Estudio de Quakers Automático

Andrés Domínguez Ruiz
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Correo: andresdominguezruiz.ad@gmail.com
Correo universidad: anddomrui@alum.us.es
UVUS: anddomrui

Pablo Núñez Moreno
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
Correo: panumo123@gmail.com
Correo universidad: pannuñmor@alum.us.es
UVUS: panñumor

El alumno Pablo Núñez Moreno no ha participado en ningún momento en el trabajo.

***Resumen*—** El objetivo principal de este trabajo es implementar modelos de Machine Learning de los nodos del grafo basado en los miembros de Quakers (personajes relacionados con el protestantismo en el siglo XVII) a partir de tanto los atributos que ya presentan como de los atributos relacionales que consideremos. En nuestro caso, las métricas relacionales que hemos utilizado son el grado de centralidad, la detección de comunidades y el grado de agrupamiento de cada nodo.

Las conclusiones obtenidas son que las métricas relacionales solicitadas añaden complejidad a los modelos que hemos ido construyendo, que los modelos ML de regresión no presentarán un gran rendimiento debido a que tenemos muchos atributos continuos poco asociados entre sí y pocos ejemplos, que para tareas de clasificación sobre árboles de decisión vamos a obtener mejores resultados con una profundidad máxima de 1, y que no se puede aplicar train Split debido a que tenemos pocos ejemplos, y al tener pocos ejemplos la complejidad de los modelos aumenta y se dan sobreajustes.

Palabras Clave—Inteligencia Artificial, Aprendizaje Automático, Quakers, Regresión, Clasificación, Random Forest, Knn, Árbol de Decisiones, CART, Machine Learning, ML

I. INTRODUCCIÓN

Este trabajo está enfocado en el tratamiento de los datos relacionados con los personajes de Quakers, que son aquellas personas que participaron en el protestantismo cristiano durante el siglo **XVII**. Este grupo también es conocido como La Sociedad Religiosa de los Amigos[1]. Los datos fueron recogidos por Melanie Walsh y fueron almacenados en su repositorio de GitHub[2]. Melanie Walsh almacenó para cada miembro de La Sociedad Religiosa de los Amigos diferentes atributos, ya sean tanto atributos categóricos como el género como atributos continuos como la fecha de nacimiento.

El objetivo de este proyecto es tratar estos datos para poder generar diferentes modelos de Machine Learning y analizar los resultados obtenidos. Para añadir complejidad a los datos y a los modelos, utilizamos métricas relacionales que aplicamos posteriormente al grafo que Melanie Walsh publicó en Github. Estos atributos relacionales son el coeficiente de agrupamiento, la detección de comunidades y la centralidad. Tras añadir estas métricas a los datos, lo primero que hice fue realizar un análisis sobre los mismos, viendo las correlaciones entre los atributos, las frecuencias, las probabilidades y las conclusiones de los mismos análisis realizados.

Luego de este análisis, es cuando empezamos con la construcción de los diferentes modelos ML y realizando varios experimentos sobre los mismos.

Durante todo el proyecto, nos hemos enfocado primero en el tratamiento y análisis correcto de los datos que nos proporcionaban, ya que ellos son la clave para obtener modelos de ML con rendimientos óptimos.

En este documento vamos a explicar los siguientes apartados:

- La descripción y análisis del conjunto de datos
- La descripción de las métricas relacionales utilizadas
- El análisis residual que se ha realizado sobre cada atributo, junto a los resultados obtenidos.
- Descripción y experimentación de los algoritmos de aprendizaje automático aplicados, que son:
    1. Modelo Knn de regresión sobre atributo ‘birthdate’
    2. Modelo CART de clasificación sobre atributo ‘gender’
    3. Modelo RandomForest de clasificación sobre atributo ‘gender’
- Conclusiones

II. DESCRIPCIÓN Y ANÁLISIS DEL CONJUNTO DE DATOS

Cada ejemplo originalmente estaba formado por los siguientes atributos:

- **Id**, un atributo categórico no ordinal y multiclase que hacía referencia al nombre completo de cada miembro de Quakers. Este atributo fue utilizado por Melanie Walsh como identificador de cada nodo en el grafo. La cantidad de clases distintas que tiene son 96

- **Label,** un atributo categórico no ordinal con valores equivalentes a Id y que puede que haga referencia al nombre de la etiqueta del nodo. La cantidad de clases distintas son 96.
- **Historical significance,** un atributo categórico no ordinal y multiclase que representa el rol que tuvo cada miembro de Quakers. La cantidad de clases distintas son 54(originalmente)
- **Gender,** un atributo categórico no ordinal y binario que representa el género de cada miembro de Quakers. Al ser binario, presenta 2 clases, las cuales son 'male' y 'female'.
- **Birthdate,** un atributo numérico continuo que representa la fecha de nacimiento de cada miembro
- **Deathdate,** un atributo numérico continuo que representa la fecha de muerte de cada miembro
- **Other id,** un atributo numérico continuo que hace referencia a la identificación de un aspecto del miembro que no es mencionado por la autora.

Al observar el conjunto de ejemplos, nos fijamos en algo muy importante, el tamaño del conjunto es muy pequeño. Sólo tenemos 96 ejemplos, y sabiendo que la gran mayoría de atributos son atributos numéricos continuos o atributos categóricos multiclase, es probable que tengamos pocos ejemplos.

Durante la lectura de los datos, encontramos los siguientes problemas:

- 1º El campo de historical_significance de una de las filas estaba vacía, haciendo que esta devolviese NaN al intentar leerla. La solución a este problema fue añadir una nueva clase , 'Nothing', lo cuál hizo que a partir de ahora trabajásemos con 55 clases en este atributo
- 2º La cabecera de los datos era incluida como una fila más, ocasionando confusión y modificando la indexación de las filas. La solución a este problema fue saltarse la cabecera a partir de uno de los parámetros que proporciona la función de Pandas [3].

III. DESCRIPCIÓN DE LAS MÉTRICAS RELACIONALES UTILIZADAS

Una vez que tuvimos los datos cargados, decidimos implementar las siguientes métricas relacionales a cada nodo del grafo: **la detección de la comunidad a la que pertenece,el coeficiente de agrupamiento y el grado de centralidad.**

Para determinar la comunidad a la que pertenecía cada nodo, implementamos el Algoritmo de Detección de Comunidades de Louvain[4] que proporcionaba la librería de NetworkX, el cuál nos devuelve un reparto de comunidades que optimizan la modularidad del grafo. Uno de los problemas que tuvimos fue que este modelo no era exacto, y cada vez que se ejecutase podía devolver un reparto diferente, afectando al modelo K vecinos. Pero esto se solucionó fijando los datos una vez ejecutados y añadidas las respectivas métricas, de esta forma a la hora de construir los modelos tendríamos los mismos datos.

Este algoritmo casi siempre devolvía 8 comunidades, en las que estaban formadas por los identificadores de cada nodo del grafo, por lo que para asignarle a cada fila del DataFrame su respectiva comunidad se realizó lo siguiente:

1º Crear un diccionario que almacenase como clave el identificador del nodo y como valor una etiqueta que hacia referencia a la comunidad a la que pertenecía

2º Leer todo el DataFrame y para cada fila asignarle su respectiva etiqueta. Para este proceso de asignación, se realizó una función para facilitar el trabajo:

```
añadir_nueva_columna:

**Entrada:**
-Un dataframe D
-Un diccionario que utilice como clave el Identificador del dataframe
- El nombre de la nueva columna a añadir

**Salida:**
Void, pero le añade la nueva columna a D

**Procedimiento:**

1º Inicializar nueva columna en D
2º Para cada fila i de D:
        -Clave= identificador de la fila i
        - Si Clave aparece en las claves del diccionario
                Añadir al campo de la nueva columna de la fila i
                el valor del diccionario.
```

Fig 1. Pseudocódigo de la función auxiliar construida, añadir_nueva_columna

Como realmente lo que devolvemos son las etiquetas de las comunidades, podemos interpretar que este atributo es un atributo categórico **ordinal**, cuyos valores seguían la siguiente estructura:

comunidad-{i}, siendo i=0,…n-1 comunidades que tengamos

Para determinar el coeficiente de agrupamiento, NetworkX proporciona una función que devuelve un diccionario teniendo como claves el identificador del grafo y como valor el coeficiente de agrupamiento normalizado. El coeficiente de agrupamiento de un nodo indica como de agrupado esta un nodo con sus vecinos[5], por lo que esta métrica va a estar muy relacionada con la siguiente, el grado de centralidad. Una vez calculado el coeficiente de agrupamiento, sólo tenemos que volver a aplicar la función de **añadir_nueva_columna** sobre los datos y ya habremos añadido la métrica.

