Aprendizaje Automático Relacional

Grupo 2:
Giraldo Santiago, Luis
Guillén Fernández, David
Departamento de Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Email: luigirsan1@alum.us.es, davguifer@alum.us.es

***Abstract*—En este trabajo, presentamos un enfoque basado en inteligencia artificial para la predicción de propiedades en grafos utilizando métricas estructurales como características de entrada. Nuestro estudio se centra en cómo las métricas derivadas de las posiciones de los nodos, tales como centralidad de grado, centralidad de cercanía y coeficiente de agrupamiento, pueden ser utilizadas para entrenar modelos de IA que predicen una propiedad del nodo del grafo. Utilizamos un conjunto de datos compuesto por grafos generados y reales, de los cuales extraemos las métricas mencionadas anteriormente. Estas métricas sirven como características de entrada para entrenar diversos modelos de aprendizaje automático, tales como Naive Bayes y Árboles de decisión. Los resultados muestran que los modelos de IA pueden aprender efectivamente a predecir propiedades del grafo con alta precisión basándose únicamente en las métricas estructurales. Además, discutimos las dificultades encontradas durante el entrenamiento de los modelos y los resultados obtenidos. Concluimos que el uso de métricas estructurales de grafos como características de entrada para modelos de IA es una buena metodología para la predicción de propiedades.**

I. Introducción

El aprendizaje automático relacional se enfoca en la creación de modelos que aprenden a partir de datos relacionales, utilizando grafos para representar las interacciones entre entidades. A diferencia de los métodos tradicionales, este enfoque explota las relaciones entre los datos, permitiendo capturar su naturaleza interconectada.

En este proyecto, nos hemos centrado en la clasificación de nodos en un grafo mediante la construcción manual de características relacionales. Los pasos clave incluyen:

- **Estudio de teoría de grafos**: Comprender métricas como centralidad y modularidad.
- **Uso de herramientas Python**: Utilizar librerías como NetworkX y Scikit-Learn.
- **Desarrollo y evaluación de modelos**: Generar modelos de clasificación basados en características relacionales y no relacionales.
- **Análisis y ajuste**: Realizar análisis de correlación y ajuste de hiper-parámetros.

II. Descripción del Conjunto de Datos y la Tarea de Predicción

El conjunto de datos utilizado en este trabajo es el `musae_git`, obtenido de [1]. Esta red está compuesta por nodos que representan usuarios y enlaces que representan las interacciones entre estos usuarios.

A. Características Principales

El conjunto de datos se divide en dos archivos principales:
1) `musae_git_edges.csv`: Este archivo contiene las aristas del grafo, representando las interacciones entre los usuarios.

- **id_node_1**: Identificador del primer nodo (usuario).
- **id_node_2**: Identificador del segundo nodo (usuario).

Ejemplo de registros:

id_node_1	id_node_2
0	23977
1	34526
1	2370
1	14683
1	29982

2) `musae_git_target.csv`: Este archivo contiene información sobre los nodos, incluyendo su nombre y la etiqueta objetivo (`ml_target`) utilizada para la tarea de predicción.

- **id_node**: Identificador del nodo (usuario).
- **name**: Nombre del usuario.
- **ml_target**: Etiqueta de clasificación, donde 0 y 1 representan las categorías a predecir (desarrollador web o de aprendizaje automático respectivamente).

Ejemplo de registros:

id_node	name	ml_target
0	Eiryyy	0
1	shawflying	0
2	JpMCarrilho	1
3	SuhwanCha	0
4	sunilangadi2	1

B. Tarea de Predicción

La tarea de predicción en este proyecto se centra en la clasificación de los nodos del grafo en dos categorías (0 y 1) basándose en sus interacciones dentro de la red.

- **Objetivo**: Predecir la etiqueta `ml_target` de cada nodo utilizando métricas estructurales del grafo como características de entrada.
- **Método**: Utilizaremos varios modelos de aprendizaje automático supervisado, tales como Naive Bayes, K-Nearest Neighbors (KNN) y Árboles de Decisión, para entrenar y evaluar los modelos de clasificación.

La tarea implica los siguientes pasos:

1) Extracción de Métricas: Calcular métricas estructurales del grafo como la centralidad de grado, centralidad de cercanía y coeficiente de agrupamiento para cada nodo.

2) Entrenamiento del Modelo: Utilizar estas métricas como características de entrada para entrenar los modelos de aprendizaje automático supervisado.

