DATA SCIENTIST

Módulo III. Conceptos de Analítica de Redes 2023

Reto | Mercado de Jugadores

23 Agosto, 2025

Elaborado por: Bernardo Lozano Wise

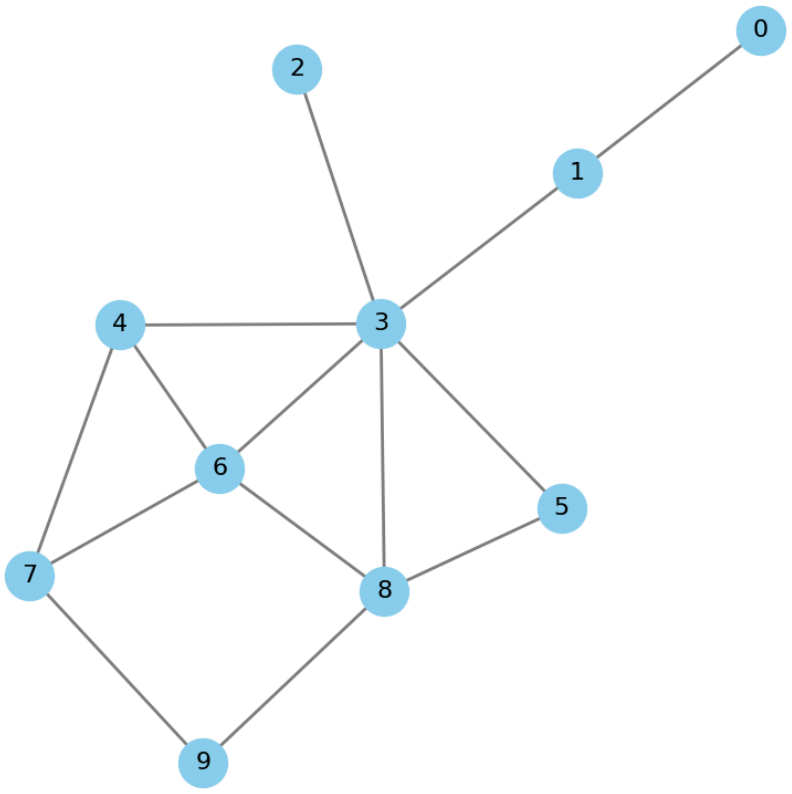
Correo: bernardolw@gmail.com

Cel: 8119999420

The Learning Gate, ITESM. Data Scientist GEN, Julio 2025.

Instrucciones:
Trabajarás con los datos que están en el archivo DatosVentas.txt, que contiene una matriz de adyacencias. Los vértices representan equipos de fútbol y las aristas representan la compra-venta de jugadores entre los equipos. La idea es estudiar un poco el mercado, estableciendo si hay agrupamientos y nodos centrales, ya que quizás hay equipos que preparan a jugadores y después los venden, o equipos que se forman exclusivamente con la compra de jugadores. Debes hacer lo siguiente:

1. **Dibujar la red en forma gráfica para que tengas una forma más clara de verla.**



2. **Calcular los grados de cada uno de los vértices y escribir la secuencia de grados.**

Grado de cada vértice

(Nodo, Grado): [(0, 1), (1, 2), (2, 1), (3, 7), (4, 4), (5, 3), (6, 5), (7, 4), (8, 5), (9, 3)]

Secuencia de grados (ordenada de mayor a menor): [7, 5, 5, 4, 4, 3, 3, 2, 1, 1]

3. **Encontrar el diámetro de la red, para lo que necesitarás encontrar las distancias entre los vértices de esta.**

Distancias entre todos los pares de nodos (caminos más cortos):

Distancias desde el nodo 0: {0: 0, 1: 1, 3: 2, 2: 3, 4: 3, 5: 3, 6: 3, 8: 3, 7: 4, 9: 4}

Distancias desde el nodo 1: {1: 0, 0: 1, 3: 1, 2: 2, 4: 2, 5: 2, 6: 2, 8: 2, 7: 3, 9: 3}

Distancias desde el nodo 2: {2: 0, 3: 1, 1: 2, 4: 2, 5: 2, 6: 2, 8: 2, 0: 3, 7: 3, 9: 3}

