Modul Praktikum 7
Data Mining

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2024

Data Graf II

Clustering Coefficient

Dalam teori graf Clustering Coefficient adalah ukuran yang digunakan untuk menggambarkan seberapa dekat node berinteraksi satu sama lain. Semakin tinggi clustering coefficient, semakin besar kemungkinan bahwa node yang terhubung juga terhubung dengan node lain.

$$C(v) = \frac{2E}{k(k-1)}$$

Dimana $C(v)$ adalah clustering coefficient dari node v. E adalah jumlah koneksi antara tetangga node v. k adalah derajat node v. Untuk mencari nilai clustering coefficient menggunakan Python harus menggunakan pustaka NetworkX.

```
pip install networkx
```

```
import networkx as nx
import matplotlib.pyplot as plt

# Membuat graf contoh
G = nx.Graph()

# Menambahkan simpul dan tepi
edges = [
    (1, 2), (1, 3), (2, 3),
    (3, 4), (4, 5), (5, 6),
    (4, 6), (1, 5), (2, 4)
]

G.add_edges_from(edges)

# Menggambar graf
nx.draw(G, with_labels=True, node_color='lightblue', node_size=800, font_size=16)
plt.show()

# Menghitung clustering coefficient untuk seluruh node
clustering_coeffs = nx.clustering(G)

# Menampilkan hasil
print("Clustering Coefficient untuk setiap node:")
for node, coeff in clustering_coeffs.items():
    print(f"Node {node}: {coeff:.2f}")

# Menghitung clustering coefficient untuk graf secara keseluruhan
average_clustering = nx.average_clustering(G)
print(f"\nAverage Clustering Coefficient: {average_clustering:.2f}")
```

Efficiency

Efficiency graf didefinisikan sebagai kebalikan dari jarak rata-rata antara semua pasangan simpul dalam graf. Semakin kecil jarak rata-rata, semakin tinggi efisiensinya. Efficiency dapat dihitung menggunakan rumus berikut:

$$E(G) = \frac{1}{n(n-1)} \sum_{i} \sum_{j>1} \frac{1}{d(v_i, v_j)}$$

Dimana $E(G)$ adalah efficiency graf, n adalah jumlah simpul dalam graf, dan $d(v_i, v_j)$ adalah jarak terpendek antara node v_i dan v_j. Perhitungan efficiency pada python dapat menggunakan source code berikut.

```
# Menghitung efisiensi graf
efficiency = nx.global_efficiency(G)

# Menampilkan hasil
print(f"Global Efficiency of the graph: {efficiency:.2f}")
```

Centrality

Mengukur seberapa "sentral" atau "penting" suatu node dalam jaringan. Node dengan centrality tinggi biasanya memiliki pengaruh lebih besar dalam jaringan. Terdapat beberapa centrality pada teori graf, yaitu degree centrality, eccentricity centrality, closeness centrality, dan betweenness centrality.

Degree centrality mengukur jumlah koneksi langsung yang dimiliki oleh suatu node. Node dengan degree centrality tinggi memiliki banyak koneksi. Degree centrality dapat dihitung menggunakan rumus:

$$C_D(v) = \deg(v)$$

Dimana $\deg(v)$ adalah derajat dari node v.

Eccentricity centrality mengukur seberapa jauh suatu node dari node lain dalam jaringan, dinyatakan dengan jarak terjauh ke node lain. Eccentricity dapat dihitung menggunakan rumus:

$$C_E(v) = \max_{u \in V} d(v, u)$$

Dimana $d(v, u)$ adalah jarak terpendek antara node v dan u.

