Inteligencia de Negocio (2019-2020)
Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada

Análisis Relacional Mediante Segmentación

Grupo de prácticas: martes
Email: simondelosbros@correo.ugr.es

Simón López Vico

8 de diciembre de 2019

Índice

1. Introducción

En esta prácticas nos encargaremos de aplicar diferentes técnicas de aprendizaje no supervisado sobre el dataset `mujeres_fecundidad_INE_2018.csv` para agrupar en distintos grupos las instancias de éste según las características que prefijemos. Los algoritmos de agrupamiento que hemos utilizado son *KMeans*, *Mini Batch K-Means*, *Agglomerative Clustering*, *Birch* y *Mean-shift*; aplicaremos estos cinco algoritmos a cada caso de estudio que fijemos. Además, modificaremos los parámetros de *KMeans* y *Agglomerative Clustering* para comprobar como cambian los resultados obtenidos por el algoritmo en función de valores diferentes.

El dataset con el que vamos a trabajar se corresponde con los datos publicados por el Instituto Nacional de Estadística en 2018 sobre una encuesta de fecundidad realizada a mujeres españolas. Se dispone de un conjunto de 14556 respuestas a una encuesta con 463 variables sobre datos personales, biográficos, hogar, vivienda, padres, relaciones de pareja, hijos, fecundidad, estudios, empleo y creencias. Las variables utilizadas en el desarrollo de esta práctica serán las siguientes:

- `EDAD`: Edad en años cumplidos.
- `NHIJOS`: Número de hijos suyos o de su pareja.
- `NDESEOHIJO`: Número de hijos deseados para los que tienen hijos y los que no.
- `INGREHOG_INTER`, `INGRESOSPAR`: Ingresos mensuales netos del hogar en intervalos e ingresos mensuales netos de la pareja.
- `SITSENTI`: Situación sentimental actual.
- `TRABAJAACT`: Tiene en estos momentos un trabajo remunerado.
- `ESTUDIOSA`: Nivel de estudios alcanzados.
- `METROSVI`, `PAGOVI`: Metros cuadrados de la vivienda y porcentaje de los ingresos del hogar destinados a pagar la vivienda.
- `V_CRECERFELIZ`: Un niño necesita un hogar con su padre y su madre para crecer felizmente.
- `V_HOMOSEXUAL`: Está bien que las parejas homosexuales tengan los mismos derechos que las parejas heterosexuales.
- `V_PRIORIDADM`: Para una mujer, la prioridad debe ser su familia más que la carrera profesional.
- `V_TARDOM`: Los hombres deben participar en las tareas domésticas en la misma medida que las mujeres.

Usaremos estos atributos para fijar nuestros casos de estudio y aplicar un análisis de agrupamiento. Para crear un subconjunto que contenga las instancias con un valor en concreto usaremos el siguiente código:

```
subset = datos.loc[(datos[VARIABLE]==valor)]
```

Dentro de la cláusula `loc` podremos añadir cualquier expresión lógica, como por ejemplo personas con más de 50 años y que son médicos o militares. De esta manera, formaremos 4 subconjuntos a los que aplicaremos *clustering*, y sobre los que hablaremos a continuación.

Por otra parte, las métricas utilizadas para medir la calidad de cada algoritmo sobre el caso de agrupamiento fijado son el número de clústeres generados, el tiempo y los valores de Calinski-Harabasz y Silhouette. Estos dos últimos se definen como:

- Calinski-Harabasz: dicha medida se calcula como

$$CH = \frac{\sigma_B^2}{\sigma_W^2} * \frac{N-k}{k-1}$$

  donde k es el número de clusters generados, N el número de instancias, σ_W^2 la varianza global dentro de los clusters y σ_B^2 es la varianza global entre clusters. Por tanto, representará la relación de la varianza dentro de los clusters y entre ellos. A mayor valor, mejor resultado.
- Silhouette: esta métrica determinará como de similar es un objeto a su propio cluster (cohesión) en comparación con el resto clusters (separación). Su valor varía en $[-1, 1]$, donde un valor alto indica que el objeto está bien emparejado con su propio grupo y mal emparejado con los grupos vecinos. Por tanto, si la mayoría de los objetos tienen un valor alto, la configuración de clustering será apropiada, mientras que si el valor es bajo o negativo, la configuración de nuestro algoritmo será mejorable.