Para calcular el grado de centralidad de cada nodo al principio se recurrió a una función que proporciona NetworkX, pero esta devolvía los valores del diccionario aplicando una normalización que no entendíamos.

Por lo tanto, para evitar confusión , se calculó manualmente el grado de centralidad de cada nodo, ya que es la cantidad de aristas que tiene cada nodo. Para obtener este valor , recurrimos al fichero de datos de aristas que proporcionaba Melanie Walsh en su repositorio, el cuál leímos y recorrimos para ir cogiendo la cantidad de aristas que tenía cada nodo.

Una vez calculado la centralidad de cada nodo, sólo teníamos que añadírselo a los datos aplicando la función de **añadir_nueva_columna** denuevo.

IV. ANÁLISIS RESIDUAL

Tras preparar todos los atributos y métricas en el conjunto de datos, decidimos realizar un análisis profundo sobre cada uno de los atributos. Sobre cada atributo hemos analizado los siguientes aspectos: la correlación con el resto de atributos[6], la frecuencia y probabilidad de cada valor aparecido en el conjunto de datos, las máximas y mínimas frecuencias , y la conclusión del análisis realizado sobre el atributo. Hay casos en los que las frecuencias y probabilidades son todas iguales, por Una vez que hicimos el análisis sobre todos los atributos, recopilamos una serie de conclusiones del análisis global.

A. Análisis sobre Id y Label:

Correlación:

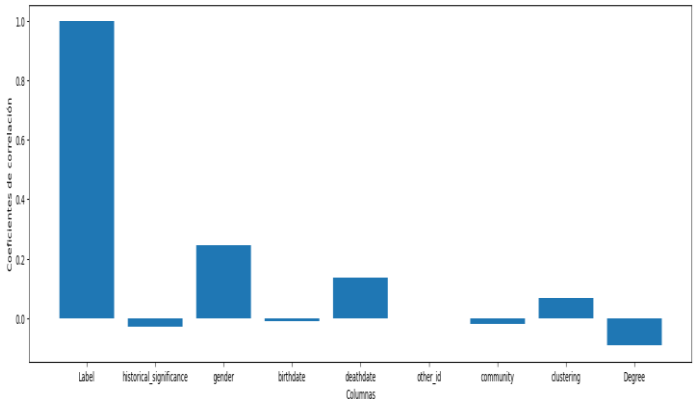

Figura 2. Grafico que muestra la correlación entre Id y el resto de atributos.

Podemos observar que other_id presenta una asociación nula, y esto es debido a que other_id no varía en ningún momento respecto a Id, lo que implicaría que si utilizamos esta columna durante la predecir Id no obtendríamos un modelo óptimo, obtendríamos seguramente un modelo sobreajustado. Pero por otro lado, podemos observar que Label presenta una asociación muy alta, implicando que tanto Id como Label pueden ser iguales, aunque la correlación entre los demás atributos es muy baja o negativa .

Como la asociación entre Id y Label es 1.0, entonces comprobamos si Id y Label son iguales con todos los ejemplos, y efectivamente, lo son, por lo tanto tenemos una columna que nos esta proporcionando contenido repetido, por lo que se optó por no realizar un análisis sobre Label.

Frecuencia y probabilidad:

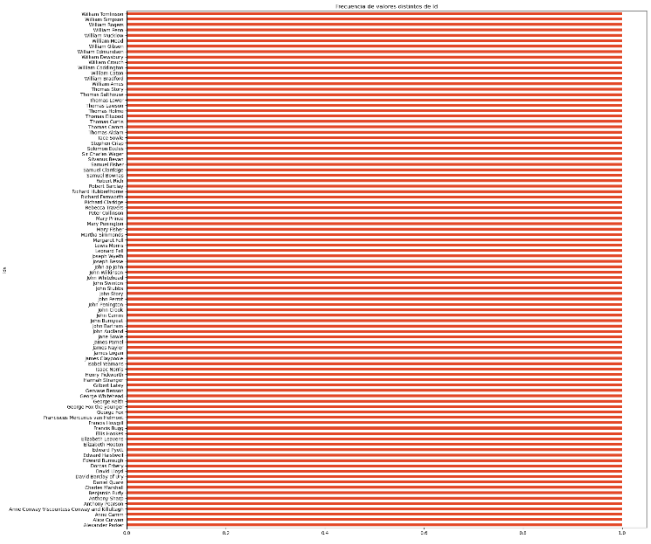

Figura 3. Gráfico de frecuencia de Id

Tras estudiar la frecuencia de cada clase del atributo, nos percatamos que tenemos 96 clases diferentes, es decir, 96 nombres distintos en todo el conjunto de datos. Esto implica que todos los miembros de Quakers tenían un nombre distinto.

Como todos los nombres tienen la misma frecuencia , las probabilidades que se obtengan serán las mismas también

Máximas frecuencias y mínimas frecuencias:

Como las frecuencias son las mismas, no es posible determinar los nombres más frecuentes y los nombres menos frecuentes.

Conclusiones del análisis sobre Id y Label:

Como no presenta mucha correlación con el resto de atributos y la frecuencia y probabilidad del mismo son mínimas, tanto Id como Label nos complicará la costrucción de modelos ML. Y como Label es idéntico a Id, las conclusiones que obtengamos serían las mismas , por lo que no sería necesario su análisis.

También podemos decir que el nombre tiene una asociación positiva mediana con gender ya que los nombres pueden depender del género y viceversa.

B. Análisis sobre historical_significance:

Correlación:

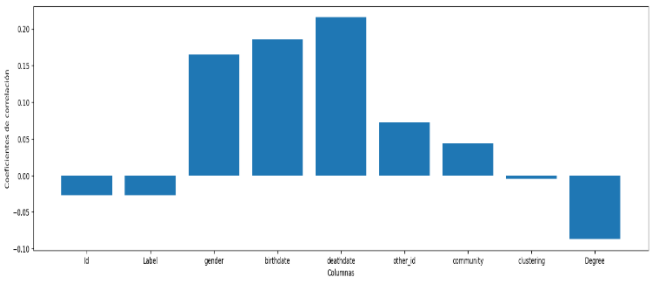

Figura 4. Gráfico que muestra la correlación entre historical_significance y el resto de atributos.

Analizando el gráfico, podemos determinar que los atributos más asociados positivamente son gender, birthdate y deathdate, implicando que estos atributos son necesarios para la construcción de un modelo ML que intente estimar historical_significance. Con el resto de atributos, o presenta una asociación positiva leve o es negativa la asociación, implicando que estos atributos son los que van a añadir complejidad al modelo.

Frecuencias y probabilidades:

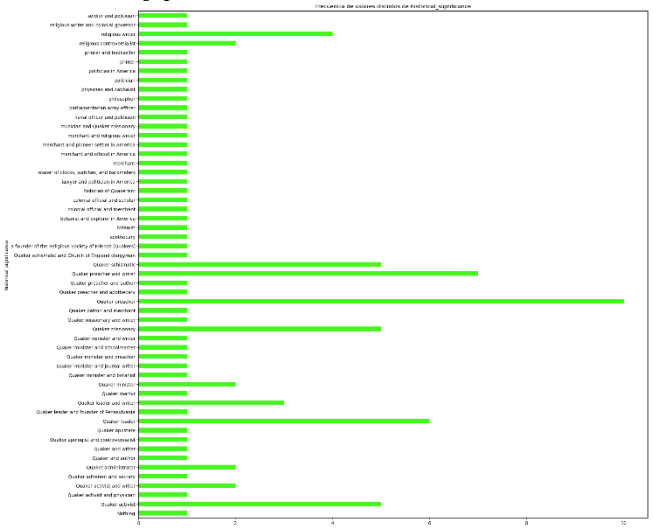

Figura 5. Gráfico de frecuencia de historical_significance

A partir de este grafo, podemos observar 55 clases diferentes, pero la gran mayoría comparten la mínima frecuencia posible, implicando que casi todos los miembros de Quakers tenían roles individuales. Las probabilidades nos dirían información similar.

Máximas y mínimas frecuencias:

El rol más frecuente en Quakers son los predicadores, lo que implica que los miembros de Quakers o eran predicadores o eran un rol diferente al resto.

Conclusión del análisis sobre historical_significance:

Tras haber tan poca correlación con el resto de atributos y tener tan pocos ejemplos con tantas clases diferentes, si intentamos construir un modelo ML que prediga historical_significance, obtendremos un rendimiento muy malo, por lo que no deberíamos utilizar este atributo. Como este atributo es uno de los solicitados por los profesores, no lo vamos a quitar para estudiar otros modelos sobre otros atributos, pero deberíamos.

