DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

El Impacto de la Información Relacional en el Aprendizaje Automático: Artistas de Spotify

Javier Fernández Castillo, Manuel Otero Barbasán

13 de junio de 2023

Resumen

Existen conjuntos de datos que pueden ser modelados por un grafo, en ellos se puede aplicar un enfoque relacional en el aprendizaje automático. Para ello se calculan ciertas métricas que ayudan a los modelos de predicción como la centralidad, el coeficiente de clustering o las comunidades (Label Propagation).

Este artículo compara una visión general del conocimiento previo sobre el Aprendizaje automático relacional con el análisis de resultados experimentales en la predicción de la popularidad de artistas de Spotify. Para ello, se parte de unos datos que incluyen además de información relevante como el numero de seguidores, las colaboraciones entre artistas. Con estos datos se modela un grafo del que se sacan las métricas relacionales.

Para realizar predicciones se utilizan los Modelos KNN, árboles de decisión y redes neuronales. Sin embargo se eliminan las métricas relacionales de algunos subconjunto de datos para analizar si estas realmente tienen relevancia. Se llega a la conclusión de que estas métricas dependiendo del dominio del problema, de los atributos y su correlación, además del modelo empleado son en mayor o menor medida de utilidad. Aunque se demuestra que en el problema que aborda este estudio supone una mejora del 2,13 % en la precisión de los resultados a través de árboles de decisión.

Palabras clave: aprendizaje automático relacional, colaboraciones de artistas, predicción, popularidad, spotify, KNN, árbol de decisión, redes neuronales.

Índice

1. Introducción **3**

2. Conjunto de datos y objetivo a predecir **3**
2.1. Datos base . . . 4
2.2. Objetivo de Predicción . . . 4
2.3. Atributos seleccionados para el entrenamiento . . . 5
2.4. Procesamiento de datos . . . 5
2.5. Importancia de los atributos a priori . . . 6
2.6. División de datos de entrenamiento y evaluación . . . 7

3. Métricas relacionales empleadas **7**
3.1. Coeficiente de clustering . . . 7
3.2. Centralidad . . . 8
3.3. Comunidades con el algoritmo de Label Propagation . . . 9

4. Variantes de datos de entrenamiento **9**
4.1. Todos los atributos (drop([])) . . . 10
4.2. Todos los atributos menos followers (drop(['followers'])) . . . 10
4.3. Atributos no relacionales (drop(['centrality','clustering','community'])) . . . 10
4.4. Atributos relacionales (drop(['followers','country-hits-count'])) . . . 10

5. Modelos empleados **11**
5.1. KNN . . . 11
5.1.1. Definición . . . 11
5.1.2. Aplicación . . . 12
5.1.3. Estudio del valor de los hiper-parámetros . . . 12
5.2. Árboles de decisión . . . 14
5.2.1. Definición . . . 14
5.2.2. Aplicación . . . 15
5.2.3. Estudio del valor de los hiper-parámetros . . . 15
5.3. Redes neuronales . . . 16
5.3.1. Definición . . . 16
5.3.2. Aplicación . . . 17
5.3.3. Estudio del valor de los hiper-parámetros . . . 17

6. Análisis de resultados **18**
6.1. KNN . . . 18
6.2. Árboles de decisión . . . 21
6.3. Redes neuronales . . . 23

7. Conclusiones **27**

8. Bibliografía **28**

1. Introducción

El aprendizaje automático relacional es un enfoque del aprendizaje automático basado en extraer información contextual y estructuralmente relevante de las relaciones entre diferentes entidades en un conjunto de datos modelable como un grafo. Este enfoque permite capturar patrones complejos y conexiones significativas entre entidades, lo que ayuda, en ciertos problemas, a mejorar la precisión de la capacidad predictiva de los modelos de aprendizaje automático.

Al extraer y analizar las interacciones y dependencias entre entidades, el aprendizaje automático relacional puede proporcionar una comprensión más profunda de los datos y permitir la toma de decisiones más informada en una amplia gama de aplicaciones, como recomendación de productos, análisis de redes sociales, bioinformática, etc.

El objetivo de este estudio es investigar la relevancia de las métricas relacionales aplicadas en un dominio específico. Para calcular estas métricas, es necesario modelar los datos como un grafo. Se utilizan datos de Spotify que incluyen información sobre artistas y sus colaboraciones. En este problema, los artistas se representan como nodos del grafo, mientras que las colaboraciones entre ellos se representan como aristas.

Los atributos relacionales que se van tener en cuenta son: centralidad, coeficiente de clustering y comunidades, aplicando el algoritmo de Label Propagation.

Una vez calculadas las métricas relacionales, son añadidas a ciertas variantes de los datos de entrenamiento de los modelos predictivos. Esto permite analizar el impacto que tienen dichos atributos en la precisión de los resultados.

Se aborda el problema utilizando distintos modelos predictivos: KNN, árbol de decisión y redes neuronales.

En los siguientes apartados, se proporciona una contextualización del atributo objetivo a predecir y se analiza la influencia de los demás atributos en el entrenamiento de los modelos. Además, se presenta un análisis detallado de los modelos seleccionados, así como de la investigación realizada para determinar los valores óptimos de los hiper-parámetros. Finalmente, se valoran los resultados obtenidos en la conclusión.

2. Conjunto de datos y objetivo a predecir

El presente estudio se fundamenta en un conjunto de datos que analiza las colaboraciones entre artistas de la plataforma Spotify, con el objetivo de examinar el impacto de la información relacional en modelos predictivos. A lo largo de este apartado se abordan de manera exhaustiva todos los aspectos relevantes del conjunto de datos, incluyendo su procesamiento, así como la influencia e importancia de cada uno de los atributos sobre el atributo objetivo seleccionado.

2.1. Datos base

Los datos en bruto empleados para la realización de este artículo son dos archivos csv. El primero de ellos corresponde a los **artistas** de Spotify. Por cada artista se conocen los parámetros que se muestran en la siguiente tabla.

	spotify_id	name	followers	popularity	genres	chart_hits
0	4IDiJcOJ2GLCK6p9q5BgfK	Kontra K	1999676.0	72	['christlicher rap', 'german hip hop']	['at (44)', 'de (111)', 'lu (22)', 'ch (31)', ...
1	652XIvIBNGg3C0KIGEJWit	Maxim	34596.0	36	[]	['de (1)']
2	3dXC1YPbnQPsfHPVkm1ipj	Christopher Martin	249233.0	52	['dancehall', 'lovers rock', 'modern reggae', ...	['at (1)', 'de (1)']
3	74terC9ol9zMo8rfzhSOiG	Jakob Hellman	21193.0	39	['classic swedish pop', 'norrbotten indie', 's...	['se (6)']
4	0FQMb3mVrAKlyU4H5mQOJh	Madh	26677.0	19	[]	['it (2)']

A continuación se detallan aspectos importantes de cada uno de ellos, y en apartados posteriores se expondrá qué problemas existieron al cargar estos atributos y como se solucionaron.

- **spotify-id**: Identificador **único** de un artista.
- **name**: Nombre del artista.
- **followers**: Número de seguidores, $followers \in \mathbf{Z}$
- **popularity**: Índice de popularidad, $popularity \in [0, 100]$.
- **genres**: Lista de géneros a los que pertenece una canción. Puede estar vacío. No se puede dar todas las posibilidades por la cantidad de géneros existente.
- **chart-hits**: Lista con la posición en la lista de éxitos de cada país. Puede estar vacío.

El segundo archivo corresponde a las colaboraciones entre artistas, que se identifican con su **spotify-id**, siendo id-0 el origen e id-1 el destino de la arista.

	id_0	id_1
0	0hk4xVujcyOr6USD95wcWb	7Do8se3ZoaVqUt3woqqSrD
1	38jpuy3yt3QIxQ8Fn1HTeJ	4csQIMQm6vI2A2SCVDuM2z
2	6PvcxssrQ0QaJVaBWHD07I	6UCQYrcJ6wab6gnQ89OJFh

2.2. Objetivo de Predicción

Existen análisis acerca de como han cambiado la forma de entender el éxito de los artistas teniendo en cuenta los nuevos servicios de música en streaming [4]. Un Índice de Popularidad (IP) en Spotify puede determinar qué canciones o artistas reciben mayor difusión. [1]

Spotify tiene una escala que se puede consultar a traves de su API a la que llama popularity. Spotify lo define como: *The popularity of the artist. The value will be between 0 and 100, with 100 being the most popular. The artist's popularity is calculated from the popularity of all the artist's tracks.* [6] Es decir, si un artista tiene canciones que son populares, su popularidad aumenta.

Algunas opiniones en páginas de marketing musical sugieren que los artistas con Índice de Popularidad (IP) de más de 20 entran en Release Radar y un con IP de más de 30 entran en Discover Weekly.[1]

Teniendo en cuenta la importancia que se le da a este parámetro se cree conveniente utilizarlo como objetivo a predecir. Además, tiene relación tanto con el número de seguidores como con las colaboraciones, lo que nos permite analizar la diferencia entre el aprendizaje automático relacional y no relacional. Estudios previos determinan que las colaboraciones afectan al número de seguidores y la popularidad[5]

2.3. Atributos seleccionados para el entrenamiento

De los atributos originales se han descartado algunos y modificado otros para poder entrenar los modelos. Sin embargo, estos atributos pueden ser visualizados con los resultados.

Los atributos que se han descartado son el identificador y el nombre del artista, ya que estos son únicos para cada linea de datos y no aportan información relevante al problema.