Finalmente, comentar que se han rellenado los valores perdidos en el dataset mediante la media y se han normalizado todos los datos con el siguiente código[1]:

```
def norm_to_zero_one(df):
    return (df - df.min()) * 1.0 / (df.max() - df.min())

for col in datos:
    datos[col].fillna(datos[col].mean(), inplace=True)

X=subset[usadas]
X_norm=X.apply(norm_to_zero_one)
```

2. Caso de estudio 1: Mujeres en paro con pareja

2.1. Descripción del caso

Vamos a estudiar el conjunto de mujeres en paro con pareja para comprobar de qué manera se agrupan función de su edad, número de hijos, los ingresos de su pareja, los metros

[1] Esto es un ejemplo, realmente se insertan los valores perdidos y más tarde, en la función `cluster_it`, se normalizarán.

cuadrados de su vivienda y el porcentaje de los ingresos del hogar destinados al pago de la vivienda. Presuponemos que las mujeres en paro con varios hijos y más de 40 años tendrá una pareja con un sueldo medio-alto para mantener a la familia; comprobemos mediante agrupamiento si se genera alguno de estos clusters.

Para empezar, obtendremos este subconjunto de la siguiente manera:

```
subset_paro = datos.loc[(datos['TRABAJAACT']==6) & (datos['SITSENTI']>1)]
usadas_paro = ['EDAD', 'NHIJOS', 'INGRESOSPAR', 'METROSVI','PAGOVI']
X_paro = subset_paro[usadas_paro]
```

El valor 6 para `TRABAJAACT` significará "No" y `SITSENTI>1` tomará los valores 2 (pareja con la que convive) o 3 (pareja con la que no convive).

Tras fijar este caso y generarlo, tendremos un subconjunto con 3642 instancias, al cuál le aplicaremos los algoritmos mencionados en la introducción, que inicializaremos con los siguientes parámetros (en los siguientes casos de estudio no incluiremos este código, pues es el mismo para todos):

```
k_means = cluster.KMeans(init='k-means++', n_clusters=5, n_init=5,
    random_state=SEED)
mini_batch = cluster.MiniBatchKMeans(init='k-means++', n_clusters=5, n_init
    =5, random_state=SEED)
agg_ward = cluster.AgglomerativeClustering(n_clusters=5, linkage='ward')
birch_ = cluster.Birch(n_clusters=5, threshold=0.1)
mean_shift = cluster.MeanShift(bandwidth=0.33)
```

Como vemos, utilizaremos el algoritmo KMeans generando 5 clusters y ejecutándolo 5 veces con diferentes semillas de centroide (`n_init=5`); Minibatch KMeans, que es una modificación de KMeans que reduce su tiempo de ejecución mediante *minibatches*, tendrá la misma inicialización. Como algoritmo de clustering jerárquico hemos elegido Agglomerative Clustering Ward, al que le tendremos que pasar el número de clusters que queremos que nos genere (`n_clusters=5`). Para Birch, aparte de fijar el número de clusters, tendremos que fijar el `threshold`, que determinará el umbral para que dos subclusters se unan. Finalmente, en Mean Shift hemos fijado el valor de `bandwidth`, con el que se formará más o menos clusters en función del conjunto de datos pasado. El valor de `SEED` será 26504148.

Una vez comentado todo esto, pasemos a ver la tabla con los resultados obtenidos:

	Nº clusters	Tiempo (seg)	C-H	S-C
KMeans	5	0.10	1151.234	0.24388
Minibatch	5	0.08	1093.096	0.23874
AGG	5	0.45	948.982	0.18812
Birch	5	0.27	913.937	0.17135
Mean-shift	7	29.08	544.750	0.24378

Tabla 2.1: Resultados obtenidos para el subset de mujeres en paro con pareja.

En los cuatro primeros algoritmos se han generado 5 clusters (tal y como especificamos en la incialización de estos) mientras que Mean Shift ha creado 7 clusters, debido al valor de `bandwidth` especificado. Aún así, de los siete clusters cuatro de ellos tienen menos de un 2 % de los datos, lo que reducirá el valor de C-H para este algoritmo.