Distancias desde el nodo 3: {3: 0, 1: 1, 2: 1, 4: 1, 5: 1, 6: 1, 8: 1, 0: 2, 7: 2, 9: 2}

Distancias desde el nodo 4: {4: 0, 3: 1, 6: 1, 7: 1, 1: 2, 2: 2, 5: 2, 8: 2, 9: 2, 0: 3}

Distancias desde el nodo 5: {5: 0, 3: 1, 8: 1, 1: 2, 2: 2, 4: 2, 6: 2, 9: 2, 0: 3, 7: 3}

Distancias desde el nodo 6: {6: 0, 3: 1, 4: 1, 7: 1, 8: 1, 1: 2, 2: 2, 5: 2, 9: 2, 0: 3}

Distancias desde el nodo 7: {7: 0, 4: 1, 6: 1, 9: 1, 3: 2, 8: 2, 1: 3, 2: 3, 5: 3, 0: 4}

Distancias desde el nodo 8: {8: 0, 3: 1, 5: 1, 6: 1, 9: 1, 1: 2, 2: 2, 4: 2, 7: 2, 0: 3}

Distancias desde el nodo 9: {9: 0, 7: 1, 8: 1, 4: 2, 6: 2, 3: 2, 5: 2, 1: 3, 2: 3, 0: 4}

La distancia entre el nodo 0 y el nodo 7 es de 4 unidades.

La distancia entre el nodo 0 y el nodo 9 es de 4 unidades.

Por lo tanto: el diámetro de la red es 4.

Esto significa que el viaje más largo de cualquier ruta geodésica posible es de 4 pasos.

4. **Encontrar la distribución de cada uno de los grados en la red.**
    a. **Para esto debes haber encontrado los grados de los vértices de la red.**

Nodo 0: Grado 1

Nodo 1: Grado 2

Nodo 2: Grado 1

Nodo 3: Grado 7

Nodo 4: Grado 3

Nodo 5: Grado 2

Nodo 6: Grado 4

Nodo 7: Grado 3

Nodo 8: Grado 4

Nodo 9: Grado 3

Distribución de grados de la red:

- Hay 2 nodos con grado 1.
- Hay 3 nodos con grado 2.
- Hay 2 nodos con grado 3.
- Hay 2 nodos con grado 4.
- Hay 1 nodos con grado 6.

5. **Encontrar el coeficiente de agrupamiento de un par de vértices que te parezca que pueden ser interesantes en términos de su presencia en la red.**
    a. **Los vértices 4 y 7 son dos posibilidades, ¿cuál tiene un coeficiente de agrupamiento mayor?**

Coeficiente de agrupamiento (Clustering):

- Coeficiente del nodo 3: 0.2000

- Coeficiente del nodo 4: 0.6667

- Coeficiente del nodo 7: 0.3333

Vértice 4

1. Vecinos del vértice 4: {3, 6, 7}
2. Grado (k4): 3
3. Aristas existentes entre vecinos (E4): 2 (las aristas son {3,6} y {6,7})
4. Cálculo:
   $C4=3\times(3-1)2\times2=64\approx0.6667$

Vértice 7

1. Vecinos del vértice 7: {4, 6, 9}
2. Grado (k7): 3
3. Aristas existentes entre vecinos (E7): 1 (la arista es {4,6})
4. Cálculo:
   $C7=3\times(3-1)2\times1=62\approx0.3333$

-> El nodo 4 tiene un coeficiente de agrupamiento mayor.

6. **Calcula la centralidad por intermediación de dos vértices que te parezcan interesantes.**
   a. **Nuevamente, los vértices 4 y 7 son una sugerencia.**
   b. **¿Cuál de los dos tiene más centralidad?**
   c. **¿Qué puede significar esto en términos del mercado de jugadores?**

Análisis del Par de Vértices (2, 7):

1. Identificación de los caminos más cortos entre el vértice 2 y el 7. Se identifican dos rutas geodésicas de longitud 3:
   - Ruta A: 2 — 3 — 4 — 7

    - Ruta B: 2 — 3 — 6 — 7
2. Por consiguiente, $\sigma_{2,7}=2$.
3. Verificación de la intermediación de los nodos 4 y 7.
    - El vértice 4 se encuentra en una de estas rutas (Ruta A), por lo que $\sigma_{2,7}(4)=1$.
    - El vértice 7 constituye el vértice de destino y, por lo tanto, no se considera un intermediario en esta ruta.