Closeness centrality mengukur seberapa cepat suatu node dapat mencapai semua node lain dalam jaringan. Node dengan closeness centrality tinggi dapat menjangkau node lain dengan cepat. Closeness centrality dapat dihitung menggunakan rumus:

$$C_C(v) = \frac{1}{\sum_{u \in V} d(v, u)}$$

Betweenness centrality mengukur seberapa sering suatu node terletak di jalur terpendek antara dua node lain. Node dengan betweenness centrality tinggi sering menjadi penghubung dalam jaringan. Betweenness centrality dapat dihitung menggunakan rumus:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Dimana σ_{st} adalah jumlat jalur terpendek antara node s dan t, dan $\sigma_{st}(v)$ adalah jumlah jalur terpendek yang melewati node v.

```python
# Menghitung Degree Centrality
degree_centrality = nx.degree_centrality(G)

# Menghitung Eccentricity
eccentricity = nx.eccentricity(G)

# Menghitung Closeness Centrality
closeness_centrality = nx.closeness_centrality(G)

# Menghitung Betweenness Centrality
betweenness_centrality = nx.betweenness_centrality(G)

# Menampilkan hasil
print("Degree Centrality:")
for node, centrality in degree_centrality.items():
    print(f"Node {node}: {centrality:.2f}")

print("\nEccentricity:")
for node, ecc in eccentricity.items():
    print(f"Node {node}: {ecc}")

print("\nCloseness Centrality:")
for node, centrality in closeness_centrality.items():
    print(f"Node {node}: {centrality:.2f}")

print("\nBetweenness Centrality:")
for node, centrality in betweenness_centrality.items():
    print(f"Node {node}: {centrality:.2f}")
```

Latihan

Bagaimana mencari clustering coefficient, efficiency dan centrality dari suatu data yang memiliki 4 fitur? Silahkan coba source code berikut.

```python
import pandas as pd
import numpy as np
import networkx as nx
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt

# 1. Membuat dataset dengan distribusi normal
data = {
    'Panjang Petal': np.random.normal(loc=3.5, scale=0.5, size=10),
    'Lebar Petal': np.random.normal(loc=1.0, scale=0.3, size=10),
    'Panjang Sepal': np.random.normal(loc=5.0, scale=0.5, size=10),
    'Lebar Sepal': np.random.normal(loc=1.5, scale=0.3, size=10)
}
df = pd.DataFrame(data)

# 2. Menghitung jarak antar titik
distances = pairwise_distances(df)

# 3. Membangun graf berdasarkan jarak (gunakan threshold untuk menghubungkan node)
threshold = 0.7  # Sesuaikan threshold ini sesuai kebutuhan
G = nx.Graph()

# Menambahkan tepi berdasarkan jarak
for i in range(len(distances)):
    for j in range(i + 1, len(distances)):
        if distances[i][j] < threshold:
            G.add_edge(i, j)
```

```python
# 4. Menghitung Clustering Coefficient
if G.number_of_nodes() > 0:
    clustering_coeffs = nx.clustering(G)
    average_clustering = nx.average_clustering(G)
else:
    clustering_coeffs = {}
    average_clustering = 0

# 5. Menghitung Efisiensi
if G.number_of_nodes() > 1:
    efficiency = nx.global_efficiency(G)
else:
    efficiency = 0
```

```python
# 6. Menghitung Centrality
degree_centrality = nx.degree_centrality(G)
eccentricity = nx.eccentricity(G)
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)

# 7. Menampilkan hasil
print("Clustering Coefficient:")
for node, coeff in clustering_coeffs.items():
    print(f"Node {node}: {coeff:.2f}")
print(f"\nAverage Clustering Coefficient: {average_clustering:.2f}")

print(f"\nGlobal Efficiency: {efficiency:.2f}")

print("\nDegree Centrality:")
for node, centrality in degree_centrality.items():
    print(f"Node {node}: {centrality:.2f}")

print("\nEccentricity:")
for node, ecc in eccentricity.items():
    print(f"Node {node}: {ecc}")

print("\nCloseness Centrality:")
for node, centrality in closeness_centrality.items():
    print(f"Node {node}: {centrality:.2f}")

print("\nBetweenness Centrality:")
for node, centrality in betweenness_centrality.items():
    print(f"Node {node}: {centrality:.2f}")
```

```python
# 8. Menampilkan graf
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G)  # Menentukan posisi node
nx.draw_networkx_nodes(G, pos, node_size=300)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos, font_size=10)
plt.title("Graph Representation of the Dataset")
plt.axis('off')  # Matikan axis
plt.show()
```