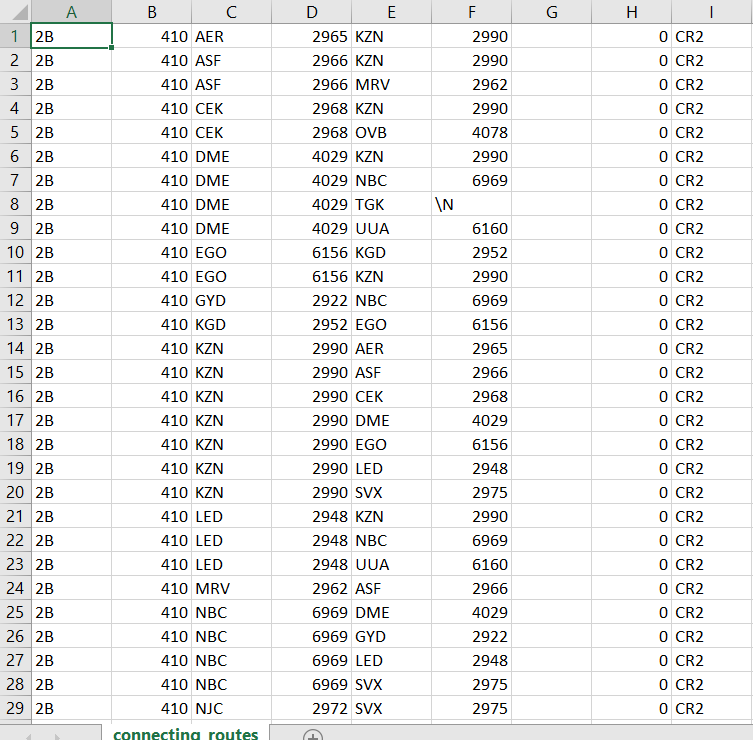
Two datasets consisting of information for the connecting routes and flight halt are provided. Create network analytics models on both datasets separately. Using various network analytics-based measures derive insights for the business to benefit from the data available.

* Create a network using an edge list matrix (directed only**).**
* Column to be used for respective datasets

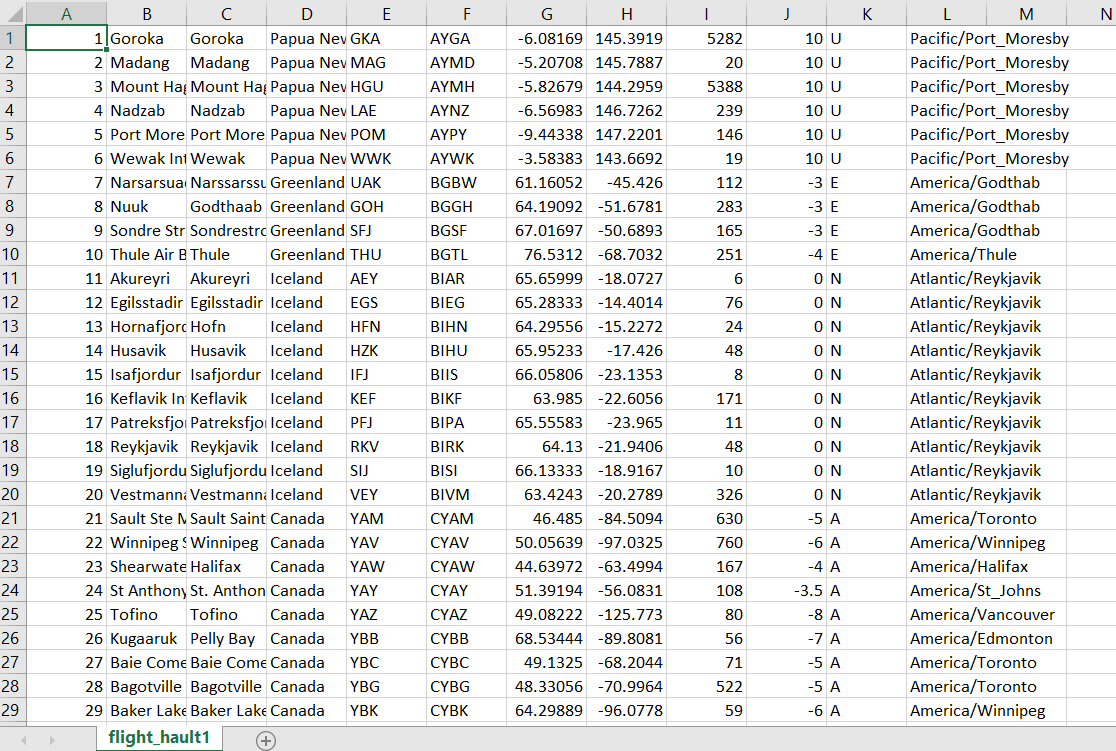
Flight\_halt = ID", "Name", "City", "Country", "IATA\_FAA", "ICAO", "Latitude", "Longitude", "Altitude", "Time", "DST", "Tz database time"