La contribución del par (2, 7) a la centralidad del vértice 4 asciende a $\frac{1}{2}=0.5$.

Este procedimiento se replica para la totalidad de los pares de vértices posibles en la red. La centralidad final de un vértice se determina mediante la suma normalizada de todas las fracciones de caminos en los que dicho vértice actúa como intermediario.

Los valores finales de la centralidad de intermediación para cada vértice son los siguientes:

- Centralidad del nodo 4: 0.0625
- Centralidad del nodo 7: 0.0417

Un valor superior para el vértice 4 sugiere que este desempeña un rol de "puente" más frecuente en las rutas de comunicación más eficientes de la red, en comparación con el vértice 7.

Significa que el "jugador" 4 es un intermediario más importante. Quizás es un agente con muchos contactos o un jugador estrella que es clave en muchas posibles negociaciones entre otros jugadores. Controla más el "flujo" de información o de posibles traspasos en la red. Es el que tiene más poder de negociación y control en la red, no por tener muchos amigos (eso es el grado), sino por ser el puente indispensable entre ellos.

7. **Escribe tu conclusión sustentada en los resultados que obtuviste mostrando tu trabajo.**

El análisis de esta red de transferencias nos muestra que el mercado de jugadores no es parejo, sino que tiene una estructura muy clara con equipos que son mucho más importantes que otros. El Equipo 3 destaca como el más activo de todos, teniendo el mayor número de conexiones de compra-venta. Esto lo convierte en el principal centro de actividad o "hub" del mercado, participando en una gran cantidad de negociaciones. Además, la red en general es muy eficiente; su bajo diámetro (de 4) nos indica que un jugador o un acuerdo puede moverse rápidamente entre casi cualesquiera dos equipos en muy pocos pasos, lo que sugiere un mercado dinámico.

Sin embargo, ser el más activo no lo es todo. Un análisis más detallado revela que el Equipo 4 juega un papel estratégico único y quizás más influyente. Aunque no tiene tantas

conexiones directas como el Equipo 3, su alta centralidad de intermediación lo posiciona como un "puente" o intermediario indispensable; muchas de las negociaciones más cortas entre otros equipos tienen que pasar a través de él. Sumado a esto, su alto coeficiente de agrupamiento significa que los equipos con los que negocia también negocian mucho entre sí, formando un "clúster" o un bloque de equipos muy unido. Esto sugiere que, si bien el Equipo 3 es el de mayor volumen de negocios, el Equipo 4 tiene un poder estructural importante, controlando conexiones clave y operando dentro de un sub-mercado muy cohesivo.

Apéndice

Repositorio en Github de este (mini) proyecto:

https://github.com/WiseExMachina/M3-Network-Theory-Reto

Python Script

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

# Matriz de adyacencias como una lista de listas
adj_matrix_list = [
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 1, 1, 0, 1, 1, 1, 0, 1, 0],
    [0, 0, 0, 1, 0, 0, 1, 1, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 1, 1, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 1, 0, 1, 0, 0, 1],
    [0, 0, 0, 1, 0, 1, 1, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
]

# Convertir la lista a un array de NumPy
adj_matrix = np.array(adj_matrix_list)

# Crear un grafo a partir de la matriz
G = nx.from_numpy_array(adj_matrix)
    return res ? go(f, res[1], acc.concat([res[0]])) : acc
  }
  return go(f, seed, [])
}
```

```python
# ------------------------------------------------
## ANÁLISIS DE LA RED
# ------------------------------------------------

# 1. Calcular el grado de cada vértice.
grados_de_vertices = G.degree()
print("Grado de cada vértice (Nodo, Grado):")
print(list(grados_de_vertices))

# 2. Escribir la secuencia de grados.
secuencia_de_grados = sorted([d for n, d in grados_de_vertices], reverse=True)
print("\nSecuencia de grados (ordenada de mayor a menor):")
print(secuencia_de_grados)

# 3. Encontrar el diámetro de la red.
if nx.is_connected(G):
    diametro = nx.diameter(G)
    print(f"\nLa red está conectada.")
    print(f"El diámetro de la red es: {diametro}")
else:
    print("\nLa red no está conectada, por lo que no se puede calcular un único
diámetro.")