Respecto al tiempo de ejecución, podemos ver la mejora que comentamos sobre Minibatch KMeans respecto a KMeans, aunque estudiando las métricas de calidad del algoritmo vemos que se reducen un poco. En este caso, si tuviéramos que elegir entre los dos algoritmos, será mejor opción usar KMeans, pues el tiempo de ejecución es ínfimo y nos favorecerá su mejora en CH y Silhouette. Por otra parte, Mean Shift ha sido el algoritmo que más tiempo ha tardado en ejecutar (29.08 segundos); esto es debido a que este algoritmo, aparte de clasificar las instancias en distintos clusters, también se encarga de calcular el número de clusters óptimo para la agrupación.

Estudiando las métricas utilizadas para comparar la calidad de los algoritmos, notamos que el mejor algoritmo es KMeans, que obtiene el mayor valor de C-H y el segundo mayor valor de Silhouetter; el mejor valor de Silhouette es para Mean Shift. Debido a los cuatro clusters generados con Mean Shift que contienen menos del 2 % de los datos, el valor de Shilouette mejorará, ya que en estos clusters tan pequeños las instancias están muy bien clasificadas al tener todas unas características muy concretas. Ésto nos puede llevar a descubrir outliers, nichos de gente con características fuera de lo común pero muy parecidos entre ellos.

Variemos ahora los parámetros de KMeans y Agglomerative Clustering para comprobar de qué manera cambian sus resultados. Comencemos con KMeans:

	Nº clusters	Tiempo (seg)	C-H	S-C
Default (K=5)	5	0.10	1151.234	0.24388
K=3	3	0.05	1467.908	0.27716
K=8	8	0.15	905.904	0.20273
init='random'	5	0.11	1151.197	0.24393

Tabla 2.2: Resultados obtenidos modificando los parámetros de KMeans.

Se ha modificado el número de clusters a generar y el método de inicialización del algoritmo a "random", que elegirá k observaciones al azar a partir de los datos de los centroides iniciales para la ejecución del algoritmo. En este último se utilizarán 5 clusters.

Podemos ver el que el tiempo de ejecución varía en respecto al número de clusters, reduciendo o aumentando en función del número de ellos. Además, cuanto menos clusters fijamos, mayor valor de CH y Silhouette obtenemos (y viceversa) dando a entender que el valor óptimo de K debe de ser menor que 5, el fijado por defecto. Por otra parte, si comparamos entre KMeans por defecto y el inicializado con "random" vemos que todos los valores de calidad se mantienen, dando a entender que el método de inicialización no variará los resultados sobre el caso de estudio que estamos trabajando.

Ahora comprobemos cómo cambia el comportamiento de Agglomerative Clustering si modificamos su `linkage` a "complete" o "average". El criterio de vinculación (`linkage`) determinará la distancia que se debe utilizar entre los conjuntos de observación para que el algoritmo fusione los pares de clusters que minimicen dicho criterio.

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
Ward	5	0.45	948.982	0.18812
Complete	5	0.39	623.489	0.08153
Average	5	0.42	387.685	0.17602

Tabla 2.3: Resultados obtenidos modificando los parámetros de Agg. Clustering.

Ward intentará minimizar la varianza de los dos clusters que juntemos, *Complete* usará la máxima distancia entre las observaciones de los dos clusters y *Average* tendrá en cuenta la media de las distancias entre todas los datos de cada cluster. En nuestro caso, vemos que la mejor opción es minimizar la varianza entre clusters, aunque el criterio *Average* también ha obtenido un buen valor para Silhouette, por lo que comprobaremos más adelante si ha encontrado algún outlier.

2.2. Interpretación de la segmentación

En esta sección comentaremos las conclusiones a las que hemos llegado mediante los algoritmos de agrupamiento ejecutados utilizando los distintos gráficos que hemos obtenido de ellos. Para ello, utilizaremos los resultados de KMeans con $K = 5$. Las proporciones de cada cluster son:

```
4:  1081 (29.68%)
2:   877 (24.08%)
3:   643 (17.66%)
1:   530 (14.55%)
0:   511 (14.03%)
```