C. Análisis sobre gender:

Correlación:

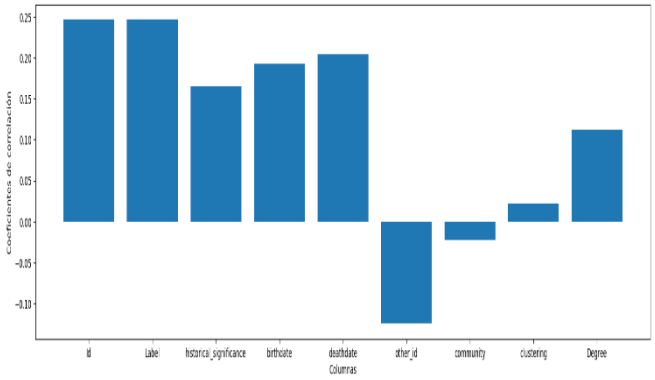

Figura 6. Gráfico de la correlación entre gender y el resto de atributos.

A partir de esta tabla, podemos determinar que other_id y community estan muy correlacionados negativamente con gender, por lo que estos atributos nos darían problemas a la hora de hacer modelos ML que intenten predecir gender. Luego tenemos que el resto de atributos estan correlacionados positivamente con gender, sobretodo Id y Label, y esto último se debe a que el nombre completo de los miembros de Quakers pueden depender del género que tienen, pero como hay tantas clases en Id y Label, si queremos construir un modelo que estime gender , puede complicarse la estimación.

Frecuencias y probabilidades:

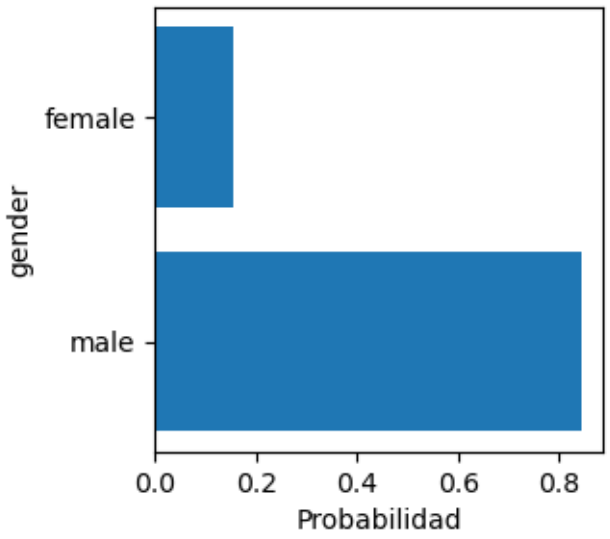

Figura 7. Gráfico de las probabilidades de gender

Analizando tanto la frecuencia como la probabilidad, sabemos que la gran mayoría de miembros de Quakers eran hombres y el resto eran mujeres.

Máximas y mínimas frecuencias:

El género más frecuente es ‘male’ y el género menos frecuente es ‘female’, y sus valores son 81 y 15 respectivamente.

Conclusión del análisis sobre gender:

Al tratarse de un atributo categórico binario y al estar correlacionado positia y ligeramente con la gran mayoria de

atributos, es un atributo el cuál podemos intentar predecir en alguno de nuestros modelos.

D. Análisis sobre birthdate:

Correlación:

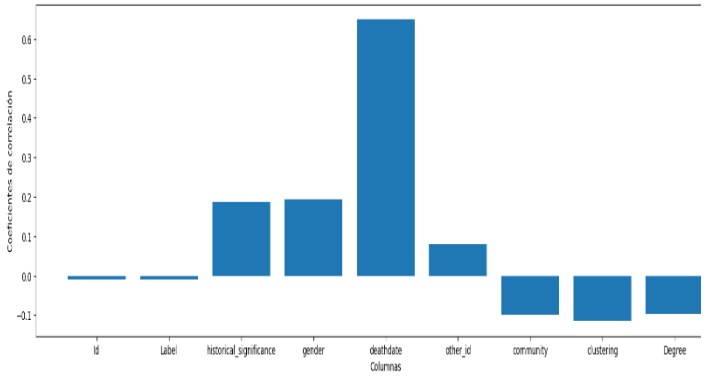

Figura 8. Gráfico de la correlación entre birthdate y el resto de atributos.

Como podemos observar, birthdate presenta asociaciones positivas con todos los atributos originales de los datos, mientras que con las métricas relacionales no. Esto último significa que si decidimos construir un modelo ML que estime la fecha de nacimiento, las métricas relacionales van a añadirle complejidad al modelo.

Por otro lado, la fecha de muerte esta fuertemente asociado con la fecha de nacimiento, implicando que las personas que nacieron en un año determinado puede que murieran en el mismo año o similar.

Frecuencias y probabilidades:

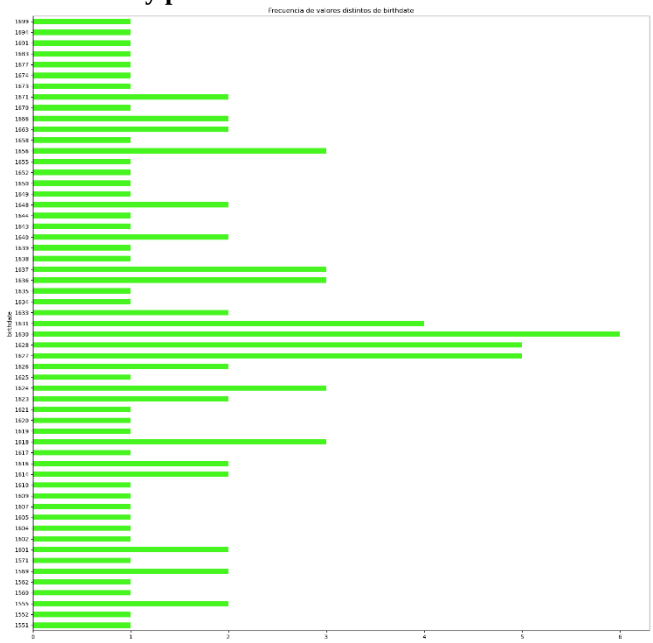

Figura 9. Gráfico de frecuencias de birthdate.

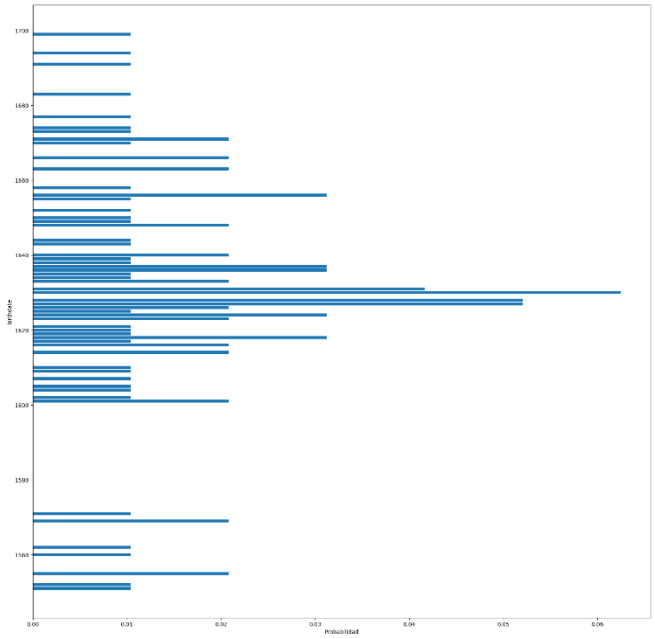

Figura 10. Gráfico de las probabilidades de birthdate.

Podemos decir que al tratarse birthdate de un atributo numérico continuo y de tener un conjunto de datos pequeño, tenemos como resultado una gran cantidad de valores y una pequeña frecuencia de cada uno. Esto implica que casi todos los miembros de Quakers nacieron en años diferentes, y al estar altamente correlacionado con deathdate, esto implicaría que murieron los miembros en momentos similares.

Pero si observamos el gráfico de probabilidades, podemos diferenciar varios momentos en los que nacieron la gran mayoría de miembros de Quakers, como por ejemplos, a partir de los 1600 nacieron la gran mayoría de miembros, pero hubo otros miembros que nacieron antes y después.

Máximas y mínimas frecuencias:

El año 1630 es el año en donde más miembros de Quakers nacieron, y como el resto de miembro nacieron en años distintos, casi todos los años sólo aparecen una vez, por lo que no se puede estimar el año con menos miembros.