connecting routes = "flights", " ID", "main Airport”, “main Airport ID", "Destination ", "DestinationID", "haults", "machinery"

**cohaltsng routes**



**Flight\_hault1**



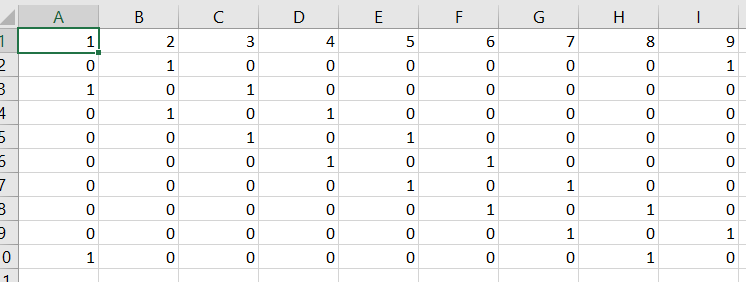
**Problem statement 2:**

There are three datasets given (Facebook, Instagram, and LinkedIn). Construct and visualize the following networks:

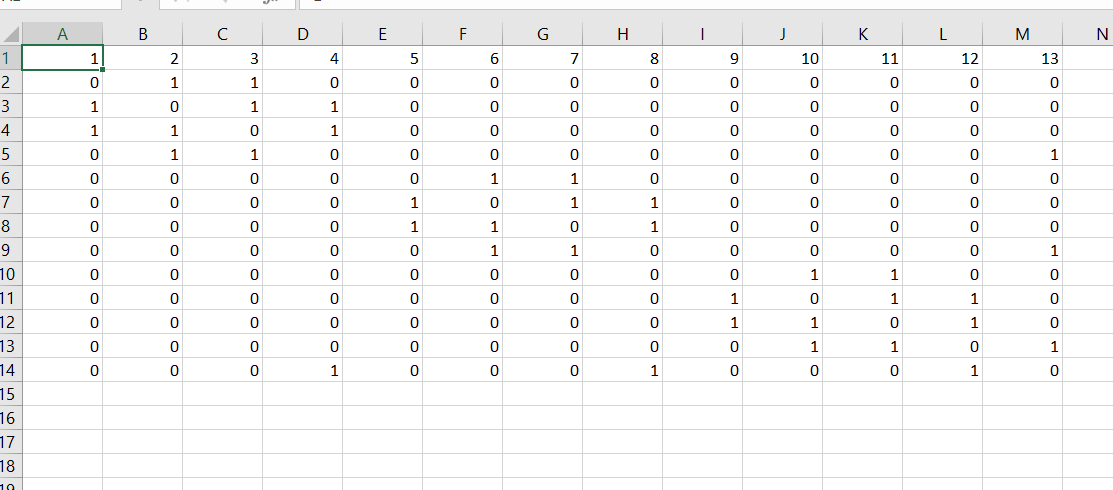
* circular network for Facebook
* star network for Instagram
* star network for LinkedIn

Create a network using an adjacency matrix (undirected only). The snapshots of those datasets are given below:

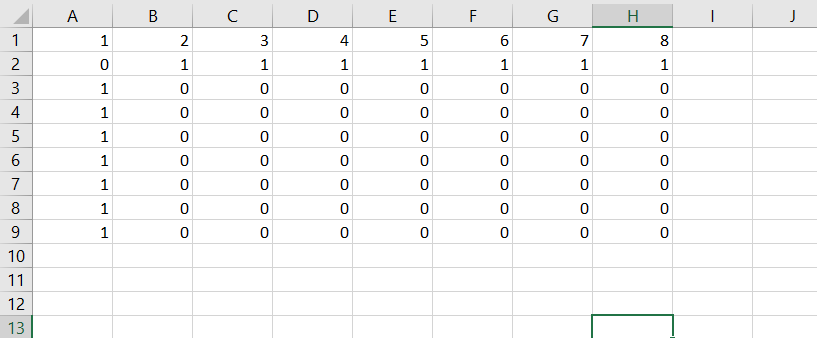
**Facebook**

****

**Instagram**

****

**LinkedIn**



|  |  |  |  |
| --- | --- | --- | --- |
| Name of Feature | Description | Type | Relevance |
| airline | Name of the airline | Nominal | Irrelevant |
| airline ID | Unique identifier for the airline | Nominal | Irrelevant |
| source airport | Name of the departure airport | Nominal | Relevant |
| source airport ID | Unique identifier for the departure airport | Nominal | Irrelevant |
| destination airport | Name of the destination airport | Nominal | Relevant |
| destination airport ID | Unique identifier for the destination airport | Nominal | Irrelevant |
| codeshare | Indicates if the flight is a codeshare | Binary (Yes/No) | Irrelevant |
| stops | Number of stops during the flight | Quantitative | Irrelevant |
| equipment | Equipment used for the flight | Nominal | Irrelevant |

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Feature | Description | Type | Relevance |
| ID | Unique identifier | Nominal | Irrelevant |
| Name | Name of the entity (e.g., airport name) | Nominal | Irrelevant |
| City | City where the entity is located | Nominal | Irrelevant |
| Country | Country where the entity is located | Nominal | Relevant |
| IATA\_FAA | IATA code or FAA code | Nominal | Irrelevant |
| ICAO | ICAO code | Nominal | Irrelevant |
| Latitude | Latitude coordinates of the entity | Quantitative | Irrelevant |
| Longitude | Longitude coordinates of the entity | Quantitative | Irrelevant |
| Altitude | Altitude of the entity | Quantitative | Irrelevant |
| Time | Time zone offset from GMT/UTC | Quantitative | Irrelevant |
| DST | Daylight Saving Time observation | Binary (Yes/No) | Irrelevant |
| Tz database time | Time zone in the TZ database format | Nominal | Irrelevant |

1.

'''

CRISP-ML(Q) process model describes six phases:

- Business and Data Understanding

- Data Preparation

- Model Building

- Model Evaluation and Hyperparameter Tuning

- Model Deployment

- Monitoring and Maintenance

Business Problem: There is a dataset consisting of information for the connecting routes. Create network analytics models on the dataset and measure degree centrality, degree of closeness centrality, and degree of in-between centrality.

Create a network using edge list matrix(directed only).

Objective(s): Maximize profitable route

Constraint(s): Minimize the transport cost

Success Criteria:

a. Business: Increase the Number of tickets booking by 10% to 15%

b. ML:

c. Economic: Additional revenue of $100K to $120K

Features:

connecting routes=c("flights", " ID", "main Airport”, “main Airport ID", "Destination ","Destination ID","haults","machinary")

'''

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

from sqlalchemy import create\_engine, text

# Creating engine which link to SQL via python

engine = create\_engine("mysql+pymysql://{user}:{pw}@localhost/{db}"

.format(user="root",# user

pw="1234", # passwrd

db="air\_routes\_db")) #database

# Reading data from loacal drive

connecting\_routes = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Network Analytics/Network Analytics/connecting\_routes.csv")

connecting\_routes.head()

# Loading data into sql database

connecting\_routes.to\_sql('connecting\_routes', con = engine, if\_exists = 'replace', chunksize = 1000, index= False)

# Reading data from sql database

sql = 'select \* from connecting\_routes;'

connecting\_routes = pd.read\_sql\_query(text(sql), con = engine.connect())

connecting\_routes.head()

connecting\_routes = connecting\_routes.iloc[0:51, 1:8]

connecting\_routes.columns

for\_g = nx.Graph()

for\_g = nx.from\_pandas\_edgelist(connecting\_routes, source = 'source airport',

target = 'destination apirport')

print(for\_g)

# # centrality:-

#

#

# \*\*Degree centrality\*\* is defined as the number of links incident upon a node (i.e., the number of ties that a node has). ... Indegree is a count of the number of ties directed to the node (head endpoints) and outdegree is the number of ties that the node directs to others (tail endpoints).