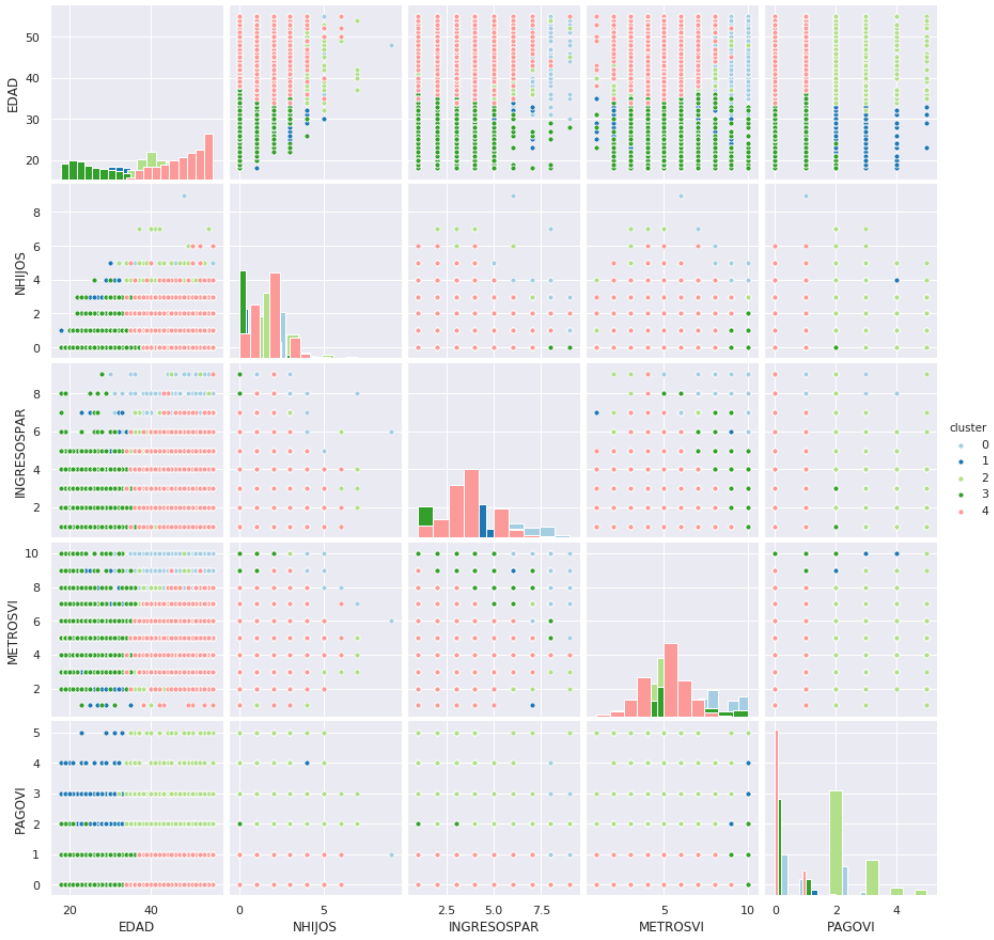

Figura 2.1: Scatter Plot para KMeans sobre el caso de estudio fijado.

La gráfica Scatter representará en dos dimensiones la distribución de los datos en función de dos de sus características. Mediante este gráfico podemos ver las fronteras que forman los puntos para determinar a qué cluster acabarán perteneciendo.

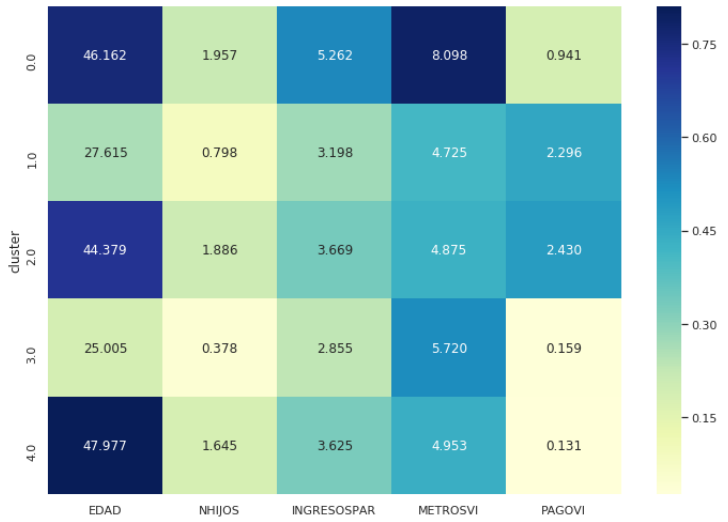

Figura 2.2: Heatmap para KMeans sobre el caso de estudio fijado.