3) Evaluación: Evaluar el rendimiento de los modelos utilizando métricas estándar como el *recall* y comparando matrices de confusión.

III. Descripción de las Métricas Relacionales Utilizadas

En este trabajo se han utilizado diversas métricas estructurales de los grafos para caracterizar los nodos y así poder predecir sus propiedades. A continuación, se describen las principales métricas utilizadas:

A. Centralidad de Grado (Degree centrality)

La centralidad de grado mide el número de conexiones directas que tiene un nodo en un grafo. Es una métrica básica que indica la importancia de un nodo dentro de la red.

$$\text{Centralidad de Grado}(v) = \deg(v)$$

donde $\deg(v)$ representa el número de aristas incidentes en el nodo v.

`degree_centrality(G, nodes)`

Parámetros:

- `G` - Grafo. Una red bipartita.
- `nodes` - Lista o contenedor. Contenedor con todos los nodos en uno de los conjuntos de nodos bipartitos.

Retorna:

`centrality` - Diccionario. Diccionario indexado por nodo con la centralidad de grado bipartita como valor.

(Fuente: [12],[21])

B. Centralidad de Cercanía (Closeness centrality)

La centralidad de cercanía mide qué tan cerca está un nodo de todos los demás nodos en la red. Se define como el inverso de la suma de las distancias más cortas desde el nodo a todos los demás nodos.

$$\text{Centralidad de Cercanía}(v) = \frac{1}{\sum_{u \neq v} d(u,v)}$$

donde $d(u,v)$ es la distancia más corta entre los nodos u y v.

`closeness_centrality(G, nodes, normalized=True)`

Parámetros:

- `G` - Grafo. Una red bipartita.
- `nodes` - Lista o contenedor. Contenedor con todos los nodos en uno de los conjuntos de nodos bipartitos.
- `normalized` - Booleano, opcional. Si es Verdadero (por defecto), normalizar por el tamaño del componente conectado.

Retorna:

`closeness` - Diccionario. Diccionario indexado por nodo con la centralidad de cercanía bipartita como valor.

(Fuente: [13],[22])

C. Centralidad de Intermediación (Betweenness centrality)

Se define formalmente un grafo como un par ordenado $G = (V, E)$, donde V es su conjunto de nodos o vértices y E su conjunto de aristas. El número de vértices se denota como $n = |V|$.

Formalmente, la intermediación $C_B(i)$ de un nodo i en una red o grafo se define como:

$$C_B(i) = \sum_{j \neq k \in V} \frac{b_{jik}}{b_{jk}}$$

donde b_{jk} es el número de caminos más cortos desde el nodo j hasta el nodo k, y b_{jik} el número de caminos más cortos desde j hasta k que pasan a través del nodo i.

`betweenness_centrality(G, nodes)`

Parámetros:

- `G` - Grafo. Un grafo bipartito.
- `nodes` - Lista o contenedor. Contenedor con todos los nodos en uno de los conjuntos de nodos bipartitos.

Retorna:

`betweenness` - Diccionario. Diccionario indexado por nodo con la centralidad de intermediación bipartita como valor.

(Fuente: [14],[23])

D. Coeficiente de Agrupamiento (Clustering coefficient)

El coeficiente de agrupamiento local de un vértice en un grafo cuantifica qué tan cerca están sus vecinos de formar un cliqué (grafo completo). Se define como la proporción de los enlaces existentes entre los vértices dentro de su vecindario en comparación con el número total de enlaces posibles. Para un grafo no dirigido, se expresa como:

$$C_i = \frac{2 \times |\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)}$$

Donde:

- C_i: Coeficiente de agrupamiento local del vértice v_i.
- k_i: Grado del vértice v_i, es decir, el número de vecinos que tiene.
- N_i: Vecindario del vértice v_i, que consiste en todos los vértices conectados directamente a v_i.
- $|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|$: Número de enlaces entre los vértices en el vecindario N_i.

`clustering(G, nodes=None, weight=None)`

Parámetros:

- `G` - Grafo. El grafo.

- `nodes` - Nodo, iterable de nodos o None (por defecto=None). Si es un solo nodo, devuelve el número de triángulos para ese nodo. Si es un iterable, calcula el número de triángulos para cada uno de esos nodos. Si es None (el valor predeterminado), calcula el número de triángulos para todos los nodos en G.
- `weight` - Cadena o None, opcional (por defecto=None). El atributo de borde que contiene el valor numérico utilizado como peso. Si es None, entonces cada borde tiene un peso de 1.