Al quitar la columna con la lista de géneros y añadir una columna por cada género con un valor binario, no se detectó mejoría en las predicciones con las pruebas realizadas, ya que añadían un total de 3000 columnas. Por tanto se ha decidido descartar dicho atributo por la complejidad que añadía al procesamiento de datos.

Se ha modificado el atributo de *chart-hits* para que tenga un valor numérico. El atributo derivado de este se llama country-hits-count y cuenta el número de países en los que el artista forma parte de la lista de éxitos.

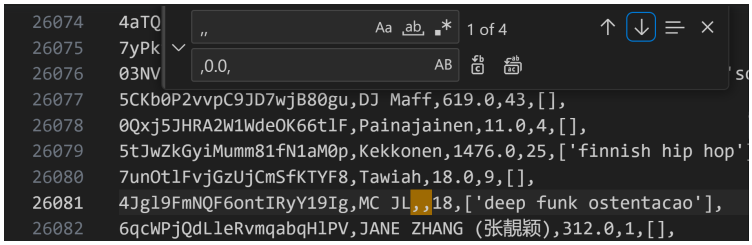
2.4. Procesamiento de datos

Para llegar a los atributos seleccionados para el entrenamiento se ha tenido que procesar los datos de la siguiente manera:

- **Creación del contador de países**: Se han recorrido los datos y en la columna de chart-hits se ha hecho una división por comas. El tamaño de esta división se ha añadido a la columna del nuevo atributo derivado. Es decir, se cuenta el número de países en los que el artista forma parte de la lista de éxitos.

	chart_hits	country_hits_count
1	['de (1)']	1
2	['at (1)', 'de (1)']	2
3	['se (6)']	1

- **Eliminación de duplicados**: En los datos de entrada se encontraron ciertas lineas con los identificadores o nombre de artistas repetidos. Esto no es compatible con nuestro modelo de datos por lo que se descartaron estas lineas.
- **Subsanación de datos nulos**: En los archivos originales existían lineas con columnas inexistentes. Se asignaron a esas casillas corruptas valores **"0"** si el atributo es numérico o **"[]"** si contenía una lista.



- **Normalización de datos**: Antes de entrenar el modelo se deben normalizar los datos para que todos los atributos tengan el mismo peso. Si esto no fuera así, followers, que puede tomar valores de gran tamaño podría tener mayor influencia que el numero de países o los atributos relacionales. La normalización se ha hecho en rango $featureRange = (0, 100)$

Una vez obtenidos los datos finales tras su procesamiento inicial, restan 156319 nodos de los 156422, es decir, se han descartado 103 nodos. Los datos finalmente tienen este aspecto:

	followers	popularity	centrality	clustering	community	country_hits_count
0	1.957457	72.0	3.593487	6.795635	0.000000	7.042254
1	0.033866	36.0	0.449186	10.714286	0.000000	1.408451
2	0.243971	52.0	2.189781	3.778677	0.008501	2.816901

2.5. Importancia de los atributos a priori

Como se ha comentado anteriormente, la popularidad se calcula en base a la repercusión de las canciones de los artistas. En una primera aproximación al problema, se llega a la conclusión que el número de reproducciones tendrá gran importancia en el comportamiento de los modelos. Sin embargo, una de las limitaciones de este estudio es la falta de un atributo, en los datos brutos, que aporte dicha información. Considerando así, el número de listas de éxito en las que aparece un artista una buena aproximación al número de reproducciones.

Otro atributo que puede tener importancia respecto al objetivo seleccionado es el número de seguidores. Es lógico pensar que un artista que tiene canciones exitosas (IP alto), tendrá una exposición mayor y por tanto mayor número de seguidores.

La comunidad y el coeficiente de clustering también tienen relevancia, sin embargo pueden ser de mayor utilidad a la hora de resolver otros problemas, como predecir la familia de géneros en los que participa un autor.

Por ultimo otro atributo a tener en cuenta es la centralidad, ya que esta determina como este artista se relaciona con otros. Aunque existen artistas que trabajan en solitario, estudios previos determinan que las colaboraciones entre artistas afectan al numero de seguidores y la popularidad de estos[5].

2.6. División de datos de entrenamiento y evaluación

En este estudio, se ha optado por utilizar la estrategia de *train/test* para la división de los datos en conjuntos de entrenamiento y prueba.

Una de las razones es que, al contar con una gran cantidad de datos, la probabilidad de seleccionar un conjunto sesgado o poco representativo mediante el *train/test* es prácticamente inexistente. Cuando se dispone de una cantidad sustancial de datos, la variabilidad y la diversidad de las observaciones se distribuyen de manera más equitativa, lo que reduce la posibilidad de introducir sesgos en los conjuntos de entrenamiento y prueba.

Además, el *train/test* es una estrategia estadísticamente sólida que se basa en la aleatoriedad y la imparcialidad en la selección de las muestras. Esto proporciona una base sólida para realizar inferencias y generalizaciones sobre la población completa a partir de los conjuntos de datos de entrenamiento y prueba seleccionados aleatoriamente.

Se han llevado a cabo experimentos con el objetivo de evaluar el impacto del cambio en los conjuntos de entrenamiento y prueba mediante la variación de la semilla utilizada (5 pruebas en total). Al realizar modificaciones aleatorias en estos conjuntos de datos, se ha observado una diferencia máxima del 0,4 % en términos de precisión de los modelos, representando un efecto prácticamente insignificante.

Estos hallazgos indican que la variabilidad introducida por la alteración de la semilla en la selección de los conjuntos de entrenamiento y prueba tiene un impacto mínimo en el rendimiento de los modelos. Incluso en el peor de los casos, la diferencia observada es tan pequeña que resulta casi despreciable en términos prácticos. Por lo tanto, se puede concluir que la elección de diferentes semillas para generar los conjuntos de datos no afecta de manera significativa la capacidad predictiva y la precisión de los modelos utilizados en este estudio, siendo el *train/test* una estrategia válida para nuestro estudio.

El porcentaje aplicado para el *train/test* será 80 % para el conjunto de entrenamiento y 20 % para el conjunto de pruebas.

3. Métricas relacionales empleadas

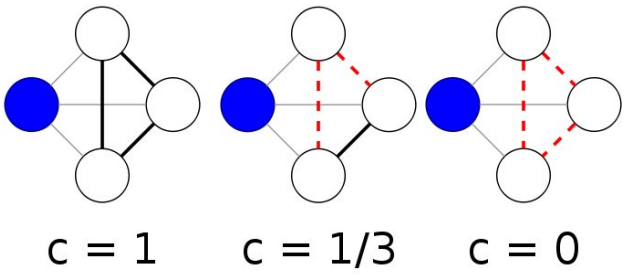
Una vez conocidos los datos de entrada, se pueden calcular ciertas métricas relacionales a raíz del grafo generado. En los anteriores apartados, se ha hecho referencia a estos atributos y a la información que se puede extraer. A continuación, se realiza una descripción teórica de cada uno de ellos y su interpretabilidad en el dominio del problema.

3.1. Coeficiente de clustering

Dado $u \in V$, el Coeficiente de Clustering de u se define como:

$$C(u) = \frac{2 \cdot \#\{(v,w) \in E : \{v,w\} \subseteq \mathcal{N}(u)\}}{gr(u)(gr(u)-1)}$$

Que cuenta la proporcion de tríangulos (ciclos de longitud 3) con un vertice en u que hay en el grafo: de entre todos los posibles triangulos que se pueden formar usando u y los nodos de su entorno, aquellos que realmente estan completos.[2]

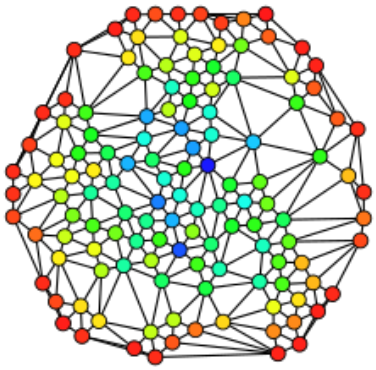


En nuestro problema, el clustering puede ayudar a identificar comunidades de artistas que comparten géneros similares. Además, puede proporcionar información valiosa sobre las interacciones entre artistas populares y menos populares, ya que, es posible que los artistas populares tiendan a colaborar principalmente entre ellos.

3.2. Centralidad

El grado de centralidad de un grafo es una medida que cuantifica la importancia o la centralidad de un nodo en relación con los demás nodos del grafo. Esta medida se basa en el número de aristas conectadas a un nodo específico.

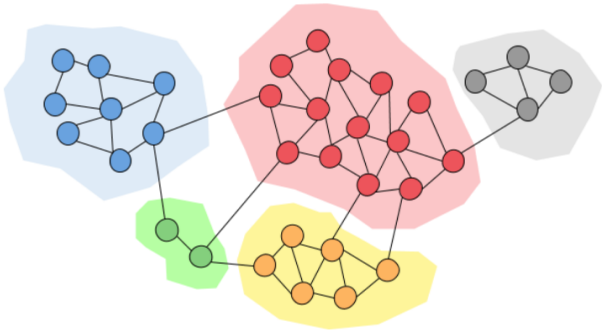
En un grafo no dirigido, el grado de centralidad de un nodo se define como el número de aristas que están conectadas a ese nodo. Cuanto mayor sea el grado de centralidad de un nodo, más conexiones tiene con otros nodos del grafo.



Este atributo, por tanto, toma importancia en la predicción de nuestro atributo objetivo (IP). *Degree centrality is an appropriate measure for a friendship social network if we are interested in looking for the most popular person in the network. The most popular person should have the highest number of friends. Thus, degree centrality is the most appropriate measure.* [3] Es coherente pensar que un artista tiende a ser más popular si tiene muchas colaboraciones con otros artistas.