#

# \*\*Eigenvector Centrality\*\* The adjacency matrix allows the connectivity of a node to be expressed in matrix form. So, for non-directed networks, the matrix is symmetric.Eigenvector centrality uses this matrix to compute its largest, most unique eigenvalues.

#

# \*\*Closeness Centrality\*\* An interpretation of this metric, Centralness.

#

# \*\*Betweenness centrality\*\* This metric revolves around the idea of counting the number of times a node acts as a bridge.

data = pd.DataFrame({"Degree": pd.Series(nx.degree\_centrality(for\_g)),

"closeness":pd.Series(nx.closeness\_centrality(for\_g)),

"eigenvector": pd.Series(nx.eigenvector\_centrality(for\_g)),

"betweenness": pd.Series(nx.betweenness\_centrality(for\_g))})

data

# Visual Representation of the Network

connecting\_routes1 = connecting\_routes.iloc[0:51, 0:8]

for\_g = nx.Graph()

for\_g = nx.from\_pandas\_edgelist(connecting\_routes1, source = 'source airport',

target = 'destination apirport')

f = plt.figure()

pos = nx.spring\_layout(for\_g, k = 0.015)

nx.draw\_networkx(for\_g, pos, ax=f.add\_subplot(111), node\_size = 15, node\_color = 'red')

plt.show()

#f.savefig("graph.png")

**Output:**

connecting\_routes.head()

Out[114]:

airline airline ID source airport ... codeshare stops equipment

0 2B 410 AER ... None 0 CR2

1 2B 410 ASF ... None 0 CR2

2 2B 410 ASF ... None 0 CR2

3 2B 410 CEK ... None 0 CR2

4 2B 410 CEK ... None 0 CR2

Index(['airline ID', 'source airport', 'source airport id',

'destination apirport', 'destination airport id', 'codeshare', 'stops'],

dtype='object')

print(for\_g)

Graph with 22 nodes and 27 edges

data

Out[125]:

Degree closeness eigenvector betweenness

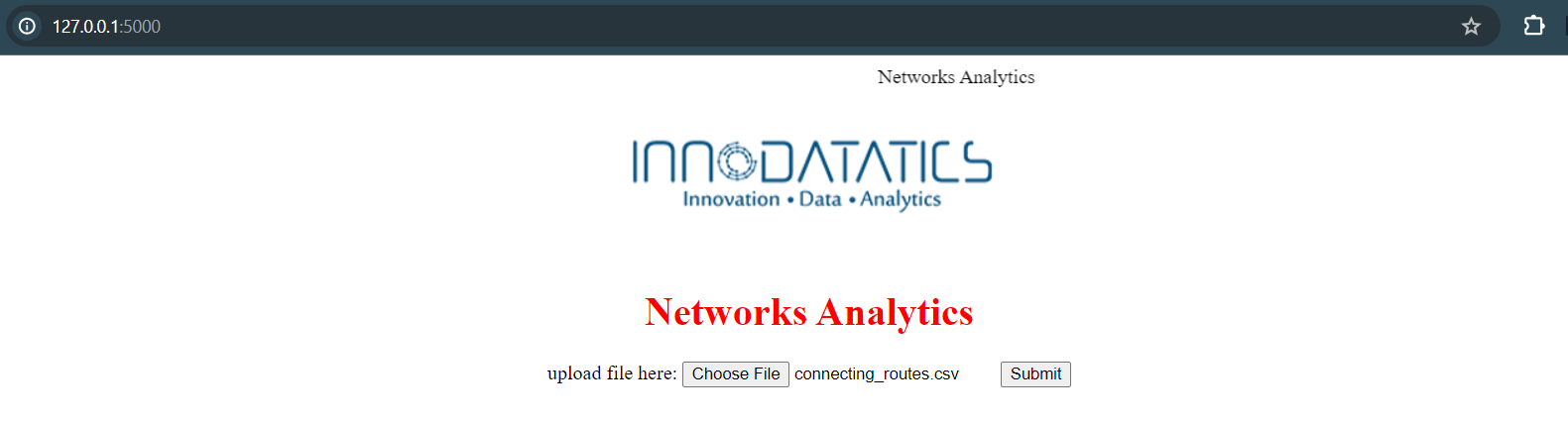
AER 0.047619 0.328125 0.139052 0.000000

KZN 0.333333 0.477273 0.480704 0.526984

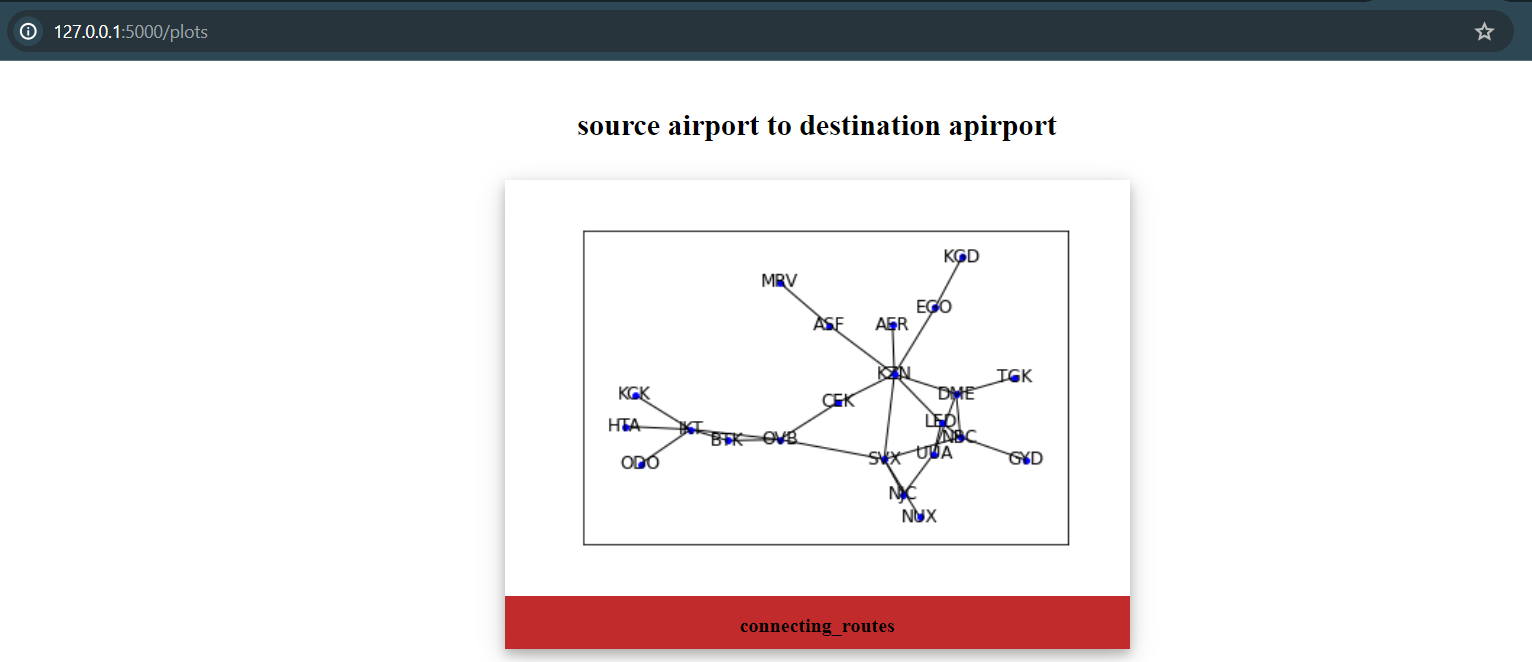
ASF 0.095238 0.338710 0.151750 0.095238

MRV 0.047619 0.256098 0.043896 0.000000

CEK 0.095238 0.403846 0.209463 0.114286

****

****

****

**2.**

'''

Problem statement 2:

Business Problem: There are three datasets given (Facebook, Instagram, and LinkedIn). Construct and visualize the following networks:

● circular network for Facebook

● star network for Instagram

● star network for LinkedIn

Create a network using an adjacency matrix (undirected only).