Conclusión del análisis de birthdate:

- Como birthdate esta altamente correlacionado con deathdate, esto significa que si queremos hacer modelos de regresión sobre este atributo , el atributo principal que ayudará a la estimación será deathdate.
- Casi todos los miembros nacieron en años diferentes y durante intervalos de tiempo diferente (algunos sobre 1550, la gran mayoria sobre 1600 y otros muchos sobre 1650)

E. Análisis sobre deathdate:

Correlación:

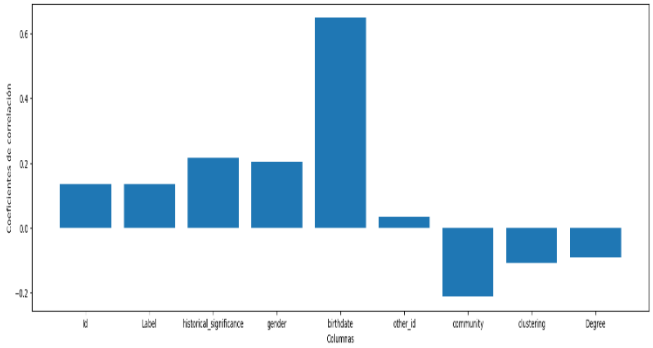

Figura 10. Gráfico de la correlación entre deathdate y el resto de atributos.

Como podemos observar, se obtienen resultados similares que el análisis sobre birthdate, siendo este último atributo mencionado el que mejor asociado esta con deathdate. Esto refuta el hecho de que muchos miembros que murieron en un mismo año o en años similares nacieron en años equivalentes o aproximados. Respecto al resto de atributos, con gender, historical_significance y other_id mantiene una correlación positiva pero baja, mientras que con el resto de atributos es una correlación negativa.

Frecuencias y probabilidades:

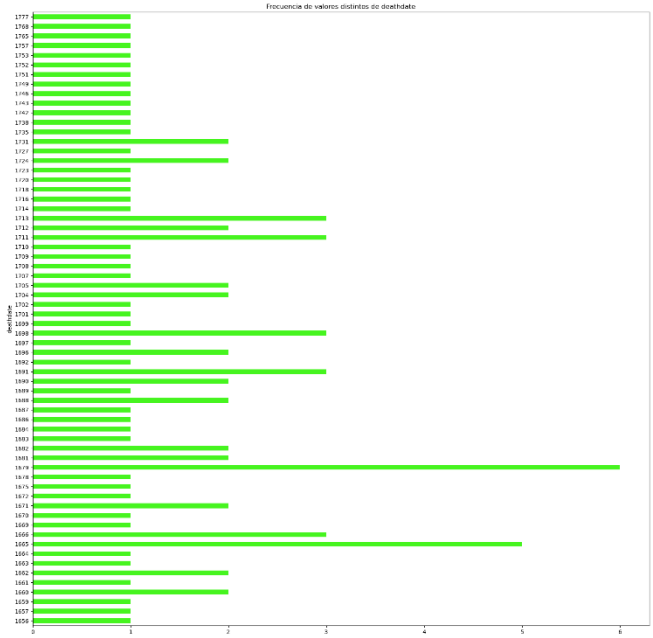

Figura 12. Gráfico de frecuencias de deathdate

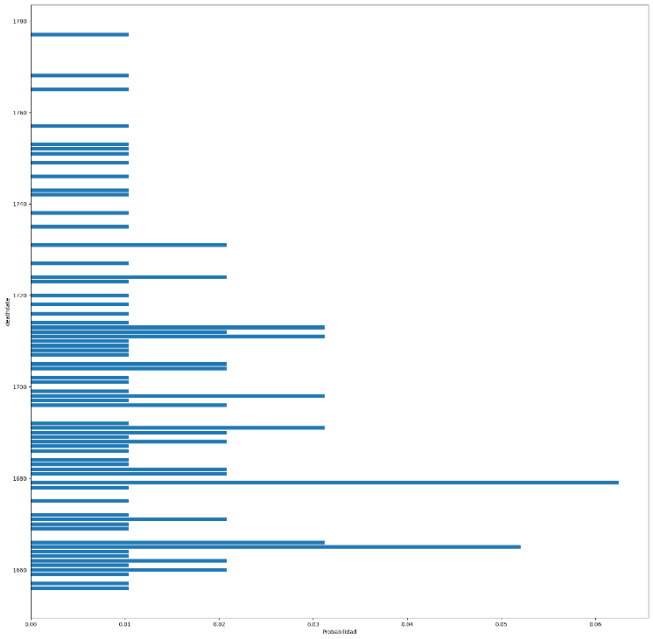

Figura 13. Gráfico de probabilidades de deathdate.

Analizando las frecuencias obtenidas, podemos determinar que ocurre lo mismo que con birthdate, tenemos muchos valores distintos pero demasiados pocos ejemplos, dando lugar a que la gran mayoría de deathdate tengan valores distintos.

Analizando las probabilidades obtenidas, podemos decir que la gran mayoria de miembros de Quakers murieron entre los años 1730 y 1660

Máximas y mínimas frecuencias: Sólo podemos determinar el año de muerte en el que murieron más miembros, el cuál fue 1679, en el que murieron 6 miembros.

Conclusión del análisis de deathdate:

- Al estar fuertemente correlacionados positivamente deathdate y birthdate, esto implica que ambos atributos son necesarios para estimarse mutuamente y que los miembros de Quakers murieron con edades similares.
- Al tener muchos valores distintos y con tan poca frecuencia, realizar un modelo ML de regresión sobre este atributo darían como resultado modelos con mal rendimiento.

F. Análisis sobre other_id:

Correlación:

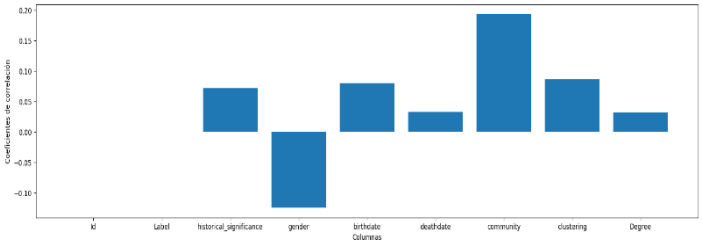

Figura 14. Gráfico de correlaciones entre other_id y el resto de atributos.

Podemos observar que other_id presenta asociaciones positivas pequeñas o negativas con los atributos originales de los datos, pero esta asociado positivamente sobretodo con las métricas relacionales. Esto implicaría que este atributo puede tener sentido en la estructura del grafo y no sobre las propiedades de los nodos.

Frecuencias y probabilidades:

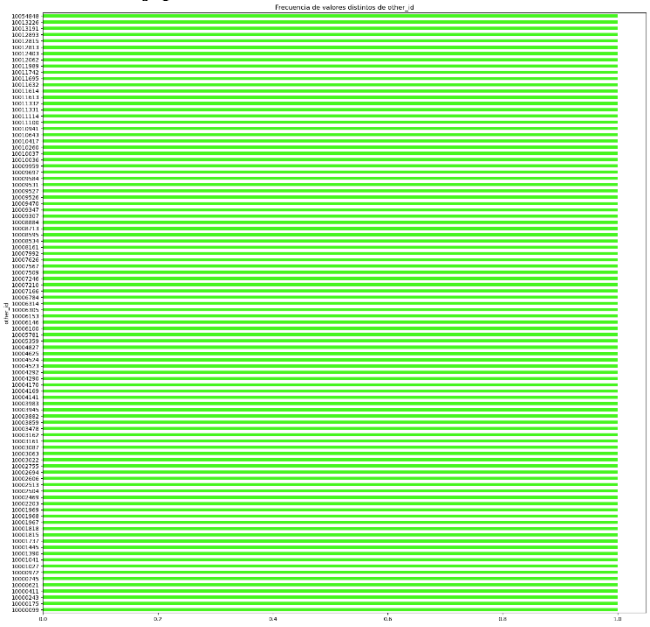

Figura 15. Gráfico de las frecuencias de other_id.

Obtenemos 96 valores distintos, por lo que ocurre lo mismo que con Id y Label, cada miembro de Quakers tiene un other_id diferente, por lo que las frecuencias y probabilidades son todas iguales, haciendo imposible obtener tanto el other_id más frecuente como el que menos.

Conclusión del análisis de other_id:

- Este atributo al estar poco correlacionado con el resto de atributos no nos sirve ni para modelos ML que estimen otros atributos ni para modelos ML que estimen other_id. Y como no es uno de los atributos que ha mencionado el profesor, podemos eliminarlo
- Como tenemos tantos valores distintos, no podemos obtener información útil que podamos utilizar para la construcción de modelos ML.
- Se ha estimado que other_id depende de los atributos relacionales del grafo y no de los propios atributos de los nodos.

