APRENDIZAJE AUTOMÁTICO RELACIONAL

Ángela López Oliva

dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Anglopoli1@alum.us.es

Alba Gonzalez Pineda

dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Albgonpin@alum.us.es

***Resumen*—Este proyecto presenta una metodología que utiliza inteligencia artificial para predecir propiedades en grafos, aprovechando métricas estructurales como variables clave. Nos centramos en cómo métricas como la centralidad de grado, la cercanía y el coeficiente de agrupamiento de los nodos pueden alimentar modelos de IA para predecir características específicas de los nodos en un grafo. Utilizamos conjuntos de datos que incluyen tanto grafos generados artificialmente como datos reales, extrayendo y aplicando estas métricas como características esenciales.**

Nuestro enfoque demostró la capacidad efectiva de los modelos de IA, como Naive Bayes y árboles de decisión, para aprender y predecir con alta precisión las propiedades del grafo basándose exclusivamente en métricas estructurales. Discutimos los desafíos encontrados obtenidos. En resumen, este estudio valida el uso de métrica estructurales de grafos como una metodología robusta y eficiente para la predicción precisa de propiedades en contextos complejos de redes.

I. INTRODUCCIÓN

El aprendizaje automático relacional se distingue por su enfoque en desarrollar modelos que aprenden de datos estructurados en forma de grafos, los cuales representan los complejos interacciones entre entidades. A diferencia de enfoques convencionales, este método aprovecha explícitamente las relaciones entre datos para datos para capturar su naturaleza interconectada y mejorar la capacidad predictiva.

En este proyecto, nuestro objetivo principal fue la clasificación precisa de nodos dentro de un grafo mediante la meticulosa construcción de características relacionales. Este enfoque abarcó varios pasos críticos: primero, adquirir un entendimiento profundo de la teoría de grafos, explorando métricas como la centralidad y la modularidad: segundo, emplear herramientas avanzadas de Python como NetworkX y Scikit-Learn para implementar soluciones; tercero, desarrollar y evaluar múltiples modelos de clasificación que integraran tanto características relacionales como no relacionales; y finalmente, realizar análisis detallados de correlación y ajuste de hiperparámetros para optimizar el rendimiento predictivo de los modelos.

II. PRELIMINARES

El proyecto se basa en el análisis del conjunto de datos musae_git, obtenido de una fuente conocida [1], que representa una red compuesta por nodos que representan usuarios y enlaces que indican las interacciones entre estos usuarios

A. Métodos empleados

El conjunto de datos está estructurado en 2 archivos principales:

1. Musae_git_edges.csv: Este archive contiene las aristas del grafo, que reflejan las interacciones entre los usuarios
    a. Id_1: Identificador del primer nodo (usuario)
    b. Id_2: Identificador del segundo nodo(usuario)
2. Musae_git_target.csv: Este archivo proporciona información detallada sobre los nodos, incluyendo el nombre de usuario y la etiqueta objetivo(ml_target) utilizada para la predicción.
    a. Id: Identificador del nodo(usuario)
    b. name: Nombre del usuario
    c. ml_target: Etiqueta de clasificación , donde 0 y 1 representan las categorías a predecir

B. Trabajo Relacionado

El objetivo principal del proyecto es clasificar los nodos del grafo en las dos categorías mencionadas (0 y 1), utilizando métricas estructurales del grafo como características predictoras.

Como Método se emplearán diversos modelos de aprendizaje automático supervisado, incluyendo Naive Bayes, K-Nearest NeighBors (kNN), y Árboles de decisión , para entrenar y evaluar la capacidad predictiva de los modelos

La ejecución de este proyecto implica calcular métricas estructurales específicas del grafo, como la centralidad del grafo, la cercanía , el coeficiente de agrupamiento…. Estas métricas se utilizarán como características de entrada para entrenar los modelos, evaluando su rendimiento mediante

métricas estándar como el recall y analizando las matrices de confusión para comparar y seleccionar el modelo más adecuado

III. METODOLOGÍA

Esta sección, describimos el método implementado en el trabajo para la clasificación de nodos en un grafo utilizando características relacionales. Nuestro enfoque se basa en el uso de métricas estructurales del grafo como características de entrada para diversos modelos de aprendizaje automático supervisado