'''

import networkx as nx

import matplotlib.pyplot as plt

# Define adjacency matrices for Facebook, Instagram, and LinkedIn

facebook\_adjacency = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Network Analytics/Assignments/3e.Network Analytics/facebook.csv")

instagram\_adjacency = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Network Analytics/Assignments/3e.Network Analytics/instagram.csv")

linkedin\_adjacency = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Network Analytics/Assignments/3e.Network Analytics/linkedin.csv")

# Convert DataFrames to NumPy arrays

facebook\_matrix = facebook\_adjacency.values

instagram\_matrix = instagram\_adjacency.values

linkedin\_matrix = linkedin\_adjacency.values

# Create networks

facebook\_network = nx.DiGraph(facebook\_matrix)

instagram\_network = nx.DiGraph(instagram\_matrix)

linkedin\_network = nx.DiGraph(linkedin\_matrix)

data1 = pd.DataFrame({"Degree": pd.Series(nx.degree\_centrality(facebook\_network)),

"closeness":pd.Series(nx.closeness\_centrality(facebook\_network)),

"eigenvector": pd.Series(nx.eigenvector\_centrality(facebook\_network)),

"betweenness": pd.Series(nx.betweenness\_centrality(facebook\_network))})

data1

data2 = pd.DataFrame({"Degree": pd.Series(nx.degree\_centrality(instagram\_network)),

"closeness":pd.Series(nx.closeness\_centrality(instagram\_network)),

"eigenvector": pd.Series(nx.eigenvector\_centrality(instagram\_network)),

"betweenness": pd.Series(nx.betweenness\_centrality(instagram\_network))})

data2

data3 = pd.DataFrame({"Degree": pd.Series(nx.degree\_centrality(linkedin\_network)),

"closeness":pd.Series(nx.closeness\_centrality(linkedin\_network)),

"eigenvector": pd.Series(nx.eigenvector\_centrality(linkedin\_network)),

"betweenness": pd.Series(nx.betweenness\_centrality(linkedin\_network))})

data3

# Plot networks

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)

nx.draw\_circular(facebook\_network, with\_labels=True, node\_size=500, node\_color='skyblue', font\_size=10, font\_weight='bold')

plt.title('Facebook Circular Network')

plt.subplot(1, 3, 2)

pos = nx.spring\_layout(instagram\_network)

nx.draw(instagram\_network, pos, with\_labels=True, node\_size=500, node\_color='skyblue', font\_size=10, font\_weight='bold')

plt.title('Instagram Star Network')

plt.subplot(1, 3, 3)

pos = nx.spring\_layout(linkedin\_network)

nx.draw(linkedin\_network, pos, with\_labels=True, node\_size=400, node\_color='skyblue', font\_size=10, font\_weight='bold')

plt.title('LinkedIn Star Network')

plt.axis('equal') # Ensure the aspect ratio is equal to avoid stretching

plt.tight\_layout()

plt.show()

**Output:**

Degree closeness eigenvector betweenness

0 0.5 0.4 0.333333 0.214286

1 0.5 0.4 0.333333 0.214286

2 0.5 0.4 0.333333 0.214286

3 0.5 0.4 0.333333 0.214286

4 0.5 0.4 0.333333 0.214286

5 0.5 0.4 0.333333 0.214286

6 0.5 0.4 0.333333 0.214286

7 0.5 0.4 0.333333 0.214286

8 0.5 0.4 0.333333 0.214286

Degree closeness eigenvector betweenness

0 2.000000 1.000000 0.707107 1.0

1 0.285714 0.538462 0.267261 0.0

2 0.285714 0.538462 0.267261 0.0

3 0.285714 0.538462 0.267261 0.0

4 0.285714 0.538462 0.267261 0.0

5 0.285714 0.538462 0.267261 0.0

6 0.285714 0.538462 0.267261 0.0

7 0.285714 0.538462 0.267261 0.0

Degree closeness eigenvector betweenness

0 0.333333 0.255319 0.197843 0.000000

1 0.500000 0.324324 0.280258 0.075758

2 0.500000 0.324324 0.280258 0.075758

3 0.500000 0.413793 0.315914 0.409091

4 0.333333 0.255319 0.197843 0.000000

5 0.500000 0.324324 0.280258 0.075758

6 0.500000 0.324324 0.280258 0.075758

7 0.500000 0.413793 0.315914 0.409091

8 0.333333 0.255319 0.197843 0.000000

9 0.500000 0.324324 0.280258 0.075758

10 0.500000 0.324324 0.280258 0.075758

11 0.500000 0.413793 0.315914 0.409091

12 0.500000 0.500000 0.334517 0.727273