G. Análisis sobre community:

Correlación:

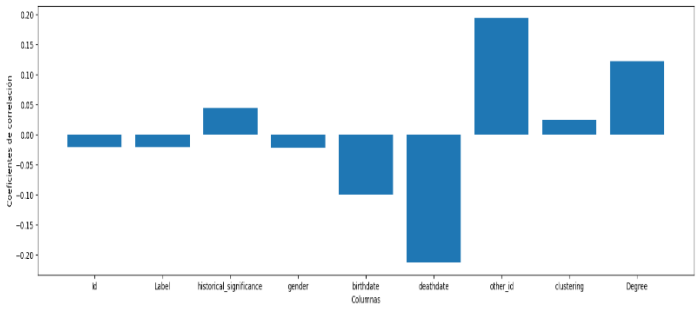

Figura 16. Gráfico de las correlaciones entre community y el resto de atributos

Podemos observar que community esta poco asociado con todos los atributos generales menos con other_id ya que este último es un atributo que hace referencia a algún identificador extra del grafo. Las asociaciones negativas implican que este atributo añadirá complejidad a la estimación de deathdate, birthdate, gender, Label e Id si intentamos crear modelos ML que estimen estos atributos.

Frecuencias y probabilidades:

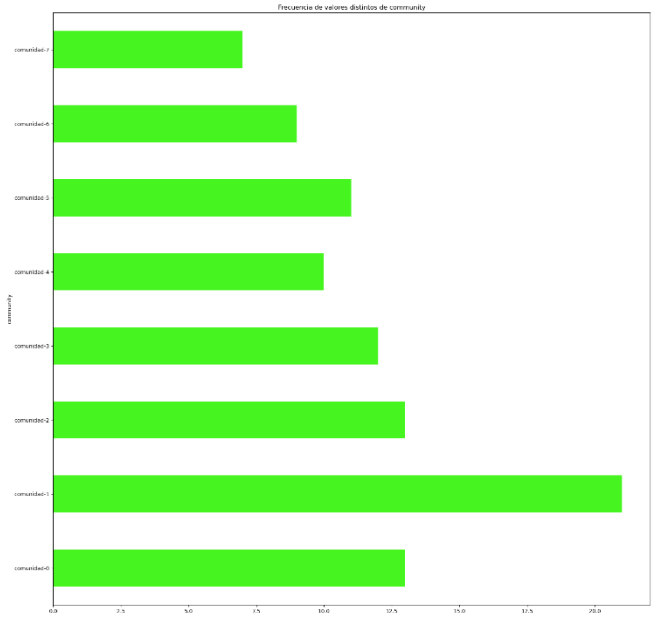

Figura 17. Gráfico de frecuencias de community

Podemos apreciar que como tenemos pocos ejemplos pero también pocas comunidades, podemos distinguir entre frecuencias máximas y mínimas, y con las probabilidades ocurrirá lo mismo.

Máximas y mínimas frecuencias:

La comunidad más frecuente es la comunidad-1, teniendo 21 nodos en él, y la comunidad menos frecuente es la comunidad-7, con 7 nodos en él.

Conclusión del análisis sobre community:

- La métrica de detección de comunidades no es útil para estos datos, ya que no esta muy asociado al resto de atributos

- Al tener pocos datos y pocas comunidades, la frecuencia y probabilidad están bien repartidas y hay variedad, pero al no estar bien asociado con los demás atributos, podemos llegar a la conclusión de que este atributo otorgará complejidad a los ML que construyamos.

H. Análisis sobre clustering (grado de agrupamiento)

Correlación:

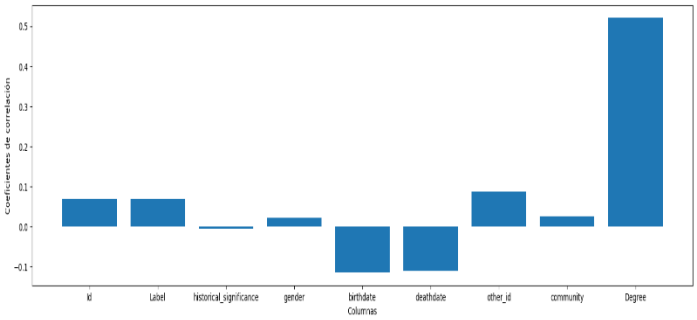

Figura 18. Gráfico de correlaciones entre clustering y el resto de atributos.

Podemos observar que el clustering presenta una asociación positiva media con Degree, implicando que algunos grados de clustering tendrán Degrees similares y viceversa. También presenta asociaciones positivas con other_id, community, Label, Id, historical_significance y gender, pero son débiles, y con el resto de atributos presenta asociación negativa. Estas asociaciones positivas sobre los atributos relacionales ocurre porque comparten propiedades relacionales entre si, es decir, dependen de datos que se obtienen de la propia lectura del grafo y no de los nodos del grafo.

Este último hecho corrobora que other_id realmente sea un atributo relacional .

Frecuencias y probabilidades:

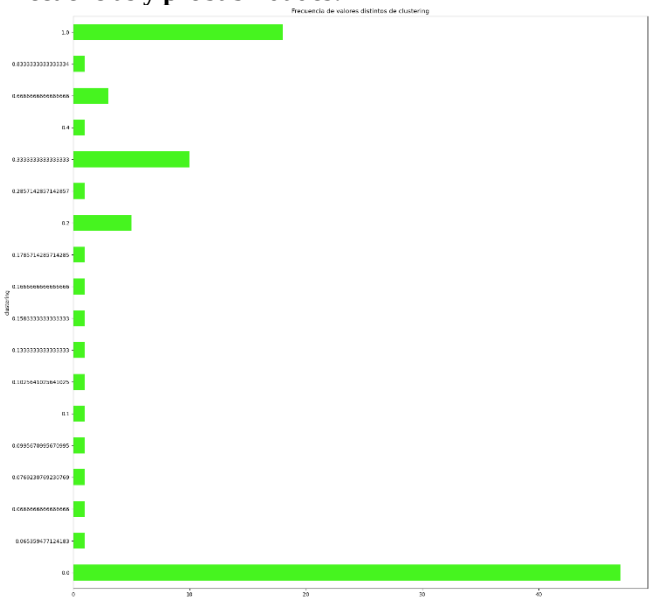

Figura 19. Gráfico de frecuencias de clustering

Aunque este atributo sea numérico continuo, gracias a este grafo hemos podido observar que tenemos 18 valores distintos, por lo que las frecuencias y probabilidades son variadas.

Tras analizar esto, podemos llegar a la conclusión de que la gran mayoría de nodos del grafo están mínimamente agrupados con sus vecinos, y al observar las demás frecuencias, podemos determinar que todo el grafo esta poco agrupado.

Máximas y mínimas frecuencias: El valor menos frecuente no se puede estimar ya que varias aparecen sólo una vez, pero la frecuencia máxima se da en el coeficiente =0., implicando que la gran mayoría del grafo esta poco agrupado.

Conclusión del análisis sobre clustering:

- Al estar relacionado el coeficiente de clustering con los vecinos de cada nodo del grafo, eso explica porque Degree y clustering están correlacionados, implicando que el coeficiente de clustering es necesario para estimar valores de Degree en modelos ML y viceversa
- No sirve de mucho esta métrica sobre los atributos no relacionales que tenemos, ya que la correlación de estas o es negativa o es positiva pero débil. Por lo que no es una buena métrica para modelos ML que estimen estos atributos, pero añade complejidad al modelo

I. Análisis sobre Degree:

Correlación:

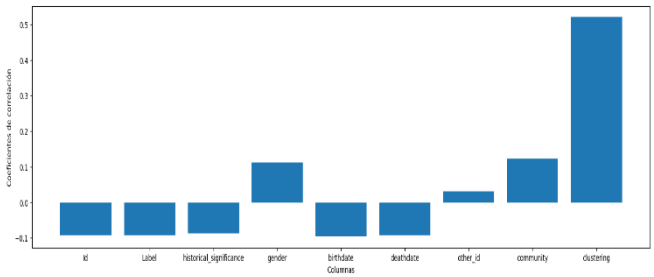

Figura 20. Gráficos de correlación entre Degree y el resto de atributos

Lo primero que podemos destacar es que presenta pocas asociaciones positivas, pero las que presentan son medianas, pequeñas o moderadas. Que tenga Degree una correlación tan elevada con clustering corrobora el análisis realizado sobre clustering, ambos dependen de la asociaciones de cada nodo.
Pero con respecto al resto de atributos, sólo presenta una asociación positiva con gender, other_id y community, con el resto de atributos presenta una asociación negativa, implicando que Degree dificultará la creación de modelos ML sobre estos atributos.