Retorna:

`out` - Flotante o diccionario. Coeficiente de agrupamiento en los nodos especificados.

(Fuente: [15],[24])

E. Agrupación cuadrada (Square clustering)

El coeficiente de agrupación cuadrada mide la tendencia de los nodos a formar cuadrados en una red:

$$C_4(v) = \frac{\sum_{u=1}^{k_v} \sum_{w=u+1}^{k_v} q_v(u,w)}{\sum_{u=1}^{k_v} \sum_{w=u+1}^{k_v} [a_v(u,w) + q_v(u,w)]}$$

donde $q_v(u,w)$ son el número de vecinos comunes de u y w, $a_v(u,w) = (k_u - (1 + q_v(u,w) + \theta_{uv})) + (k_w - (1 + q_v(u,w) + \theta_{uw}))$ son los nodos distintos de v que forman cuadrados, y $\theta_{uw} = 1$ si u y w están conectados y 0 en caso contrario. La suma se realiza sobre todos los pares de nodos distintos u y w en el conjunto de vecinos de v.

`square_clustering(G, nodes=None)`

Parámetros:

- `G` - Grafo. El grafo.
- `nodes` - Contenedor de nodos, opcional (por defecto=todos los nodos en G). Calcula el coeficiente de agrupamiento para nodos en este contenedor.

Retorna:

`c4` - Diccionario. Un diccionario indexado por nodo con el valor del coeficiente de agrupamiento cuadrado.

(Fuente: [16])

F. Triángulos (Triangles)

El número de triángulos es una medida que cuenta cuántos grupos de tres nodos están todos conectados entre sí y pasan por el nodo en cuestión.

`triangles(G, nodes=None)`

Parámetros:

- `G` - Grafo. El grafo NetworkX.
- `nodes` - Nodo, iterable de nodos o None (por defecto=None). Si es un nodo singular, devuelve el número de triángulos para ese nodo. Si es un iterable, calcula el número de triángulos para cada uno de esos nodos. Si es None (el valor por defecto), calcula el número de triángulos para todos los nodos en G.

Retorna:

`out` - Diccionario o entero. Si nodes es un contenedor de nodos, devuelve el número de triángulos indexado por nodo (diccionario). Si nodes es un nodo específico, devuelve el número de triángulos para el nodo (entero).

(Fuente: [17])

G. Comunidades de modularidad ávidas (Greedy modularity communities)

Detecta comunidades en un grafo utilizando la maximización ávida de modularidad.

`greedy_modularity_communities(G, weight=None, resolution=1, cutoff=1, best_n=None)`

Parámetros:

- `G` - Grafo NetworkX. El grafo sobre el cual detectar comunidades.
- `weight` - String o None, opcional (por defecto=None). El nombre de un atributo de borde que contiene el valor numérico utilizado como peso. Si es None, entonces cada borde tiene un peso de 1. El grado es la suma de los pesos de las aristas adyacentes al nodo.
- `resolution` - Flotante, opcional (por defecto=1). Si la resolución es menor que 1, la modularidad favorece comunidades más grandes. Si es mayor que 1, favorece comunidades más pequeñas.
- `cutoff` - Entero, opcional (por defecto=1). Un número mínimo de comunidades por debajo del cual se detiene el proceso de fusión. El proceso se detiene en este número de comunidades incluso si la modularidad no está maximizada. El objetivo es permitir al usuario detener el proceso temprano. El proceso se detiene antes del corte si encuentra un máximo de modularidad.
- `best_n` - Entero o None, opcional (por defecto=None). Un número máximo de comunidades por encima del cual el proceso de fusión no se detendrá. Esto obliga a la fusión de comunidades a continuar después de que la modularidad comience a disminuir hasta que queden `best_n` comunidades. Si es None, no se fuerza a continuar más allá de un máximo.

Retorna:

`communities` - Lista. Una lista de frozensets de nodos, uno para cada comunidad. Ordenado por longitud con las comunidades más grandes primero.

(Fuente: [18])

H. Comunidades asincrónicas LPA (Asyn lpa communities)

Este algoritmo ofrece una manera eficaz de identificar agrupaciones de nodos en un grafo mediante un proceso de propagación de etiquetas que no requiere una sincronización global entre los nodos. Su naturaleza probabilística permite la detección de comunidades con resultados variables en diferentes ejecuciones.