A continuación, detallamos los pasos clave del proceso:

A. Extracción de Métricas Estructurales

El primer paso consiste en calcular varias métricas estructurales para cada nodo en el grafo. Estas métricas capturan propiedades importantes de la posición y conectividad de los nodos dentro de la red.
Métricas calculadas:

1. Centralidad de Cercanía (Closeness_Centrality):
   Evalúa la proximidad de un nodo a todos los demás nodos del grafo

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

   Donde d(u,v) es la menor distancia entre u y v
   `closeness_centrality(Grafo)`

   Acción:
   Calcular la centralidad de cercanía para cada nodo en el grafo **'Grafo'**. Específicamente la centralidad de cercanía se define como la inversa de la suma de las distancias mas cortas desde un nodo hasta todos los demás nodos en el grafo.
   Entrada:
   - Grafo**:** Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
   Proceso:
   -La función calcula la longitud del camino más corto desde cada nodo hasta todos los demás nodos en el grafo.
   -Para cada nodo, se suma la longitud de todos estos caminos más cortos.
   -La centralidad de cercanía para un nodo se calcula como la inversa de esta suma, normalizada por el numero de nodos en el grafo menos uno.
   Retorno:
   -La función retorna un diccionario **closeness** donde:
   -Las claves son los id de los nodos del grafo
   -Los valores son los valores de la centralidad de cercanía correspondiente a cada nodo
   (Fuente: [2])

2. Centralidad de Intermediación (Betweenness_Centrality):

Mide cuantas veces un nodo actúa como puente a lo largo del camino más corto entre dos otros nodos.

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

Donde :
-bjk es el numero de caminos mas cortos desde el nodo j hasta el nodo k
-bjik es el número de caminos mas cortos desde j hasta k que pasan a traves del nodo i
`betweenness_centrality(Grafo)`

Acción:
Calcular la centralidad de intermediación para cada nodo en el grafo **'Grafo'**. Específicamente, la centralidad de intermediación se define como la fracción de todos los caminos mas cortos en el grafo que pasan por un nodo determinado.
Entrada:
- Grafo**:** Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
-La función identifica todos los caminos más cortos entre cada par de nodos en el grafo
- Para cada nodo, se cuenta cuantos de estos caminos pasan a través de dicho nodo
-La centralidad de intermediación se calcula como la fracción de todos estos caminos que pasan por el nodo
Retorno:
-La función retorna un diccionario 'betweenness' donde:
-Las claves son los id de los nodos del grafo
-Los valores son los valores de la centralidad de intermediación correspondiente a cada nodo
(Fuente[3])

3. Centralidad de Grado (Degree_Centrality)

Mide el número de conexiones directas que tiene un nodo

$$C_D(v) = \delta(v)$$

Donde $\delta(v)$ representa el número de aristas indicentes en el nodo v
`degree_centrality(Grafo)`

Acción: Calcular la centralidad de grado para cada nodo en el grafo '**Grafo'**. Específicamente, la centralidad de grado se define como el número de aristas incidentes en un nodo
Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
-La función cuenta el número de aristas incidentes en cada nodo

Retorno:
-La función retorna un diccionario **degree** donde:
-Las claves son los id de los nodos del grafo
-Los valores son os valores de la centralidad de grado correspondiente a cada nodo
(Fuente: [4])

4. Coeficiente de Agrupamiento(Clustering_Coefficient)

Mide la proporción de triángulos completos en los que está involucrado un nodo

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

Donde Ci es el coeficiente de agrupamiento local del vértice v; ki es el grado del vértice vi, es decir, el número de vecinos que tiene; Ni es el "vecindario" del vertice vi, que consiste en todos los vertices conectados directamente a vi; y donde ejk representa el número de enlaces entre los vértices en el vecindario Ni

`clustering(Grafo)`

Acción:
Calcular el coeficiente de agrupamiento para cada nodo en el grafo '**Grafo'**. Específicamente, el coeficiente de agrupamiento se define como la proporción de triángulos que pasan por un nodo dividido por el número de posibles triángulos que podrían pasar por ese nodo
Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
-La función identifica todos los triángulos que pasan por cada nodo
-Calcula la proporción de estos triángulos con respecto al número máximo posible de triángulos
Retorno:
-La función retorna un diccionario **clustering** donde:
-Las claves son los id de los nodos del grafo
-Los valores son los valores del coeficiente de agrupamiento correspondiente a cada nodo
(Fuente:[5])