3.3. Comunidades con el algoritmo de Label Propagation

La detección de comunidades con Label Propagation es un método de análisis de redes que busca identificar grupos o comunidades dentro de un grafo. El algoritmo asigna etiquetas a los nodos y propaga la información entre ellos, actualizando las etiquetas en cada iteración según las etiquetas de los nodos vecinos. Este proceso continúa hasta alcanzar un estado de equilibrio donde las etiquetas ya no cambian significativamente, y los nodos con etiquetas similares tienden a formar comunidades.



En nuestro dominio del problema, la detección de comunidades es ciertamente útil para encontrar patrones de artistas que se dedican a un mismo género. Sin embargo, este atributo no toma una importancia, a priori, relevante en nuestro atributo objetivo (IP).

4. Variantes de datos de entrenamiento

Para realizar un análisis detallado del impacto de atributos relacionales en nuestro conjunto de datos, se realizan 4 variantes del conjunto de atributos para estudiar como se comportan los modelos cuando de descartan y se añaden alguno de ellos.

En los apartados de los modelos empleados y resultados obtenidos se seguirá la misma estructura y orden en el que aparecen las variantes presentadas a continuación.

4.1. Todos los atributos (drop([]))

Para esta variante del conjunto de datos se utilizarán todos los atributos incluyendo relacionales y no relacionales.

	followers	centrality	clustering	community	country_hits_count
0	1.957457	3.593487	6.795635	0.000000	7.042254
1	0.033866	0.449186	10.714286	0.000000	1.408451
2	0.243971	2.189781	3.778677	0.008501	2.816901

4.2. Todos los atributos menos followers (drop(['followers']))

Para esta variante del conjunto de datos se utilizarán todos los atributos incluyendo relacionales y no relacionales menos el atributo followers, con el objetivo de analizar como afecta este atributo a la precisión de los modelos.

	centrality	clustering	community	country_hits_count
0	3.593487	6.795635	0.000000	7.042254
1	0.449186	10.714286	0.000000	1.408451
2	2.189781	3.778677	0.008501	2.816901

4.3. Atributos no relacionales (drop(['centrality','clustering','community']))

Para esta variante del conjunto de datos se utilizarán únicamente los atributos no relacionales.

	followers	country_hits_count
0	1.957457	7.042254
1	0.033866	1.408451
2	0.243971	2.816901

4.4. Atributos relacionales (drop(['followers','country-hits-count']))

Para esta variante del conjunto de datos se utilizarán únicamente los atributos relacionales.

	centrality	clustering	community
0	3.593487	6.795635	0.000000
1	0.449186	10.714286	0.000000
2	2.189781	3.778677	0.008501

5. Modelos empleados

Para sacar conclusiones sobre el impacto de los atributos relacionales en los modelos de predicción, se ataca el problema con 3 modelos distintos: KNN, árboles de decisión y redes neuronales.

Para comprender mejor la estructura de este apartado, se contextualiza a continuación el orden que se seguirá en cada uno de los modelos:

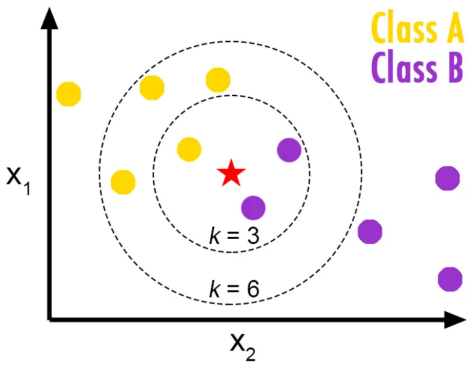
- Definición teórica del modelo.
- Aplicación del modelo a nuestro dominio del problema.
- Estudio del valor de cada hiper-parámetros.

5.1. KNN

5.1.1. Definición

El modelo de k-Nearest Neighbors (k-NN) es un algoritmo de aprendizaje automático supervisado utilizado para la clasificación y regresión. En este modelo, los datos se representan en un espacio multidimensional, donde cada instancia se encuentra en una posición definida por sus atributos.

Para clasificar una nueva instancia en el modelo k-NN, se calcula la distancia entre la instancia desconocida y todas las demás instancias en el conjunto de entrenamiento. Luego, se seleccionan los k vecinos más cercanos según la distancia y se asigna a la nueva instancia la clase mayoritaria entre esos vecinos (en el caso de clasificación) o se realiza una combinación de sus valores de salida (en el caso de regresión).



Los hiper-parámetros con los que se cuenta en k-NN son la k, que representa el número de vecinos más cercanos que se tienen en cuenta durante la predicción; y la métrica de distancia, que puede tomar alguno de los siguientes valores, entre otros, dependiendo problema a atacar:

- **Distancia Euclídea**: Calcula la longitud directa entre dos puntos en un espacio euclidiano, midiendo la similitud basada en la geometría de las características.

- **Distancia Manhattan**: Mide la distancia vertical y horizontal entre dos puntos, sumando las diferencias absolutas en cada dimensión, siendo útil cuando las características tienen unidades diferentes o cuando solo importa la distancia en cada dimensión.

- **Distancia de Hamming**: Utilizada para datos categóricos o binarios, mide la cantidad de dimensiones diferentes entre dos instancias, siendo adecuada para clasificar objetos con características discretas.

5.1.2. Aplicación

Se ha usado k-NN regressor de la librería de scikit-learn, ya que el atributo a predecir es el índice de popularidad, que puede tomar valores entre 0 y 100. Se descarta además el estudio previo de la distancia de Hamming debido a que nuestro conjunto de atributos son, en su totalidad, numéricos y no nos interesa si los valores de cada dimensión cambian o no.

5.1.3. Estudio del valor de los hiper-parámetros

- **K-vecinos**: se estudia para cada una de las variantes cuál es el número de vecinos óptimo para conseguir los mejores resultados.
- **Métrica de distancia**: se estudia para cada una de las variantes cuál es la métrica de distancia más óptima para conseguir los mejores resultados.

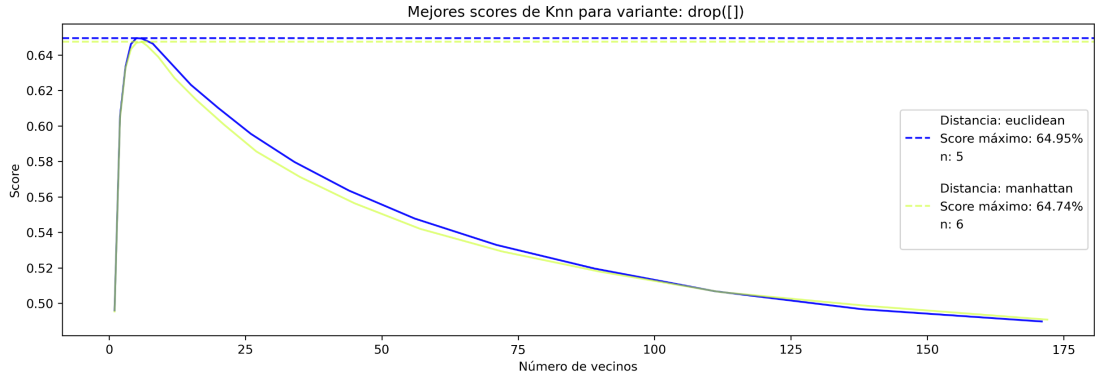
A continuación, se decide para cada una de las variantes los hiper-parámetros óptimos. Se ha utilizado el siguiente algoritmo que va aumentando el número de vecinos progresivamente si no encuentra mejores resultados:

Algorithm 1 allCombinationsKnnForDataSet

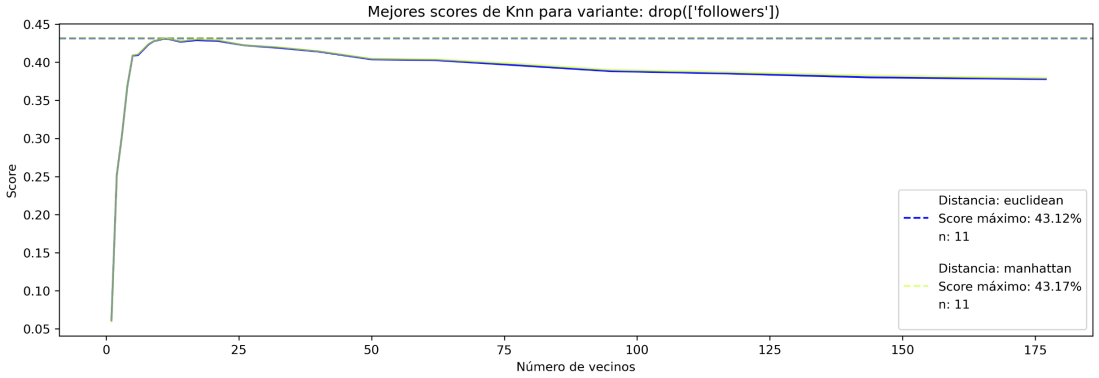
Require: *metrics* es una lista de distancias de knn, $jumpHit$: salto en acierto, $jumpFactorFail$ es el factor por el que se multiplica el salto en caso de fallo, $variantDropping$ es una variante de datos, $nMax$ es el número máximo de vecinos

```
scores ← {}
for m = 0 hasta tamaño(metrics)−1 do
    scores[m] ← []; best ← 0; jump ← 1; n ← 1
    while n < nMax do
        (pred, score) ← knnRegressor(metrics[m], n, variantDropping)
        scores[m] ← ((n, score, pred))
        if score ≥ best then
            jump ← jumpHit; best ← score
        else
            jump ← ⌈jump × jumpFactorFail⌉
        end if
        actualiza valores
        n ← n + jump
    end while
end for
return scores
```