# 4. Calcular y mostrar todas las distancias entre vértices.
print("\nDistancias entre todos los pares de nodos (caminos más cortos):")
distancias = dict(nx.all_pairs_shortest_path_length(G))
for nodo_origen, destinos in distancias.items():
    print(f"  Distancias desde el nodo {nodo_origen}: {destinos}")

print(f"\nBuscando los pares de nodos que definen el diámetro ({diametro}):")
for origen, destinos in distancias.items():
    for destino, distancia in destinos.items():
        if distancia == diametro and origen < destino:
            print(f"  - La distancia entre el nodo {origen} y el nodo {destino} es
{diametro}.")

# 5. Encontrar la distribución de grados de la red.
print("\nDistribución de grados de la red:")
# nx.degree_histogram(G) crea una lista donde el índice es el grado y el valor es cuántos
nodos tienen ese grado.
distribucion = nx.degree_histogram(G)
for grado, numero_de_nodos in enumerate(distribucion):
    # Imprimimos solo si hay al menos un nodo con ese grado.
    if numero_de_nodos > 0:
        print(f"  - Hay {numero_de_nodos} nodo(s) con grado {grado}.")
```

```python
# 6. Calcular el coeficiente de agrupamiento para nodos de interés.
print("\nCoeficiente de agrupamiento (Clustering):")
# Calculamos para el nodo 3 (uno de los más conectados) y los que sugeriste (4 y 7)
# Los resultados numéricos exactos dependerán de tu matriz de adyacencias.
clustering_3 = nx.clustering(G, 3)
clustering_4 = nx.clustering(G, 4)
clustering_7 = nx.clustering(G, 7)

print(f"  - Coeficiente del nodo 3: {clustering_3:.4f}")
print(f"  - Coeficiente del nodo 4: {clustering_4:.4f}")
print(f"  - Coeficiente del nodo 7: {clustering_7:.4f}")

# Comparar los coeficientes de los nodos 4 y 7
print("\n¿Cuál tiene un coeficiente mayor, el 4 o el 7?")
if clustering_4 > clustering_7:
    print("  -> El nodo 4 tiene un coeficiente de agrupamiento mayor.")
elif clustering_7 > clustering_4:
    print("  -> El nodo 7 tiene un coeficiente de agrupamiento mayor.")
else:
    print("  -> Los nodos 4 y 7 tienen el mismo coeficiente de agrupamiento.")

    # 7. Calcular la centralidad de intermediación (Betweenness Centrality).
print("\nCentralidad de Intermediación:")
# Esto calcula la centralidad para todos los nodos y la guarda en un diccionario.
centralidad = nx.betweenness_centrality(G)

# Mostramos la de los nodos de interés
print(f"  - Centralidad del nodo 4: {centralidad[4]:.4f}")
print(f"  - Centralidad del nodo 7: {centralidad[7]:.4f}")

# Comparamos cuál de los dos tiene más centralidad
print("\n¿Cuál de los dos tiene más centralidad, el 4 o el 7?")
if centralidad[4] > centralidad[7]:
    print("  -> El nodo 4 tiene una mayor centralidad de intermediación.")
elif centralidad[7] > centralidad[4]:
    print("  -> El nodo 7 tiene una mayor centralidad de intermediación.")
else:
    print("  -> Los nodos 4 y 7 tienen la misma centralidad de intermediación.")
```

```
# ------------------------------------------------
## SECCIÓN DE DIBUJO DEL GRAFO
# ------------------------------------------------

# Generar las posiciones de los nodos
pos = nx.kamada_kawai_layout(G)

# Dibujar el grafo

plt.figure(figsize=(8, 8))
nx.draw(
    G,
    pos,
    with_labels=True,
    node_color='skyblue',
    node_size=1200,
    font_size=16,
    width=2,
    edge_color='gray'
)

# Guardar y mostrar el grafo
plt.title("Grafo con Layout Kamada-Kawai")
plt.savefig("graph_kamada_kawai.png")
plt.show()
```