Frecuencias y probabilidades:

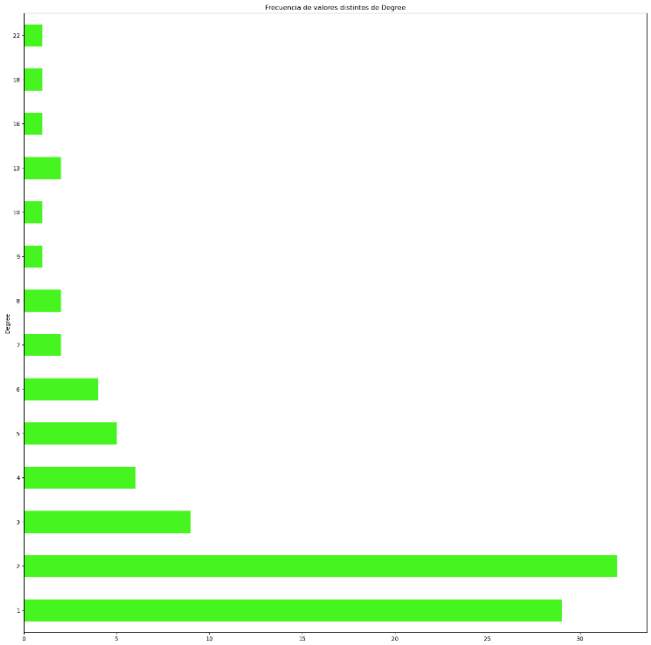

Figura 21. Gráfico de frecuencias de Degree.

A partir de este gráfico y el estudio de las probabilidades, podemos identificar que la gran mayoría de nodos presentaban entre 1 y 3 aristas (siendo el más frecuente 2 aristas), y esto explicaría porque los coeficientes de agrupamiento más frecuentes son aquellos con valores más próximos a 0.

Máximas y mínimas frecuencias: El grado de centralidad más frecuente es 2, y el mínimo no es determinable debido a que varios nodos presentan una cantidad distinta de aristas.

Conclusión del análisis sobre Degree:
- No es una métrica adecuada para predecir la gran mayoria de atributos de los datos, ya que sólo presenta una asociación positiva destacable con los atributos 'gender' y 'clustering'.
- Al tener pocos valores vistos y pocos ejemplos, las frecuencias y probabilidades que se observan son variadas.

J. CONCLUSIONES FINALES DEL ANÁLISIS RESIDUAL:

- Las métricas relacionales van a añadir complejidad a los modelos ML que construyamos, ya que estan poco correlacionados con los atributos propios de cada nodo, pero presentan frecuencias y probabilidades variadas.
- Id, Label y other_id al presentar valores distintos con la misma frecuencia van a añadir demasiada complejidad a los modelos, por lo que vamos a omitirlos para hacer más sencilla la construcción de los modelos ML que hagamos.
- historical_significance es un atributo que esta asociado positivamente con la gran mayoria de atributos, pero las frecuencias y probabilidades de esta no son muy variadas, por lo que añadirá complejidad a los modelos pero no tanto como otros atributos.
- Con gender ocurre lo mismo que con historical_significance, pero al tratarse de un atributo binario tenemos sólo 2 clases posibles, facilitando la estimación de la misma en los modelos ML
- Con birthdate y deathdate supimos que la gran mayoria de personas que murieron en un año o en un año similar nacieron en un año en común o en uno aproximado, implicando que muchos de los miembros de Quakers tenían edades similares.
- other_id se trata de un atributo relacional, es por eso que las correlaciones con esta sólo son positivas con las métricas relacionales.

Tras modificar el conjunto de datos y prepararlo, es cuando empezamos con la construcción de los modelos.

V. CONSTRUCCIÓN Y EXPLICACIÓN DE LOS MODELOS

A partir de las conclusiones obtenidas en el análisis residual, decidimos realizar 3 modelos de aprendizaje automático: un modelo Knn de regresión para estimar birthdate, un modelo CART de clasificación para clasificar gender y un modelo Random Forest para clasificar gender.

A. Modelo Knn de regresión para estimar birthdate:

Este algoritmo consiste en la búsqueda de los k vecinos de entrenamiento más cercanos a los ejemplos de los que quieres estimar su valor. Para saber los k vecinos más cercanos, es necesario fijar una métrica de distancia de antes, ya que con esta se calculará la distancia que tendremos luego en cuenta para la estimación. Luego de obtener los k vecinos más cercanos, como estamos en una tarea de regresión, haremos la media de sus atributos objetivos, y ese valor resultante será la estimación del valor[7].

El primer paso para construir nuestro modelo es codificar los atributos categóricos que tenemos(historical_significance, gender, community), y para mejorar la estimación, podemos normalizar los atributos continuos que tenemos(deathdate, clustering, Degree).

Tras codificar y normalizar los atributos, aparece nuestro primer problema ,**¿ deberíamos aplicar validación cruzada porque tenemos pocos ejemplos o podremos separar nuestro conjunto de ejemplos en conjunto de entrenamiento, conjunto de validación y conjunto de test?**
Para saberlo, decidimos estudiar los resultados de ambos métodos a la vez que estudiábamos su mejor k y su mejor métrica de distancia.

Para estudiar ambos casos el procedimiento ejecutado fue el mismo:
1º Vamos recorriendo todos los valores de k posible
2º Creamos modelos Knn de regresión aplicando todas las métricas de distancia

3º Entrenamos los modelos y los evaluamos (aplicando la validación cruzada sería comprobando el valor del score obtenido en la parte de test, y aplicando el train Split sería entrenando el modelo con los ejemplos de entrenamiento y luego evaluar el modelo a partir de los ejemplos de test)
4º Nos quedamos con aquel modelo que genere un mejor rendimiento.

Como también era necesario evaluar el ajuste de los modelos, decidimos estudiarlo a la vez que buscábamos el modelo óptimo, lo cúal lo hemos hecho de la siguiente forma:

- Para el caso en el que se ha aplicado el Train Split, se ha entrenado el modelo para cada iteración de k con los ejemplos de entrenamiento, y luego guardábamos los resultados obtenidos al intentar predecir los ejemplos de validación (mostrados en el %R2 train) y los resultados obtenidos al intentar predecir los ejemplos de test(mostrados en el %R2 test)
- Para el caso en el que se aplicó la validación cruzada, la función de Sklearn nos devuelve un diccionario con los resultados sobre test y los resultados sobre train. Pues sobre estos resultados hacemos la media y los guardamos para más tarde mostrarlo por pantalla.

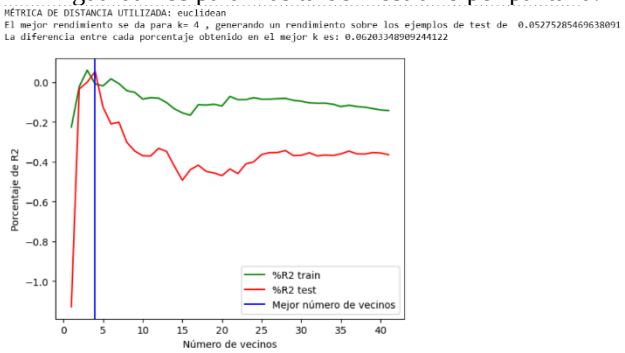

Figura 22. Modelo óptimo con caso Train Split + Estudio del ajuste

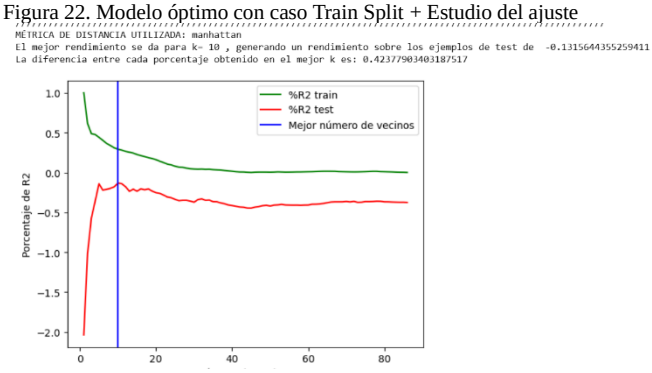

Figura 23. Modelo óptimo con caso Validación cruzada + Estudio del ajuste

Analizando ambos modelos óptimos , podemos percatarnos de una cosa, al utilizar todos los ejemplos , llega un momento el los modelos en los que a partir de un k la información que captura el modelo deja de influenciar tanto en el rendimiento, implicando que cuanto más ejemplos usemos y mayor k utilicemos, la información que se capture dejará de ser relevante.
En el modelo óptimo en el que se dividen los conjuntos de ejemplos también corroboran lo explicado, pero presentan tanto un mejor rendimiento como un mejor ajuste (lo parece).