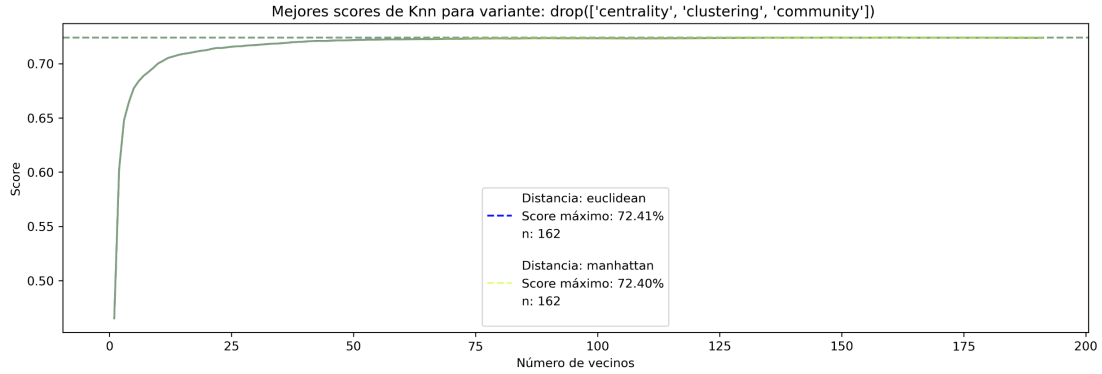
- **Todos los atributos (drop([])):**
  K-vecinos = 5
  Distancia = Euclídea



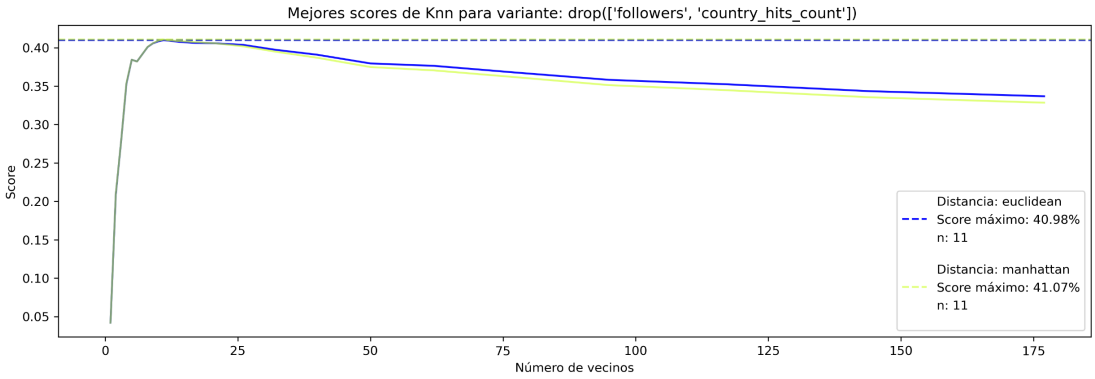
- **Todos los atributos menos followers (drop(['followers'])):**
  K-vecinos = 11
  Distancia = Manhattan



- **Atributos no relacionales (drop(['centrality','clustering','community'])):**
  K-vecinos = 162
  Distancia = Euclídea



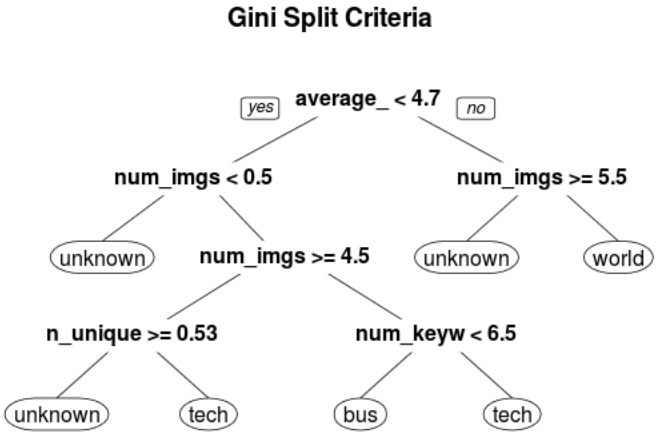
- **Atributos relacionales (drop([’followers’,’country-hits-count’])):**
  K-vecinos = 11
  Distancia = Manhattan



5.2. Árboles de decisión

5.2.1. Definición

Los árboles de decisión son modelos de aprendizaje automático que utilizan una estructura en forma de árbol para tomar decisiones o hacer predicciones. Cada nodo interno del árbol representa una característica o atributo, y las ramas salientes representan las posibles combinaciones de valores para esa característica. Los nodos hoja representan las predicciones o decisiones finales. Los árboles de decisión se construyen a partir de un conjunto de datos de entrenamiento, utilizando algoritmos que buscan la mejor forma de dividir los datos en cada nodo, maximizando la pureza o la reducción de la incertidumbre en la clasificación o predicción.



El hiper-parámetro principal con el que cuenta un árbol de decisión es la profundidad máxima del árbol, que controla la profundidad máxima permitida para el árbol. Limitar la profundidad ayuda a evitar el sobreajuste y mejora la generalización del modelo.

5.2.2. Aplicación

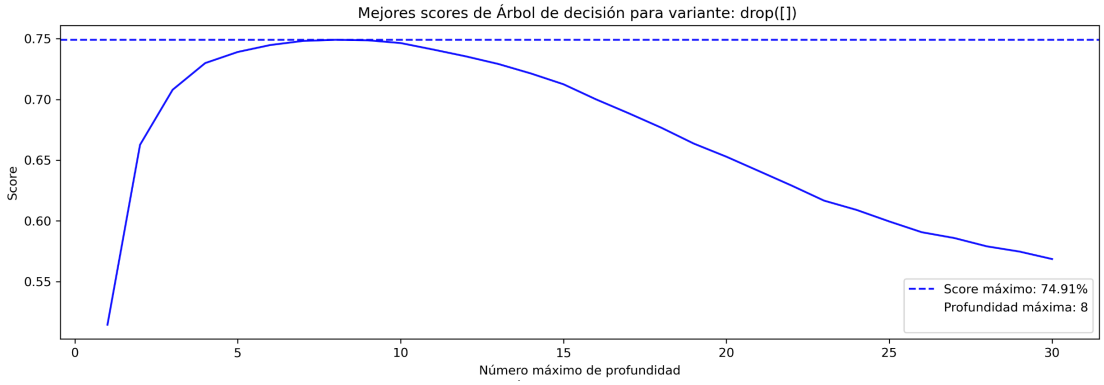
Se ha usado DecisionTreeRegressor de la librería de scikit-learn, ya que el atributo a predecir es el índice de popularidad, que puede tomar valores entre 0 y 100. Se ha probado con los valores del hiper-parámetro que se explican a continuación.

5.2.3. Estudio del valor de los hiper-parámetros

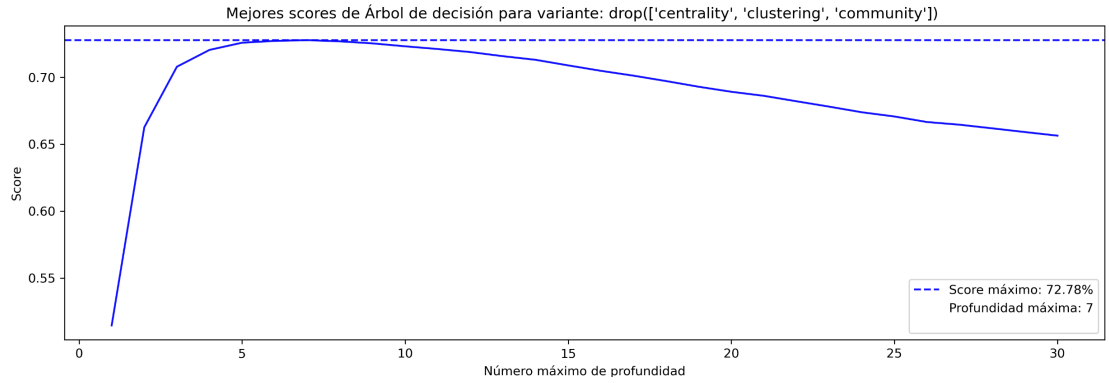
- **Profundidad máxima** : se estudia para cada una de las variantes cuál es la profundidad máxima del árbol óptima para conseguir los mejores resultados. Para ello se ha realizado un bucle que entrena con el hiper-parámetro contenido en un rango: $maxDepth \in [1, 31]$

A continuación, se decide para cada una de las variantes los hiper-parámetros óptimos:

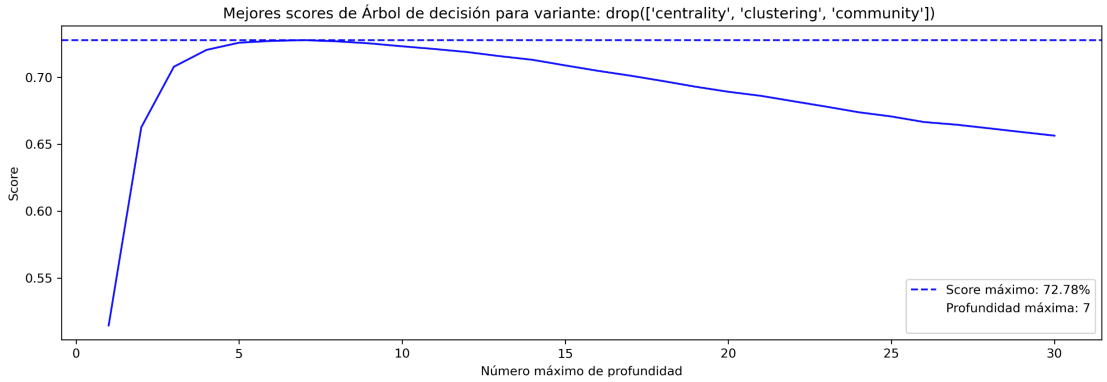
- **Todos los atributos (drop([])):**
  Profundidad máxima: 8