`asyn_lpa_communities(G, weight=None, seed=None)`

Parámetros:

- G - Grafo.

- *weight* - Cadena. El atributo de borde que representa el peso de un borde. Si es `None`, se asume que cada borde tiene un peso de uno.
- *seed* - Entero, `random_state` o `None` (por defecto). Indicador del estado de generación de números aleatorios. Consulta Randomness.

Retorna:

communities - Iterable. Iterable de comunidades proporcionadas como conjuntos de nodos.

(Fuente: [19])

I. Número núcleo (Core Number)

Devuelve el número núcleo para cada nodo.

Un k-núcleo es un subgrafo maximal que contiene nodos de grado k o más.

El número núcleo de un nodo es el valor k más grande de un k-núcleo que contiene ese nodo.

`core_number(G)[source]`

Parámetros:

- `G` - Grafo NetworkX. Un grafo no dirigido o dirigido.

Retorna:

`core_number` - Diccionario. Un diccionario indexado por nodo con el número núcleo.

(Fuente: [20])

Estas métricas estructurales del grafo proporcionan información valiosa sobre la posición y el rol de los nodos dentro de la red, lo que permite entrenar modelos de aprendizaje automático para predecir la etiqueta objetivo de los nodos de manera efectiva.

IV. Descripción de los Algoritmos de Aprendizaje Automático Utilizados

En este trabajo, hemos empleado tres algoritmos de aprendizaje automático: Naive Bayes, árbol de decisión y k-vecinos más cercanos (k-NN), lo cuales se han podido entrenar gracias a la información sacada de [5], [7], [8], [9], [10] y [11]. A continuación, se describen estos algoritmos y los métodos utilizados para optimizar sus hiperparámetros y evaluar su desempeño.

A. Naive Bayes

Naive Bayes es un clasificador probabilístico basado en el teorema de Bayes, que asume que las características son independientes entre sí. Este algoritmo es especialmente útil en situaciones donde la independencia de las características es una aproximación razonable. A pesar de su simplicidad, Naive Bayes puede ser muy efectivo en tareas de clasificación. Para su entrenamiento hemos decidio normalizar todos los atributos, menos ”id node” y también hemos decidido discretizar todos los atributos, menos los de ”name” e ”id node”, con 700 intevalos de igual tamaño.

B. Árbol de Decisión

El árbol de decisión es un modelo predictivo que se utiliza tanto para clasificación como para regresión. Este algoritmo construye un árbol a partir de las características de los datos, donde cada nodo interno representa una prueba en una característica, cada rama representa el resultado de la prueba y cada hoja representa una clase o un valor de regresión. Los árboles de decisión son fáciles de interpretar y visualizar, lo que los hace muy útiles en la práctica. Para su entrenamiento hemos decidio normalizar todos los atributos, menos ”id node”.

C. k-Vecinos Más Cercanos (k-NN)

El algoritmo k-NN es un método de clasificación basado en instancias que clasifica un punto de datos en función de la mayoría de los k vecinos más cercanos en el espacio de características. Este algoritmo es sencillo de implementar y puede ser muy efectivo con la selección adecuada de la métrica de distancia y el valor de k.

D. Optimización de Hiperparámetros

Para encontrar los mejores modelos posibles, hemos utilizado el método de búsqueda por rejilla (Grid Search), para la optimización de hiperparámetros. Este método implica la definición de un conjunto de valores posibles para cada hiperparámetro y la evaluación exhaustiva de todas las combinaciones posibles. Al utilizar Grid Search, hemos podido identificar las configuraciones de hiperparámetros que maximizan el rendimiento de cada algoritmo.

1) Naive Bayes: En el caso del modelo de Naive Bayes, la búsqueda del hiperparámetro consistía en encontrar un buen valor de alpha, donde alpha hace referencia al suavizado de Laplace. Como se ve en el proyecto, dió como resultado que el suavizado de Laplace debía de ser 1.

2) Árbol de decisón: En el caso del modelo del árbol de decisión, la búsqueda del hiperparámetro consistía en encontrar un buen valor para la profundida máxima del árbol y para la cantidad mínima de ejemplos que necesita para particionar. Como se ve en el proyecto, dió como resultado que debía tener una profundidad máxima de 5 nodos y que como mínimo debia de tener 5 ejemplos para particionar.