5. Agrupamiento de Triángulos

Mide el número de triángulos en los que participa cada nodo

`triangles(Grafo)`

Acción:
Calcular el número de triángulos en los que participa cada nodo en el grafo 'Grafo'
Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
La función cuenta el número de triángulos en los que cada nodo participa
Retorno:
-La función retorna un diccionario **triangles** en los que cada nodo participa
-Las claves son los id de los nodos del grafo
-Los valores son los valores del número de triángulos correspondientes a cada nodo
(Fuente:[6])

6. Agrupamiento de Cuadrados (Squares)

Aunque no es una función directa en NetworkX, se puede calcular la participación de un nodo en estructuras cuadradas

`square_clustering(Grafo)`

Acción:
Calcular el número de cuadrados en los que participa cada nodo en el grafo '**Grafo**'
Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
La función identifica y cuenta el número de cliques de tamaño 4 en los que cada nodo participa
Retorno:
-La función retorna un diccionario **squares** donde:
-La clave son los Id de los nodos del grafo
-Los valores son los valores del número de cuadrados correspondientes a cada nodo
(Fuente[7])

7. Núcleos (K-Core)

Identifica los subgrafos máximos en los que cada nodo tiene al menos un grafo k

`core_number(Grafo)`

Acción:
Calcular el número de núcleo (core number) para cada nodo en el grafo '**Grafo'**. El núcleo de un nodo es el mayor valor de k para el cual el nodo pertenece a un k-nucleo
Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
-La función encuentra el mayor subgrafo tal que cada nodo en el subgrafo tiene al menos grado k
-Calcula el número de núcleo para cada nodo
Retorno:
-La función retorna **k_core** donde:
-Las claves son los id de los nodos del grafo
-Los valores son los números de núcleo correspondiente a cada nodo.
(Fuente:[8])

8. Comunidades (Greedy Modularity)

Identifica comunidades dentro del grafo utilizando la maximización de la modularidad.

`greedy_modularity_communities(Grafo)`

Acción:
Detectar comunidades en el grafo '**Grafo'** utilizando el algoritmo de maximización de modularidad. Una comunidad

es un grupo de nodos que están más densamente conectados entre sí que con el resto del grafo.

Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.

Proceso:
-La función utiliza un algoritmo para maximizar el modularidad, lo que implica agrupar nodos en comunidades de manera que la densidad de las conexiones internas a las comunidades sea mayor que la de las conexiones externas.
Retorno:
-La función retorna una lista de conjuntos **comunidades** donde:

-Cada conjunto contiene los nodos pertenecientes a una comunidad

(Fuente: [9])

9. Partición asíncrona de etiquetas (Asyn_LPA)

Divide el grafo en comunidades utilizando el algoritmo de propagación de etiquetas

```
asyn_lpa_communities(Grafo)
```

Acción:
Detectar comunidades en el grafo **Grafo** utilizando el algoritmo de propagación de etiquetas. Este método es asíncrono y puede encontrar particiones eficientemente en grafos grandes.
Entrada:
-Grafo: Un objeto de tipo 'networkx.Graph' que representa el grafo cuyos nodos y conexiones se quieren analizar.
Proceso:
-La función asigna etiquetas a cada nodo, las cuelaes se propagan iterativamente a los nodos vecinos hasta que se alcanza una condición de estabilidad.
-Agrupa los nodos con la misma etiqueta en comunidades
Retorno:
-La función retorna una lista de conjuntos **asyn_lpa** donde:

-Cada conjunto contiene los nodos pertenecientes a una comunidad

(Fuente:[10]

B. Algoritmos de Aprendizaje Automático Utilizados

En este proyecto, hemos empleado diversos algoritmos de aprendizaje automático para analizar y clasificar datos provenientes de grafos. Estos algoritmos fueron seleccionados por su capacidad para manejar datos relacionales y estructurados, proporcionando herramientas efectivas para la clasificación y agrupación dentro del contexto de redes complejas representadas como grafos. A continuación, se detallan los algoritmos utilizados:

1. Árbol de decisión

Descripción: Los árboles de decisión son modelos de aprendizaje automático no lineales que utilizan una estructura de árbol para dividir el espacio de características en regiones y realizar decisiones de clasificación
Aplicación: Se implementaron para la clasificación de nodos utilizando métricas estructurales y relaciones derivadas del grafo, hemos normalizado todos los atributos menos 'id'
(Fuente: [11])

2. Naive Bayes

Descripción: Naive Bayes es un clasificador probabilístico basado en el teorema de Bayes con una suposición "ingenua" de independencia condicional entre las características.
Aplicación: Ha sido utilizado para la clasificación de nodos en función de las características estructurales derivadas del grafo. Normalizamos todos los nodos menos 'id', además hemos discretizado todos los atributos menos 'id' y 'name'.
(Fuente: [12])

3. K-Nearest Neighbors (KNN)

Descripción: KNN es un método de clasificación que estima la función de densidad de probabilidad de las clases por mayoría de votos de los k ejemplos de entrenamiento más cercanos en el espacio de características
Aplicación: Se empleó para la clasificación de nodos en base a las características estructurales calculadas.
(Fuente: [13])

C. Hiperparámetros calculados

1. Hiperparámetros Árbol de Decisión
   -Profundidad máxima del árbol: Controla la profundidad máxima del árbol para evitar sobreajuste
   -Número mínimo de muestras por hoja: Establece el número mínimo de muestras requeridas para estar en una hoja
   (Fuente: [14])

2. Hiperparámetros Naive Bayes
   -Naive Bayes no suele tener hiperparámetros significativos para ajustar en términos de complejidad del modelo, en nuestro caso, ha consistido en la búsqueda del valor en referencia al suavizado de Laplace
   (Fuente: [15])

3. Hiperparámetros KNN
   -Número de vecinos (k): Define cuántos vecinos considerar para la clasificación
   -Método de distancia: Puede incluir distintas métricas como la distancia euclídea, manhattan…
   (Fuente: [16])

D. Validaciones

Para evaluar de manera exhaustiva la confianza y el desempeño de nuestros modelos predictivos, hemos implementado dos métodos fundamentales de validación: la validación por retención y la validación cruzada. Estas técnicas son esenciales para asegurar que nuestros modelos sean capaces de generalizar bien con datos no vistos y para proporcionar una evaluación robusta de su capacidad predictiva

1. Validación por retención

Esta técnica divide el conjunto de datos en dos partes principales: un conjunto de entrenamiento y un conjunto de prueba. El conjunto de entrenamiento se utiliza para entrenar el modelo, mientras que el conjunto de prueba se reserva exclusivamente para evaluar su rendimiento. Esta técnica es rápida de implementar y útil para conjuntos de datos grandes, aunque puede estar sujeta a variaciones dependiendo de cómo se haya realizado la participación inicial de los datos

2. Validación cruzada

La validación cruzada es una técnica más sofisticada que divide repetidamente el conjunto de datos en diferentes subconjuntos de entrenamiento y prueba. Proporciona una evaluación mas robusta del modelo al promediar los resultados de todas las iteraciones, lo que ayuda a mitigar el impacto de la participación inicial de los datos en la evaluación del modelo.

IV. Resultados

Para alcanzar los resultados deseados, analizamos los modelos que hemos entrenado con distintos conjuntos de datos, a los cuales se les omitieron las métricas de agrupamiento, comunidad, núcleo y centralidad. De esta manera, pudimos determinar cuales son las mejores métricas y necesarias a la hora de entrenar nuestros modelos:

A. Modelo de Naive Bayes

El modelo de Naive Bayes fue entrenado con varios conjuntos de datos modificados para evaluar la importancia de diferentes métricas

1.Dataset sin métricas de agrupamiento: La confianza del modelo subió ligeramente, indicando que estas métricas no son muy importantes para este modelo

2.Dataset sin métricas de comunidad: La confianza del modelo disminuyó considerablemente, sugiriendo que estas métricas son cruciales para el rendimiento del modelo