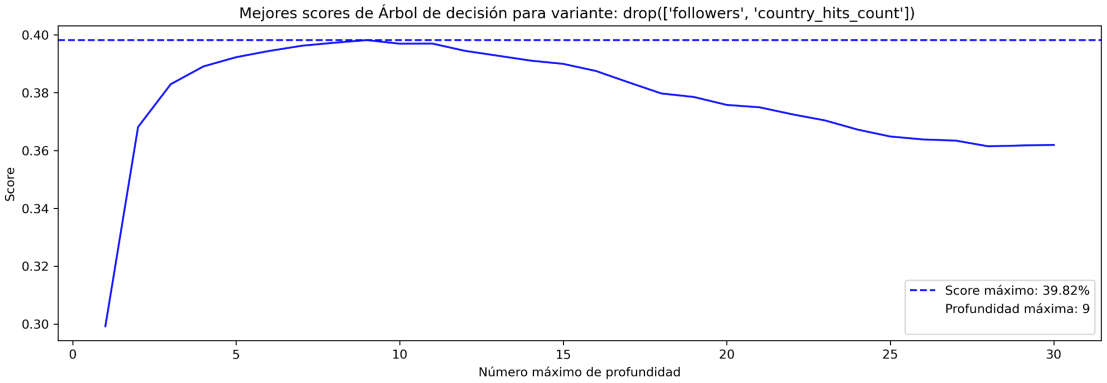
- **Todos los atributos menos followers (drop(['followers'])):**
  Profundidad máxima: 7



- **Atributos no relacionales (drop([’centrality’,’clustering’,’community’])):** Profundidad máxima: 7



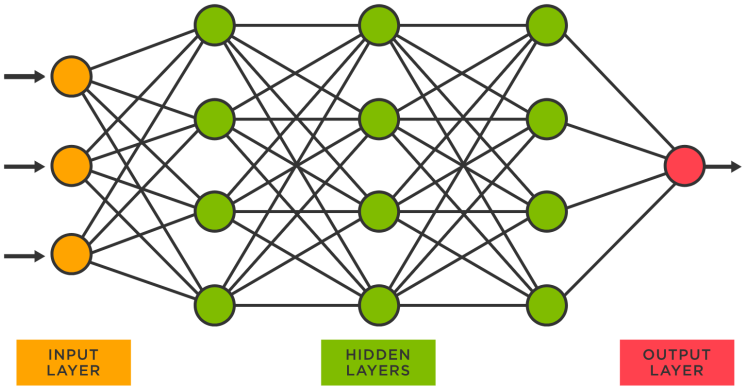
- **Atributos relacionales (drop([’followers’,’country-hits-count’])):** Profundidad máxima: 9



5.3. Redes neuronales

5.3.1. Definición

Una red neuronal es un modelo computacional inspirado en el funcionamiento del cerebro humano, que se utiliza para procesar información y realizar tareas de aprendizaje automático. Está compuesta por neuronas, conectadas entre sí mediante conexiones ponderadas. Además está compuesta por una capa de entrada, una capa de salida y n capas ocultas, $n \in R$. La red neuronal es capaz de aprender y extraer patrones complejos a partir de datos, lo que la hace útil en diversas aplicaciones como reconocimiento de imágenes, procesamiento del lenguaje natural y análisis de datos.



Los hiper-parámetros con los que cuenta una red neuronal, entre otros, son el número de capas, el número de neuronas por cada capa, la función de activación de las capas ocultas y de la capa de salida, el factor de aprendizaje, el tamaño de lotes y las épocas de entrenamiento.

5.3.2. Aplicación

Para la red neuronal que ha utilizado el parquete keras de tensorflow. La capa de entrada(1) salida (8) tienen como función de activación la *identidad*. Cabe mencionar que como se tratan cuatro variedades de datos distintos, la capa de entrada ha cambiado el número de neuronas para coincidir con el numero de atributos que recibe la red en cada caso. En las siete capas intermedias (1-7) se han utilizado un numero de neuronas descendente, a excepción de las capas 1 y 3 que utilizan 2 neuronas y la función de activación *tangente* en lugar de la *identidad*. Para poder medir el ajuste de las predicciones a los datos se ha implementado manualmente el coeficiente de determinación R^2

5.3.3. Estudio del valor de los hiper-parámetros

Se ha realizado un estudio manual para la elección de los hiper-parámetros. Debido a la complejidad de modelar una red neuronal de forma óptima, se ha decidido utilizar los mismos hiper-parámetros para las 4 variantes.

Números de capas ocultas: 7
Número de neuronas por capa oculta: 2, 50, 2, 30, 7, 3, 2
Función de activación de las capas ocultas: tanh, identidad, tanh, identidad, identidad, identidad, identidad
Función de activación de la capa de entrada y de salida: identidad, identidad
Factor de aprendizaje: 0,00008
Número de lotes: 100
Número de épocas de entrenamiento: 20
Optimizador: SGD
Pérdidas: mean-square-error

6. Análisis de resultados

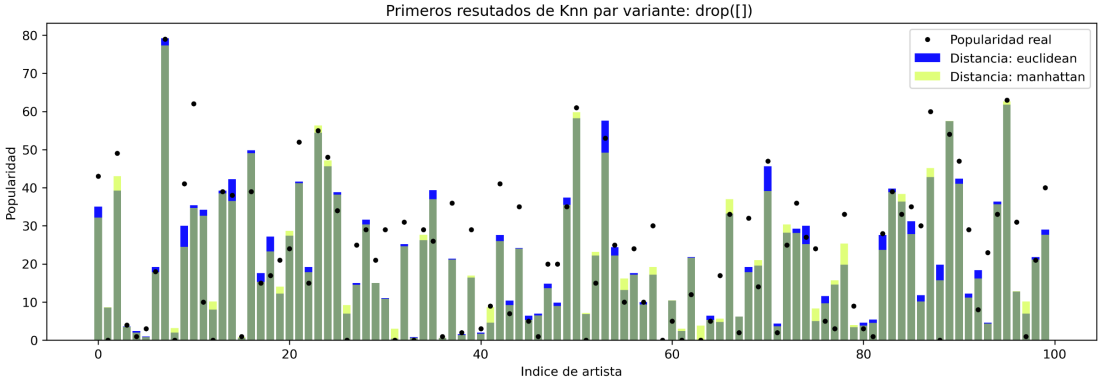
A continuación se exponen los resultados obtenidos con cada uno de las variantes de datos de entrenamiento para los mejores hiper-parámetros de esa variante. Este bloque se divide en 3 apartados, una por cada modelo empleado. Para que sea homogéneo y fácil de comprender se sigue la siguiente estructura en los 3 apartados:

- Resumen de los hiper-parámetros por cada subconjunto de entrenamiento.
- Gráficas de las 100 primeras predicciones con información del valor esperado y la predicción realizada.
- Gráficas con los resultados.
- Comentarios.

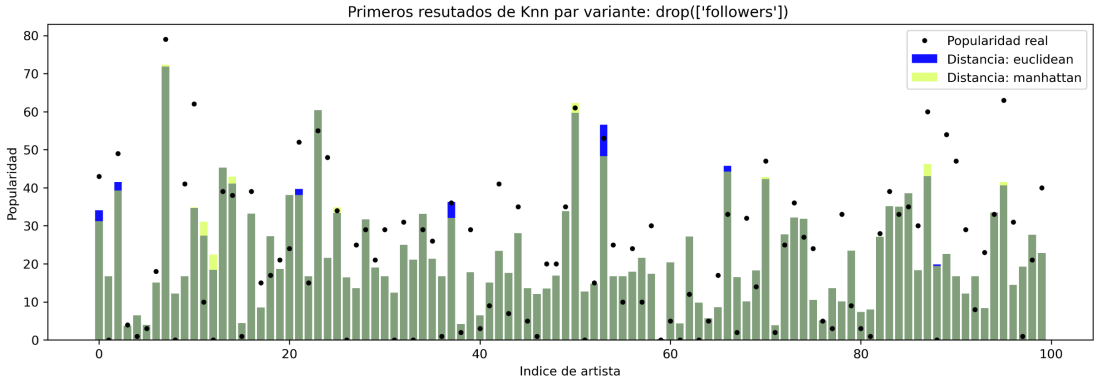
6.1. KNN

A continuación se muestran las 100 primeras predicciones realizadas para cada una de las variantes del problema y lo hiper-parámetros de los resultados finales. En la gráfica se aprecia por cada barra una predicción realizada. Los puntos representan el valor esperado y la barra la predicción realizada. El contorno verde hace referencia a la intersección entre la distancia Euclídea y la de Manhattan. Si un color sobresale por la parte superior de la barra significa que la distancia referente a dicho color predice un valor más alto (no necesariamente mejor).