3) KNN: En el caso del modelo de KNN, la búsqueda del hiperparámetro consistía en encontrar un buen valor para la cantidad de vecinos a tener en cuenta, osease la k, y como debería medirse la distancia, en nuestro caso o euclidea o manhattan. Como se ve en el proyecto, dió como resultado que debiamos tener en cuanta a un vecino y que la distancia debía ser medida como la manhattan.

E. Evaluación del Desempeño

Para evaluar la confianza y el desempeño de los modelos, hemos implementado dos métodos de validación: la validación por retención y la validación cruzada.

Validación por Retención: Este método implica dividir el conjunto de datos en dos partes: un conjunto de entrenamiento, en nuestro caso conformaba un 80 por ciento del dataset, y un conjunto de prueba, el 20 por ciento del dataset. El modelo se entrena con el conjunto de entrenamiento y se evalúa con el conjunto de prueba. Este enfoque es sencillo pero puede ser sensible a cómo se realiza la partición de los datos. Como en nuestro dataset se daba que tenía un 75 por ciento de desarrolladores web y un 25 por ciento de desarrolladores IA, teníamos que tener eso en cuenta a la hora de mirar el nivel de confianza que nos proporcionaba esta técnica.

Validación Cruzada: Para obtener una evaluación más robusta y menos dependiente de la partición específica de los datos, hemos utilizado la validación cruzada k-fold. En este método, el conjunto de datos se divide en k subconjuntos (folds). El modelo se entrena k veces, cada vez utilizando k-1 folds como conjunto de entrenamiento y el fold restante como conjunto de prueba. Los resultados se promedian para obtener una estimación más fiable del rendimiento del modelo. En nuestro caso pusimos que el dataset de dividiera en 10 secciones y como ocurria en la validación por retención habia que tener en cuenta que el dataset no estaba dividido en un 50/50, si no que habia más desarrolladores web que desarrolladores IA.

La combinación de estos métodos de optimización y validación nos ha permitido seleccionar el modelo más adecuado y obtener una evaluación precisa de su desempeño. Los resultados obtenidos nos dicen que el modelo con mejor confianza es del Naive Bayes, que se puede observa que está mejor entrenado que una máquina aleatoria, luego le sigue el árbol de decisiones que también está mejor entrenado que una máquina de alatoria y por último el knn que no nos dió un buen resultado estaba pero entrenado que una máquina aleatoria.

V. Resultados Alcanzados

En este trabajo, para llegar a los resultados que queriamos comprobar, hemos elegido los dos modelos que nos daban una mejor confianza y lo hemos entrenado con distintos dataset, a los cuales le faltaban las mátricas de agrupación, comunidad, núcleo y centralidad. Para asi poder determinar cuales son las métricas más influyentes y necesarias a la hora de entrer nuestros modelos.

A. Modelo de Naive Bayes

El modelo de Naive Bayes, cuando lo entrenamos con el dataset el cual no tenia las métricas de agrupamiento, observamos que la confianza bajó un poco, asi que podemos decir que no son muy importantes para este modelo. Para el dataset donde no tenía las métricas de comunidad, observamos que la confianza bajó considerablemente, asi que podemos decir que estás métrica son muy importantes para este modelo. Para el dateset donde no tenía las métricas de núcleo, observabos que la confianza bajó un poco, asi que es el mismo caso que con las de agrupamiento. Y por último para el dataset son las métricas de centralidad, observamos que la confianza aunmentó, así que podemos decir que estas metricas solo añaden ruido y no son necesarias para este modelo.

B. Modelo de árbol de decisiones

El modelo de árbol de decisiones, cuando lo entrenamos con el dataset el cual no tenia las métricas de agrupamiento, observamos que la confianza subió un poco, asi que podemos decir estas métricas solo añaden ruido y no son necesarias para este modelo. Para el dataset donde no tenía las métricas de comunidad, observamos que la confianza bajó considerablemente, asi que podemos decir que estás métrica son muy importantes para este modelo. Para el dateset donde no tenía las métricas de núcleo, observabos que la confianza se quedó igual, así que podemos decir que estás métricas son un poco indiferentes para este modelo. Y por último para el dataset son las métricas de centralidad, observamos que la confianza aunmentó, así que podemos decir que estas métricas son como las de agrupamiento.