Como todavía no tenía claro que caso utilizar, decidí ir más allá y aplicar la validación final sobre el modelo óptimo capturado con el Train Split. Para visualizar mejor el resultado, decidí representar la "supuesta" recta de regresión que deberíamos obtener:

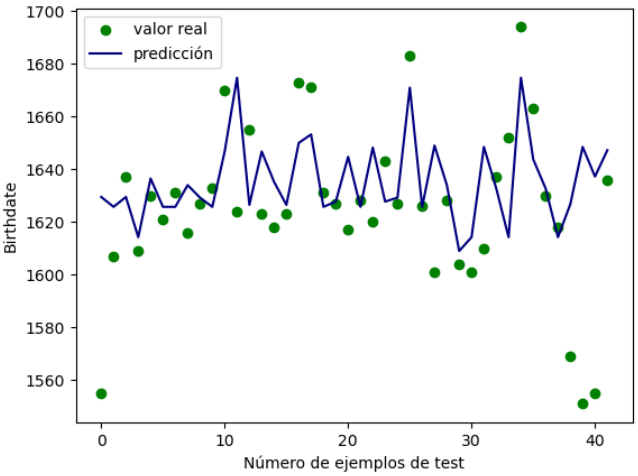

Figura 24. Validación final sobre Train Split

Como podemos observar en la gráfica, se está produciendo un sobreajuste de los datos , implicando que hemos utilizado pocos ejemplos para entrenar el modelo, y como de por si tenemos información compleja, las estimaciones son malas.

Resultado de este experimento: Hemos demostrado que nuestro conjunto de datos es tan pequeño que si aplicamos Train Split sólo vamos a obtener modelos sobre ajustados[8]. Por lo tanto, no podemos realizar la validación final tras encontrar el hiperparámetro óptimo.

Como en los modelos generados con la validación cruzada la diferencia entre R2 es mayor, esto puede significar que tenemos información compleja. Por lo tanto,**¿qué ocurriría si facilitamos la información?** Para intentar facilitarla, vamos ha facilitar los datos que más afectan a birthdate, que son los atributos relacionales. Y para facilitar los datos , lo que hemos hecho es lo siguiente:
1º Normalizar los atributos numéricos continuos de las métricas que no estaban normalizadas
2º Discretizarlas en 4 intervalos, (-inf,0)[0,0.5),[0.5,1)[1,inf)
3º Codificarlas denuevo para poder tratarlas

Una vez hecho esto, obtenemos el siguiente modelo con el siguiente ajuste:

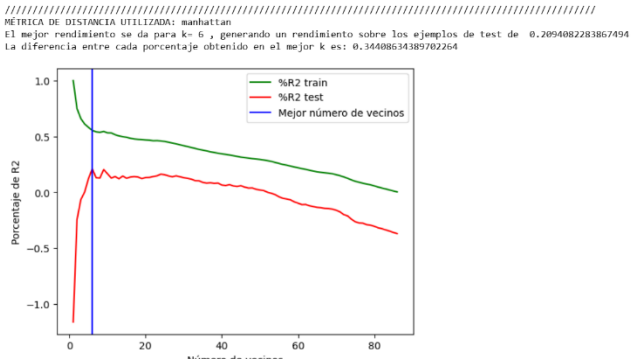

Figura 25.Modelo kNN óptimo facilitado + Estudio del ajuste

Resultado del experimento: Como podemos observar, el rendimiento a mejorado, y el ajuste que se presenta es mejor.

Resultados alcanzados:
Aunque hayamos escogido el número de vecinos óptimo y el ajuste parece el correcto, en el modelo se predice un modelo sobre ajustado, ya que hemos seguido utilizando poca información y ha hecho aumentar la complejidad del modelo. Esto se soluciona al facilitar los datos que complican la estimación de birthdate, que en este caso son todos los atributos relacionales. Tras facilitar los atributos (normalizando, discretizando y codificando lo necesario), obtenemos mejores resultados tanto en la cantidad de vecinos óptimo(tras facilitarlo, esta cantidad se minimiza) como en el ajuste que se da.
Ahora, el modelo Knn de regresión óptimo se da con 6 vecinos y utilizando la métrica manhattan, y como su R2 sobre test es >0, esto indica que el modelo es capaz de predecir la media de valores.

B. Modelo CART de clasificación sobre gender:

Este modelo se basa en la construcción de árboles de decisión, y nuestro objetivo a la hora de crearlos es encontrar los mejores pares atributo – umbral que minimice la impureza de los contenedores resultantes . Como gender es un atributo categórico binario, se puede aplicar CART.

Para construir el modelo óptimo, aquí el hiperparámetro que tenemos que optimizar es la profundidad del árbol que generamos. Lo primero que hicimos fue codificar todos los atributos que teníamos para después ir estudiando los resultados de cada modelo aplicando una profundidad diferente.
Al final del todo, el valor de nuestra profundidad será aquella profundidad que devuelva el mejor resultado sobre los ejemplos de test que devuelve la validación cruzada.

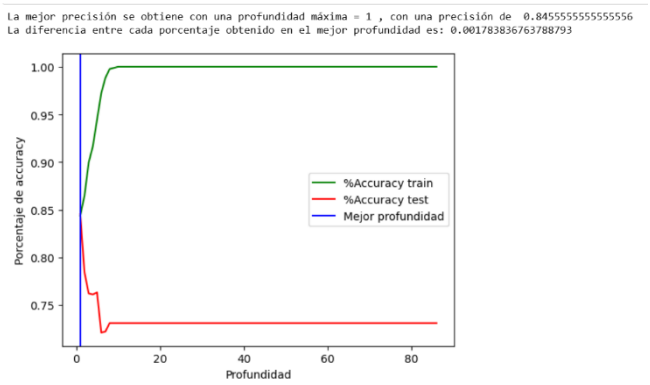

Figura 26. Gráfico del estudio del ajuste sobre CART

Como podemos apreciar en el gráfico, cuanta mayor es la profundidad del árbol mayor es la diferencia entre el accuracy medio obtenido por los ejemplos de entrenamiento y el accuracy medio obtenido por los ejemplos de test. Lo que implica que cuanto mayor sea la profundidad que tengamos en el árbol , menos preciso será el modelo y más sobreajustado estará. Como gender estaba asociado positivamente con la gran mayoría de atributos (menos los relacionales), **¿qué pasaría si quitamos los atributos que están asociados positivamente con gender?¿Obtendremos un peor modelo?**

Para probar esto, se decidió replicar el estudio del modelo, pero a la hora de preparar los atributos decidimos quitar esos atributos, y el resultado obtenido fue el siguiente:

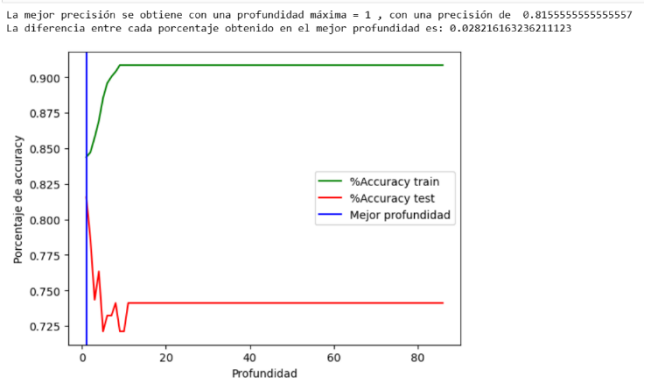

Figura 27. Gráfico del estudio del ajuste sobre CART ignorando los atributos más correlacionados con gender

Lo que ocurre es que se encuentra el primer par atributo-umbral óptimo y sigue ocurriendo lo mismo, el mejor resultado se obtiene con una profundidad de 1. A partir de 1, baja drásticamente el rendimiento. Esto puede significar que aunque se hayan ignorado los atributos más correlacionados de gender, sigue habiendo pares atributo-umbral que minimizan la impureza.

Resultados alcanzados:
- El modelo CART de clasificación con profundidad máxima 1 presenta un rendimiento óptimo, ya que tanto la tasa media de accuracy obtenida durante la validación cruzada se aproxima a ser 1, y el accuracy obtenido tras predecir las

clases de los ejemplos de prueba es 0.85, por lo que es un buen modelo.