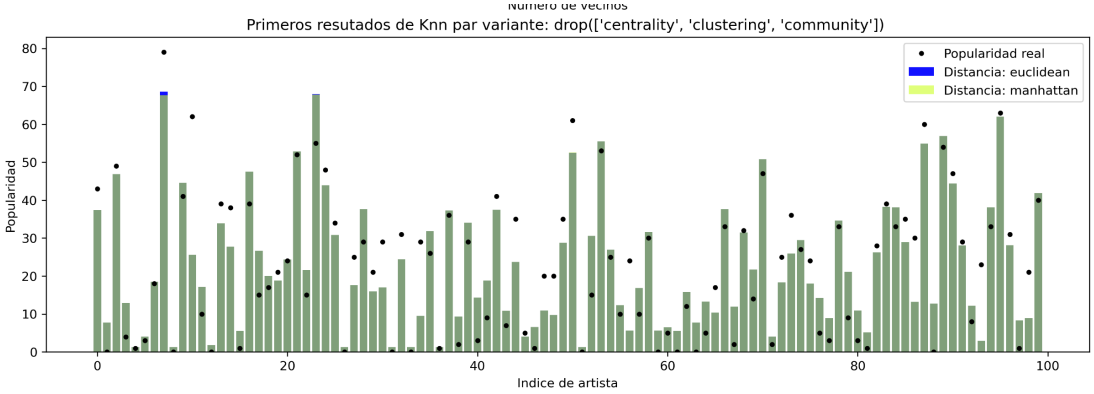
- **Todos los atributos (drop([])):**
  K-vecinos = 5
  Distancia = Euclídea



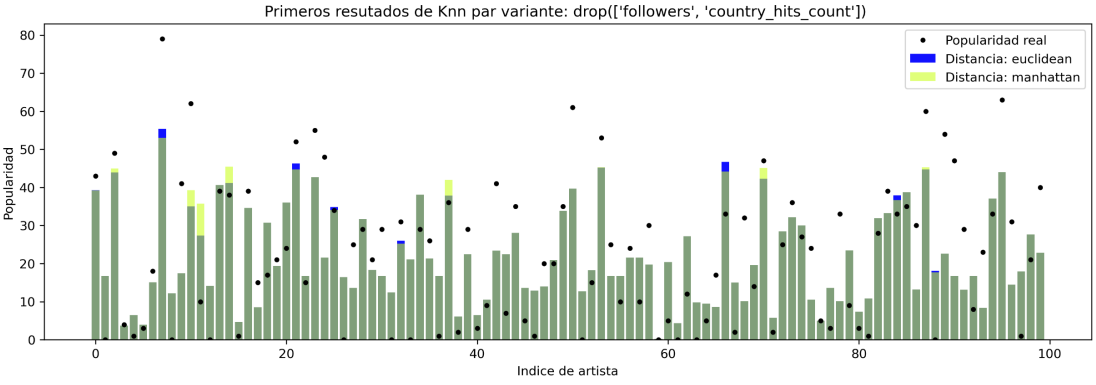
- **Todos los atributos menos followers (drop([’followers’])):**
  K-vecinos = 11
  Distancia = Manhattan



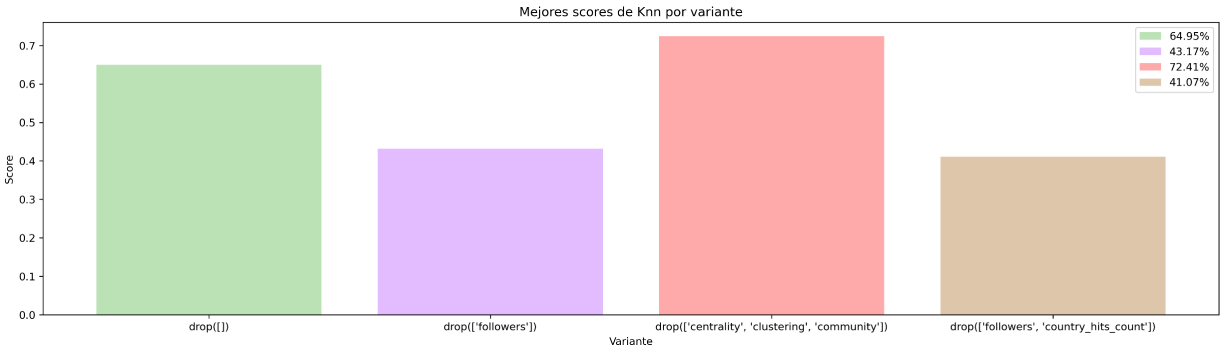
- **Atributos no relacionales (drop([’centrality’,’clustering’,’community’])):**
  K-vecinos = 162
  Distancia = Manhattan



- **Atributos relacionales (drop([’followers’,’country-hits-count’])):**
  K-vecinos = 11
  Distancia = Manhattan



Por último, se muestran a continuación los resultados finales para cada una de las variantes con los mejores hiper-parámetros.



En primer lugar, se puede observar que el mejor resultado se obtiene en la variante de los atributos no relacionales (roja), seguida de la variante que cuenta con todos los atributos (verde). Una de las razones por la que incluir atributos relacionales empeora la capacidad predictiva del modelo, es la correlación que existe entre el atributo followers y popularidad. Al ser estos atributos tan dependientes entre sí, si se incluyen nuevos atributos, los puntos se dispersan en el espacio y no se consigue una mejora sustancial del modelo debido a los atributos relacionales, sino todo lo contrario.

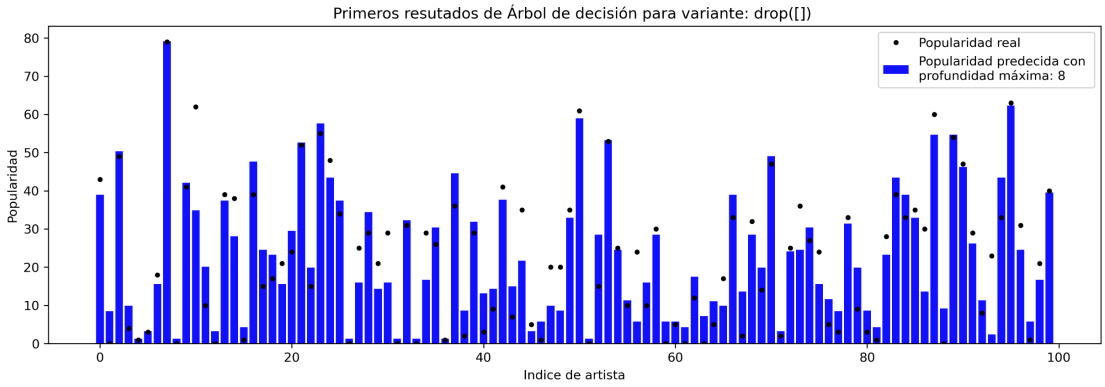
La importancia del atributo followers se puede apreciar en la segunda barra (morada). Al eliminar el atributo followers se puede observar con respecto a la variante de todos los atributos (verde) como empeora de manera significativa. Por otro lado, al eliminar el atributo country charts count (marrón) se observa que el resultado disminuye únicamente en un 2 %, por lo que dicho atributo no es esencial, aunque mejora parcialmente la precisión del modelo. Esta última variante, también nos hace saber que el modelo es capaz de acertar en un 41,07 % de los casos usando solamente atributos relacionales.

Como conclusión, se puede extraer que k-NN se comporta de una forma más óptima descartando los atributos relacionales debido a la correlación que existe ente followers y popularidad. Sin embargo, se saca en claro que la información relacional puede ser muy útil en ciertos contextos, ya que, por sí sola, es capaz de llegar a unos resultados más que válidos. Si no se contara con el atributo followers, el resultado de nuestro modelo sería 43,17 % de precisión.

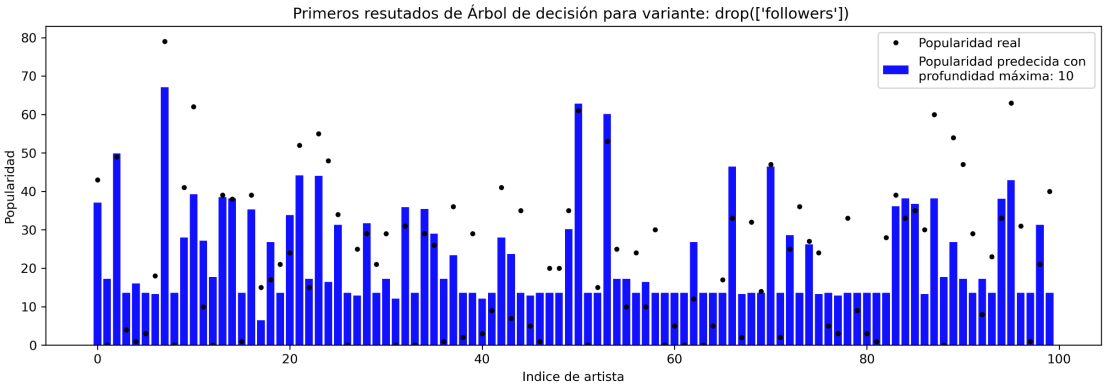
6.2. Árboles de decisión

A continuación se muestran las 100 primeras predicciones realizadas para cada una de las variantes del problema y los hiper-parámetros de los resultados finales. En la gráfica se aprecia por cada barra una predicción realizada. Los puntos representan el valor esperado y la barra la predicción realizada.