3.Dataset sin métricas de núcleo: La confianza aumentó levemente, similar a las métricas de agrupamiento, indicando que estas métricas no son tan esenciales

4.Dataset sin métricas de centralidad: la confianza aumentó, lo que implica que estas métricas agregan ruido y no son necesarias para este modelo

B. Modelo de Árbol de Decisión

El modelo de árbol de decisión mostró variaciones en su desempeño al ser entrenado con conjuntos de datos modificados:

1.Dataset sin métricas de agrupamiento: La confianza del modelo no tuvo gran cambio, lo que sugiere que estas métricas solo agregan ruido y no son necesarias

2.Dataset sin métricas de comunidad: La confianza del modelo disminuyó significablemente, indicando que estas métricas son muy importantes para este modelo

3.Dataset sin métricas de núcleo: La confianza disminuyó, lo que sugiere que estas métricas son indiferentes para el modelo

4.Dataset sin métricas de centralidad: la confianza aumentó, indicando que estas métricas también agregan ruido, similar a las métricas de agrupamiento

C. KNN

El modelo Knn no mostró casi ninguna variación en su desempeño al ser entrenado con conjuntos de datos modificados:

1.Dataset sin métricas de agrupamiento: Al eliminar las métricas de agrupamiento podemos observar que se mantuvo constante por tanto estas métricas, solo agregaron ruido

2.Dataset sin métricas de comunidad: Las métricas de comunidad tampoco suponen un gran cambio en en el modelo

3.Dataset sin métricas de núcleo: En referente a las métricas de núcleo su impacto es más crucial que las demás.

4.Dataset sin métricas de centralidad: Al remover las métricas de centralidad disminuye, pero un poco

V. Conclusiones

Basándonos en los resultados obtenidos, el modelo más prometedor entre los evaluados parece ser Naive Bayes, respaldándonos de ciertas razones:

1. Eficiencia y Rapidez: Naive Bayes es altamente eficiente computacionalmente y suele tener tiempos de entrenamiento y predicción muy rápidos en comparación con otros modelos más complejos
2. Alto porcentaje de confianza: Ha demostrado un rendimiento sólido con un recall promedio de aproximadamente 74.50%, lo cual indica una capacidad robusta para identificar correctamente la clase positiva en el conjunto de datos
3. Capacidad para manejar datos desbalanceados: Es efectivo incluso cuando hay desbalance entre las clases en el conjunto de datos. Dado que aproximadamente el 75% de los datos de desarrolladores web y 25% de desarrolladores IA, Naive Bayes ha mostrado una capacidad para manejar esta desigualdad y producir resultados significativos.

VI. BIBLIOGRAFÍA

[0] Repositorio Github: https://github.com/angelalop03/AprendizajeAutomaticoRelacional-G1.git

[1] Dataset[Online]: https://snap.stanford.edu/data/github-social.html

[2] Centralidad de cercanía[Online]: https://es.wikipedia.org/wiki/Centralidad_de_cercan%C3%ADa

[3]Centralidad de intermediación[Online]: https://es.wikipedia.org/wiki/Centralidad_de_intermediaci%C3%B3n

[4]Centralidad de Grado[Online]: https://es.wikipedia.org/wiki/Centralidad_de_grado

[5]Coeficiente de agrupamiento[Online]: https://es.wikipedia.org/wiki/Coeficiente_de_agrupamiento

[6]Triángulos[Online]: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.triangles.html#triangles

[7]Squares[Online]: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.square_clustering.html#square-clustering

[8]K-Core[Online]: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.core.core_number.html#networkx.algorithms.core.core_number

[9]Greedy_modularity_communities[Online]: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html

[10] Asyn_lpa_communities[Online]: 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

[11]Árbol de decisión[Online]: https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n

[12] Naive Bayes [Online]: https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo

[13]KNN [Online]: https://www.elastic.co/es/what-is/knn

[14]Hiperparámetros árbol de decision[Online]: https://scikit-learn.org/stable/modules/tree.html

[15]Hiperparametros Naive Bayes[Online]: https://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes

[16]Hiperparámetros KNN[Online]: https://scikit-learn.org/stable/modules/neighbors.html