- Que el mejor modelo se obtenga con una profundidad de 1 significa que se ha encontrado un par atributo-umbral tan bueno que si le añades más profundidad al árbol el rendimiento del modelo empeorará, y como gender está asociado positivamente sobre todo con deathdate,historical_significance y birthdate, estos atributos probablemente sean los pares atributo-umbral obtenidos, ya que si los ignoramos, obtenemos un peor resultado

C. Modelo RandomForest de clasificación para 'gender':

La diferencia entre RandomForest y CART es que RandomForest lo que hace es generar varios árboles de decisión y quedarse con el que genere la menor impureza posible, mientras que CART se encarga de buscar todas las combinaciones de pares atributo-umbral que minimicen la impureza de los conjuntos[9].

Para encontrar el modelo de RandomForest que óptimice la clasificación de gender , el procedimiento es similar al CART, ya que también nosotros estudiamos la profundidad máxima de los árboles de decisión que se vayan generando . La cantidad de árboles de decisión que se vayan generando es otro parámetro del modelo.

Empezamos primero codificando los atributos del conjunto de ejemplos. Luego, estudiaremos los resultados obtenidos al iterar sobre la profundidad de los árboles de decisión, y como a la vez vamos a ir estudiando el ajuste del modelo generado, vamos a representar al final los resultados obtenidos sobre test y sobre train que nos proporciona la validación cruzada . Al final del todo, el valor de nuestra profundidad será aquella profundidad que devuelva el mejor resultado sobre los ejemplos de test que devuelve la validación cruzada.

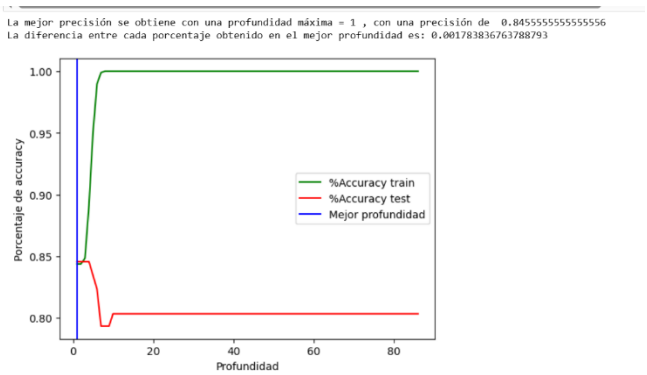

Figura 28. Gráfico del estudio del ajuste sobre RandomForest

Tras estudiar el ajuste del modelo, podemos observar que sucede igual que con CART, el mejor modelo de RandomForest de clasificación se da cuando la profundidad máxima del mejor árbol estimado es 1. Y como es posible modificar la cantidad de árboles de decisión generados, **¿qué pasaría si lo incrementamos?**

El procedimiento es el mismo, pero ahora lo que hemos hecho es iterar además sobre la cantidad de árboles utilizados.

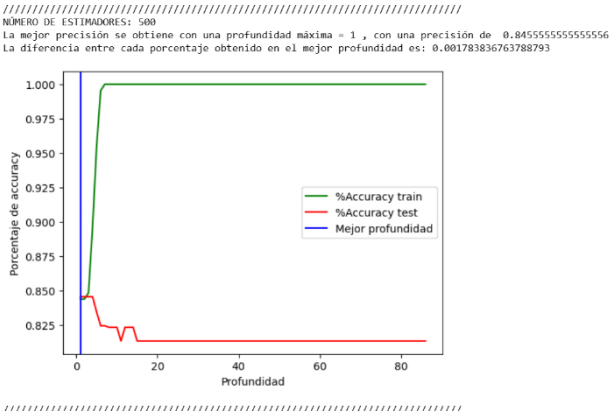

Figura 29. Estudio del ajuste sobre RandomForest

Como se puede apreciar, el resultado sobre cualquiera de los nuevos casos siguen siendo similares al resultado obtenido en CART, implicando que da igual cuantos árboles utilices, el mejor árbol siempre va a ser aquel que tenga como máxima profundidad uno, si es superior, el error entre resultados empieza a crecer y el resultado empeora.

Resultados alcanzados:

Al estar basado en CART, devuelve un resultado equivalente, pero se pueden observar 2 cosas:

- La velocidad del modelo es inferior a la del CART y se debe a que este algoritmo depende de la cantidad de árboles de decisión que se construyan
- Cuanto mayor es la profundidad del árbol, peor es la clasificación que realiza, dando a entender que entre todos los árboles de decisión que se han generado, el árbol idónea es igual que el árbol generado en CART.

VI. MODELO SELECCIONADO

El modelo ML óptimo para Quakers es un modelo CART de clasificación sobre el atributo 'gender' porque como tenemos pocos ejemplos y tenemos tantos atributos continuos con poca correlación entre ellas, clasificar un atributo categórico binario es más efectivo que realizar cualquier otra tarea sobre cualquier otro atributo de los datos que tenemos. Y debido a la complejidad de los datos y a la cantidad de ejemplos es tan pequeña, el modelo CART óptimo se obtendrá con sólo un par atributo-umbral y una profundidad de 1.

VII. CONCLUSIONES

- Las métricas relacionales aplicadas generan complejidad para este conjunto de datos, ya que analizando tanto las asociaciones que tienen los atributos relacionales con los atributos del csv , es posible de que el grafo original se realizase sin tener en cuenta las propiedades de cada nodo. Esto explicaría la complejidad que le añade a los modelos ML

construidos. Y como tenemos pocos ejemplos, esta complejidad afecta gravemente al modelo.

- Se ha descubierto que other_id es un atributo relacional y es por eso que esta poco correlacionado con la gran mayoría de atributos originales.
- En el modelo Knn de regresión sobre birthdate, los atributos que añaden demasiada complejidad al modelo son todos los atributos relacionales, y esto se debe a que estos no están asociados positivamente con birthdate, pero deathdate ayuda al modelo debido a que estos dos atributos están muy asociados positivamente (asociación moderada). Para obtener mejores resultados, se ha demostrado que es necesario facilitar los atributos que complican la estimación de birthdate, los cuales eran todos los atributos relacionales.
- Si intentamos aplicar Train Split en vez de la validación cruzada, esto nos devolverá siempre modelos sobre ajustados, ya que estaríamos disminuyendo la cantidad de ejemplos de entrenamiento y esto generaría más complejidad.
- En el modelo CART de clasificación sobre gender, aquí no afecta tanto las métricas relacionales, ya que gracias a la asociación positiva con el resto de atributos se obtienen modelos con rendimiento elevado y con el ajuste idóneo. Al comprobar que ocurriría si quitáramos los atributos de historical_significance, birthdate y deathdate, pudimos observar que se obtienen resultados parecidos.
- Si añadimos más pares atributo-umbral al árbol, obtenemos peores resultados de forma drástica, ya que empieza a capturar información poco relevante y/o compleja.
- En el modelo de RandomForest como obtenemos resultados parecidos al CART, esto demuestra que no importa cuantas combinaciones de árboles hagas, el mejor árbol de decisión que se puede obtener es un árbol de decisión con una profudidad máxima de 1.

REFERENCIAS

[1] Página web de la Sociedad Religiosa de los Amigos utilizada. https://es.wikipedia.org/wiki/Sociedad_Religiosa_de_los_Amigos. Consultada el 08/06/2023

[2] Página web de GitHub. https://github.com/melaniewalsh/sample-social-network-datasets/tree/master/sample-datasets/quakers. Consultada el 27/05/2023 y el 08/06/2023

[3] Página de explicación de las funciones y parámetros que implementa la librería de Pandas. https://aprendeconalf.es/docencia/python/manual/pandas/ . Consultada el 29/05/2023

[4] Página de Wikipedia que explica el método de Louvain. https://es.wikipedia.org/wiki/M%C3%A9todo_de_Louvain. Consultada el 29/05/2023 y el 13/06/2023

[5] Página web de la teoría del Coeficiente de agrupamiento utilizada. https://es.wikipedia.org/wiki/Coeficiente_de_agrupamiento. Consultada el 30/05/2023.

[6] Explicación de la correlación lineal. https://www.cienciadedatos.net/documentos/pystats05-correlacion-lineal-python.html. Consultada el 10/06/2023

[7] Página utilizada para verificar conceptos sobre Knn . https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos. Consultada el 13/06/2023

[8] Página utilizada para repasar y solucionar el sobreajuste. https://machinelearningparatodos.com/que-es-el-sobreajuste-u-overfitting-y-por-que-debemos-evitarlo/ . Consultada el 11/06/2023

[9] Página de Wikipedia que expande el contenido sobre el algoritmo de RandomForest. https://es.wikipedia.org/wiki/Random_forest . Consultada el 12/06/2023