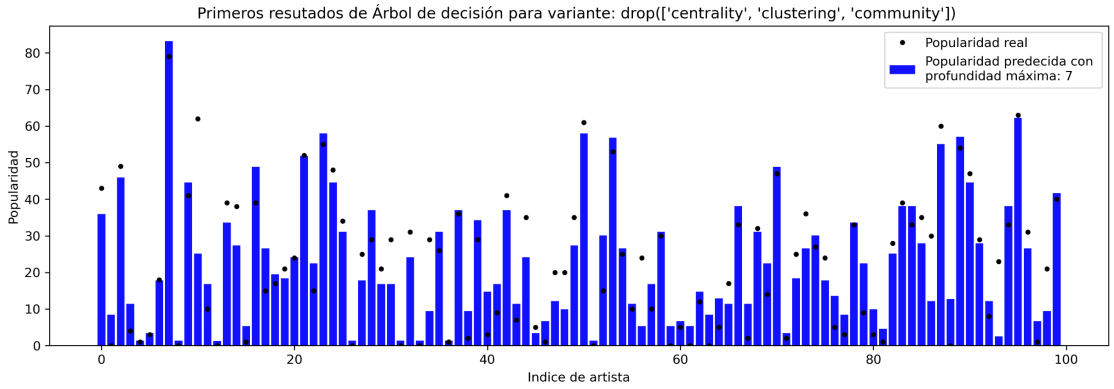
- **Todos los atributos (drop([])):**
  Profundidad máxima: 8



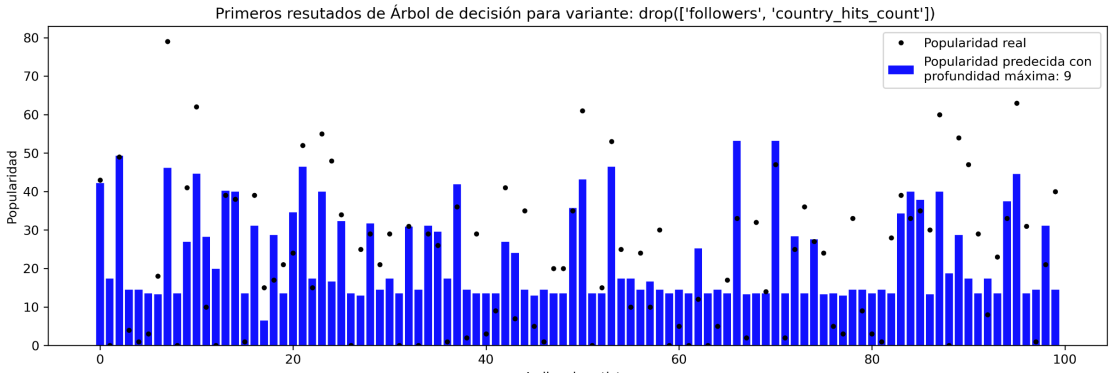
- **Todos los atributos menos followers (drop(['followers'])):**
  Profundidad máxima: 7



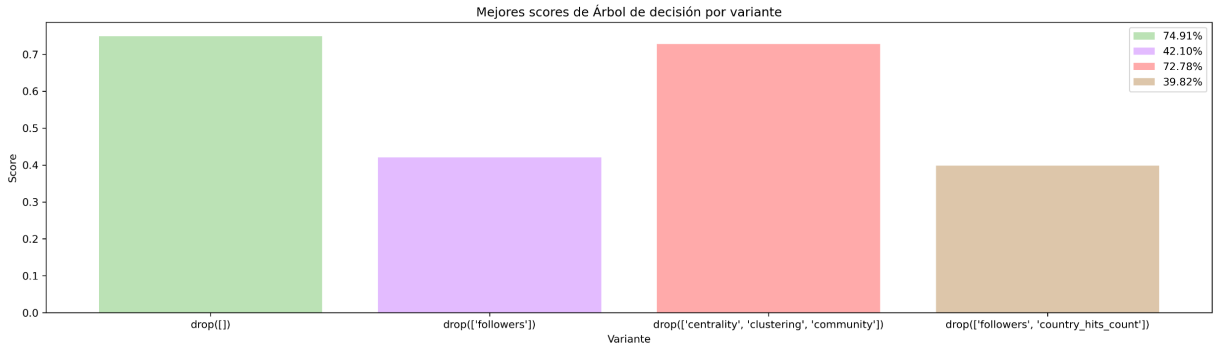
- **Atributos no relacionales (drop([’centrality’,’clustering’,’community’])):** Profundidad máxima: 7



- **Atributos relacionales (drop([’followers’,’country-hits-count’])):** Profundidad máxima: 9



Por último, se muestran a continuación los resultados finales para cada una de las variantes con los mejores hiper-parámetros.



A diferencia de k-NN, árboles de decisión si consigue un mejor resultado al añadir los atributos relacionales. Es decir, es más preciso con todos los atributos (verde) y menos preciso al eliminarlos (rojo). Esto se debe a que consigue una generalización a raíz de los atributos relacionales, y a través de preguntas a dichos atributos consigue aumentar la precisión.

Como se puede observar en la barra en la que se elimina el atributo followers (morada), dicho atributo sigue siendo esencial, ya que eliminarlo supone un empeoramiento del 32,81 % respecto a nuestro mejor resultado (verde). La misma línea se sigue con los country charts hits, que hace que el modelo mejore parcialmente pero no marca la diferencia.

Como conclusión se puede extraer que los árboles de decisión han conseguido aprovechar los atributos relacionales y aumentar el rendimiento y precisión del modelo. Además, se mantiene la importancia del atributo followers.

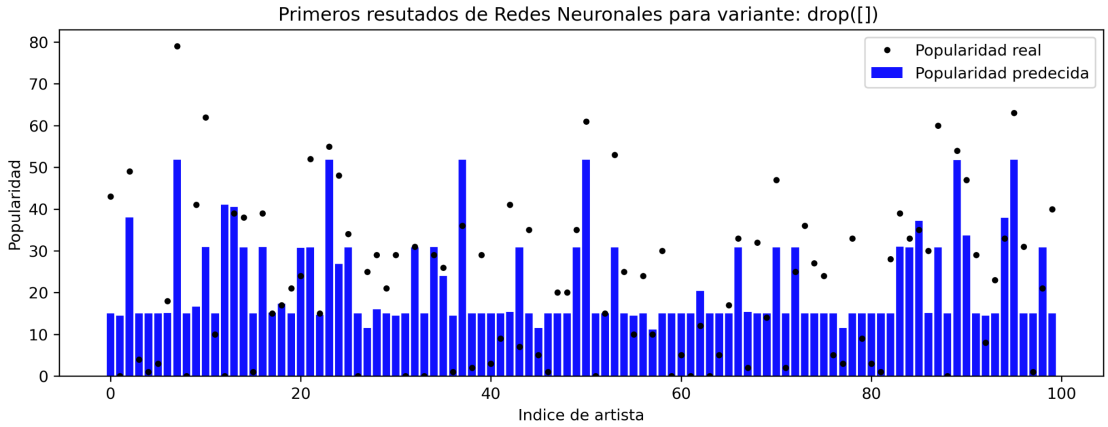
6.3. Redes neuronales

A continuación se muestran las 100 primeras predicciones realizadas para cada una de las variantes del problema y los hiper-parámetros de los resultados finales. En la gráfica se aprecia por cada barra una predicción realizada. Los puntos representan el valor esperado y la barra la predicción realizada.

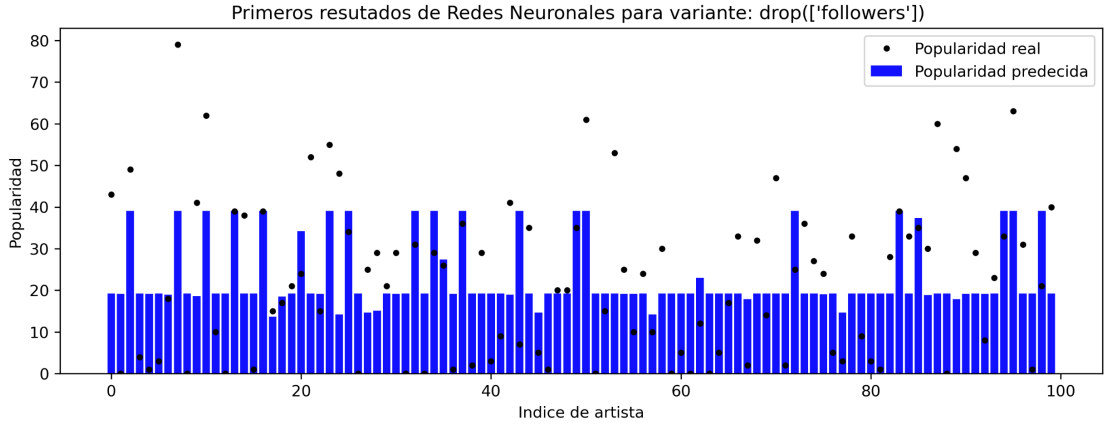
Los hiper-parámetros para las 4 variantes son comunes, por lo que se presentan a continuación:

Números de capas ocultas: 7
Número de neuronas por capa oculta: 2, 50, 2, 30, 7, 3, 2
Función de activación de las capas ocultas: tanh, identidad, tanh, identidad, identidad, identidad, identidad
Función de activación de la capa de entrada y de salida: identidad, identidad
Factor de aprendizaje: 0,00008
Número de lotes: 100
Número de épocas de entrenamiento: 20
Optimizador: SGD
Pérdidas: mean-square-error