VI. Conclusiones

En este trabajo, tras el estudio de la utilidad de las métricas hemos observado que el modelo de Naive Bayes sigue siendo el que mejor nivel de confianza nos da. También hemos observado que las métricas de comunidad son las más necesarias para el contexto de este problema y asi conseguir un buen rendimiento con los modelos que hemos utilizado para el ánalisis. Por último, hemos visto que las métricas de centralidad lo unico que hacen es provocar ruido en los modelos, empeorando así su nivel de confianza.

Para posteriores análisis sobre este dataset, es muy recomendable que se añadan más métricas de comunidad ya que son bastante importantes a la hora del entrenamiento de nuestros modelos y también recomendamos que no se utilicen métricas de centralidad ya que empeorarán los resultados obtenidos.

Referencias

[1] Información del dataset. [Online]. Available: https://snap.stanford.edu/data/github-social.html. [Accessed: 09-Jun-2024].

[2] Documentación de NetworkX. [Online]. Available: https://networkx.org/documentation/stable/index.html. [Accessed: 09-Jun-2024].

[3] Documentación de Neo4j. [Online]. Available: https://neo4j.com/. [Accessed: 09-Jun-2024].

[4] Repositorio del proyecto. [Online]. Available: https://github.com/davguifer/AprendizajeAtomaticoRelacional-G2. [Accessed: 09-Jun-2024].

[5] Página web de la asignatura. [Online]. Available: https://www.cs.us.es/docencia/aulavirtual/course/view.php?id=19. [Accessed: 09-Jun-2024].

[6] Herramienta web. [Online]. Available: https://chatgpt.com/. [Accessed: 09-Jun-2024].

[7] Documentación de python. [Online]. Available: https://www.python.org/. [Accessed: 09-Jun-2024].

[8] Documentación de scikit-learn. [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 09-Jun-2024].

[9] Documentación de pandas. [Online]. Available: https://pandas.pydata.org/docs/index.html. [Accessed: 09-Jun-2024].

[10] Documentación de NumPy. [Online]. Available: https://numpy.org/. [Accessed: 09-Jun-2024].

[11] Documentación de matplotlib. [Online]. Available: https://matplotlib.org/. [Accessed: 09-Jun-2024].

[12] Información sobre la métrica "Centralidad de grado". [Online]. Available: https://es.wikipedia.org/wiki/Centralidad_de_grado. [Accessed: 09-Jun-2024].

[13] Información sobre la métrica "Centralidad de cercanía". [Online]. Available: https://es.wikipedia.org/wiki/Centralidad_de_cercan%C3%ADa. [Accessed: 09-Jun-2024].

[14] Información sobre la métrica "Centralidad de intermediación". [Online]. Available: https://es.wikipedia.org/wiki/Centralidad_de_intermediaci%C3%B3n. [Accessed: 09-Jun-2024].

[15] Información sobre la métrica "Coeficiente de agrupamiento". [Online]. Available: https://en.wikipedia.org/wiki/Clustering_coefficient. [Accessed: 09-Jun-2024].

[16] Información sobre la métrica "Agrupación cuadrada". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.square_clustering.html#square-clustering. [Accessed: 09-Jun-2024].

[17] Información sobre la métrica "Triángulos". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.triangles.html#triangles. [Accessed: 09-Jun-2024].

[18] Información sobre la métrica "Comunidades de modularidad ávidas". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html. [Accessed: 09-Jun-2024].

[19] Información sobre la métrica "Comunidades asíncronas LPA". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.label_propagation.asyn_lpa_communities.html#networkx.algorithms.community.label_propagation.asyn_lpa_communities. [Accessed: 09-Jun-2024].

[20] Información sobre la métrica "Número núcleo". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.core.core_number.html#networkx.algorithms.core.core_number. [Accessed: 09-Jun-2024].

[21] Información sobre la métrica "Centralidad de grado". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.bipartite.centrality.degree_centrality.html#networkx.algorithms.bipartite.centrality.degree_centrality. [Accessed: 09-Jun-2024].

[22] Información sobre la métrica "Centralidad de cercanía". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.bipartite.centrality.closeness_centrality.html#networkx.algorithms.bipartite.centrality.closeness_centrality. [Accessed: 09-Jun-2024].

[23] Información sobre la métrica "Centralidad de intermediación". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.bipartite.centrality.betweenness_centrality.html#networkx.algorithms.bipartite.centrality.betweenness_centrality. [Accessed: 09-Jun-2024].

[24] Información sobre la métrica "Agrupación cuadrada". [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.square_clustering.html#networkx.algorithms.cluster.square_clustering. [Accessed: 09-Jun-2024].