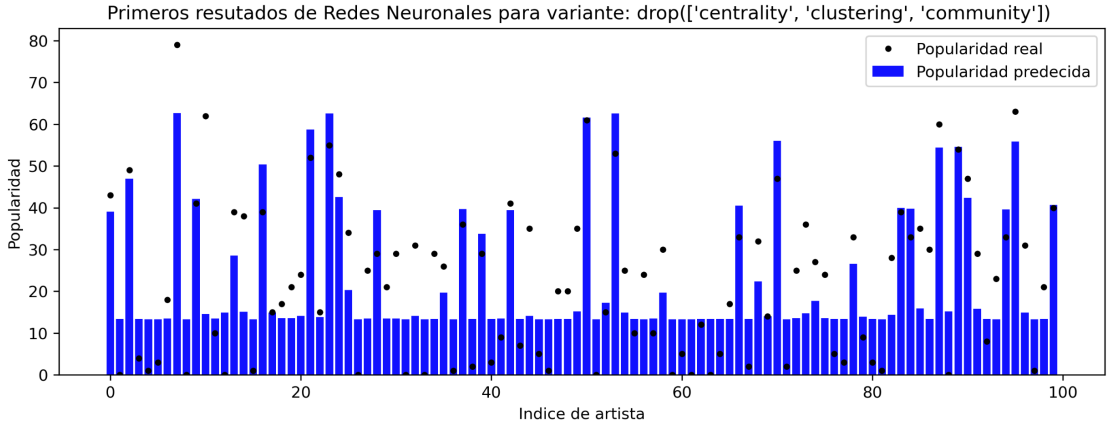
- **Todos los atributos (drop([])):**



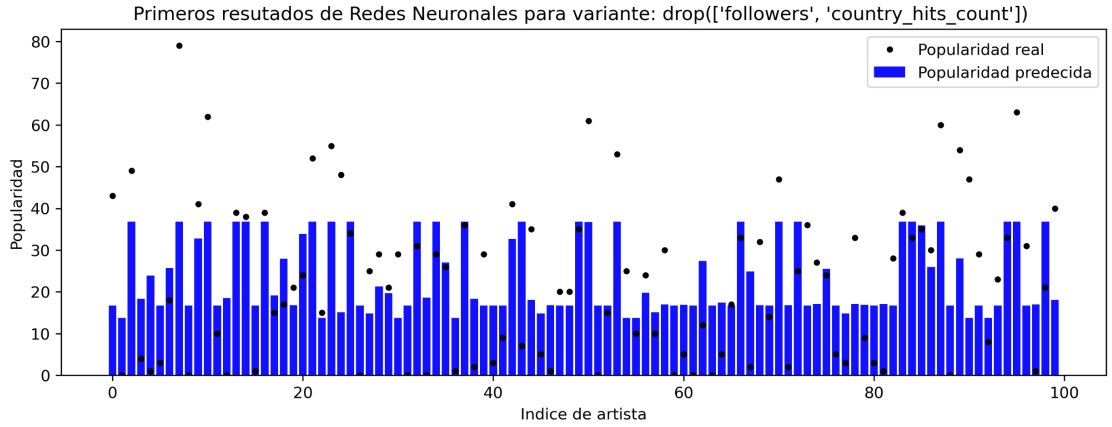
- **Todos los atributos menos followers (drop(['followers'])):**



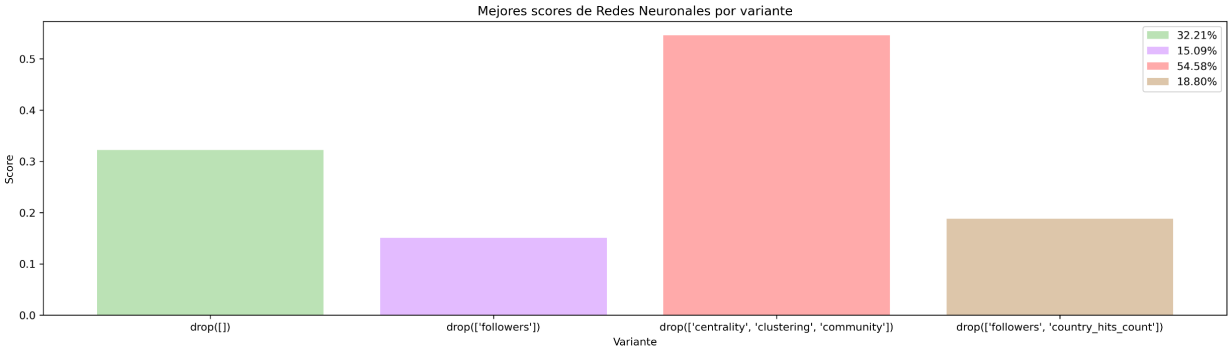
- **Atributos no relacionales (drop([’centrality’,’clustering’,’community’])):**



- **Atributos relacionales (drop([’followers’,’country-hits-count’])):**



Por último, se muestran a continuación los resultados finales para cada una de las variantes con los mejores hiper-parámetros.



Cabe destacar en primer lugar, el bajo rendimiento de este modelo respecto a k-NN y árboles de decisión. Esto se debe a que las redes neuronales son ciertamente más complejas que dichos modelos y están hechas para abordar problemas más genéricos. Esto implica que el modelo debe poder adaptarse a situaciones más variables, lo que conlleva a un mayor número de hiper-parámetros y más complejidad a la hora de decidir los valores que tomarán.

Aunque no sea el modelo más óptimo, se puede deducir y extraer información útil. En primer lugar, se puede apreciar nuevamente la importancia del atributo followers (comparación entre verde y morada).

Además, al eliminar los atributo relacionales (rojo), la red neuronal mejora significativamente la capacidad de acierto. Sin embargo, al eliminar los atributos no relacionales (marrón), el modelo es mucho menos preciso.

Como conclusión se puede extraer que el atributo followers sigue siendo el atributo más importante. Por otro lado, se puede observar que la red neuronal, con los hiper-parámetros actuales, es incapaz de ajustar sus pesos para tener en cuenta los atributos relacionales (marrón). Siendo el modelo más preciso si únicamente se dejan los atributos no relacionales (rojo).

7. Conclusiones

El propósito de este estudio es comprobar si los atributos relacionales pueden afectar a la predicción de los resultados de modelos de aprendizaje automático como sugiere la literatura científica.

Antes de entrar en profundidad en estos atributos, es necesario remarcar que según el estudio realizado para los 3 modelos, el atributo followers es **determinante** para la predicción del atributo objetivo (IP). La demostración queda clara ya que, al quitar únicamente este atributo, la precisión baja considerablemente.

En el estudio a priori se indicaba la posibilidad de que el atributo derivado country hits count tuviera gran relevancia. Lo cierto es que mejora pacialmente, aunque no de manera significativa los resultados.

Con respecto al impacto de los atributos relacionales sobre los resultados obtenidos, se concluye que la repercusión dependerá del modelo en el que se empleen. Mientras que en árboles de decisión se ha conseguido una mejor precisión, en k-NN y redes neuronales hay un empeoramiento sustancial de los resultados.

La mejora en árboles de decisión se debe a que el modelo es capaz de generalizar los atributos relacionales y realizar preguntas óptimas a raíz de estos, encontrando resultados más precisos. Por otro lado, el impacto del atributo followers sobre k-NN es muy significativo. Esto es apreciable, ya que, al añadir los atributos relacionales, es decir, mayor número de dimensiones en el espacio, los puntos se dispersan con más facilidad. Esto hace que las distancias entre los puntos varíen considerablemente hasta el punto de empeorar la precisión del modelo. Respecto a la red neuronal, el modelo es incapaz de ajustar los pesos y generalizar dichos atributos, por lo que no se adapta a los atributos relacionales.

Para nuestro dominio del problema y el contexto de nuestros atributos, el modelo que mejor se ha comportado ha sido árboles de decisión incluyendo atributos relacionales y no relacionales con una precisión en los resultados del 74,91 %.

Una vez analizada la repercusión de los resultados, es prudente afirmar que de forma general, los atributos relacionales realmente ayudan a los modelos de aprendizaje de datos modelables por un grafo, siempre que se tenga en cuenta que la existencia de atributos con una fuerte correlación con el objetivo pueden causar que los atributos relacionales empeoren las predicciones.

Se concluye por tanto, en que la aplicación de información relacional para la mejora de modelos predictivos, dependerá del dominio del problema, de los atributos y sus correlaciones, además del modelo que se quiera emplear para la tarea de predicción. En este caso, se obtiene una mejora en la precisión gracias a los atributos relacionales para el modelo de árboles de decisión, siendo dicho modelo el más óptimo para resolver el problema planteado.

8. Bibliografía

Referencias

[1] CD BABY. *El algoritmo de Spotify: lo que los músicos deben saber*. 2022. URL: https://musicodiy.cdbaby.com/el-algoritmo-de-spotify-lo-que-los-musicos-deben-saber/ (visitado).

[2] Pedro Almagro Blanco. «Descubrimiento de Conocimiento en Grafos Multi-Relacionales». En: (abr. de 2017), pág. 31. URL: https://idus.us.es/bitstream/handle/11441/71286/Descubrimiento%20de%20Conocimiento%20en%20Grafos%20Multirelacionales.pdf?sequence=1&isAllowed=y.

[3] Yu Cheng y Daniel Suthers. «Social Network Analysis — Centrality Measures». En: (mayo de 2011), pág. 10. URL: http://www.jade-cheng.com/uh/coursework/ics-668/project.pdf.

[4] R. Scott Hiller y Jason M. Walter. «The Rise of Streaming Music and Implications for Music Production». En: *Review of Network Economics* (dic. de 2017). DOI: 10.1515/rne-2017-0064.

[5] R. Scott Hiller y Jason M. Walter. «The Rise of Streaming Music and Implications for Music Production». En: *Review of Network Economics* (dic. de 2017). Discussion, pág. 13. DOI: 10.1515/rne-2017-0064.

[6] Spotify. *Web API Reference*. 2023. URL: https://developer.spotify.com/documentation/web-api/reference/get-an-artist (visitado).