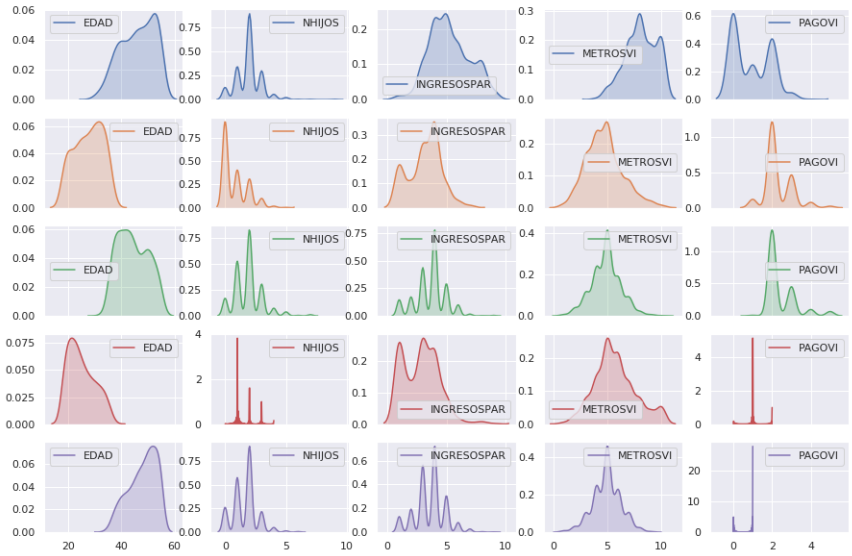

Figura 2.3: KPlot para KMeans sobre el caso de estudio fijado.

Utilizaremos estas dos gráficas para explicar los 5 clusters obtenidos para mujeres en paro con pareja. En las casillas del Heatmap se representarán las medias de cada atributo para cada cluster; como ya sabemos, la media no suele ser un buen representante de los datos, por lo que nos apoyaremos en el KPlot para ver la densidad de cada valor dentro de su cluster.

Viendo el heatmap para el cluster 0, podemos decir que contiene las mujeres de (aproximadamente) 46 años, 2 hijos, que su pareja gana entre 2000 y 2500 euros al mes, cuya casa tiene entre 121 y 150 m^2 y que utilizan menos del 20 % del sueldo para pagar la vivienda. Si nos vamos al KPlot, vemos que la edad se distribuye entorno a los 40-55 años y que, aunque haya un pico en 2 hijos, se dan varios valores para el número de hijos. Además, en el pago de la vivienda hay mucha gente que tiene el valor 0 (la vivienda ya está pagada), pero también hay bastante instancias que pagan mas del 20 % de su sueldo en vivienda. Por tanto, este cluster lo podríamos definir como el de mujeres amas de casa con una casa grande, pues no trabajan y tienen una vivienda medianamente grande, la mayoría en su propiedad. Aunque estén en paro no tienen por qué estar buscando trabajo, pues ya tienen la casa pagada y con el sueldo de su pareja pueden llevar la familia adelante.

En el cluster 1 vemos las personas de 27 años, la mayoría sin hijos, con pareja que gana entre 1000 y 1500 euros al mes y con una casa de entre 61 y 90 m^2 por la que gastan entre el 20 % y 40 % de su sueldo. Mediante KPlot vemos que algunas mujeres tienen 1 o 2 hijos pero la mayoría no tienen, y aunque la media del sueldo sea entre 1000 y 1500, hay varios casos en los que su pareja cobra menos de 1000 euros. Podemos definir este cluster como las mujeres jóvenes que aún no tienen hijos o tienen algún hijo, que están empezando a pagar su casa (por eso pagan más del 20 % de su sueldo) y que seguramente estén buscando trabajo para complementar el sueldo de su pareja, que es bastante bajo.

El tercer cluster (número 2) es similar al cluster 0, con una media de 44 años y dos hijos aproximadamente, pero en este caso el sueldo de la pareja es mucho menor, así como el tamaño de la vivienda. Por tanto, podremos definirlo como familias humildes que todavía no han terminado de pagar su casa.

Para el cluster número 3 tenemos gente de aproximadamente 25 años sin hijos, cuya pareja gana entre 500 y 1000 euros y con una casa mediana (entre 91 y 105 m2) de la que no tienen que aportar apenas para el pago. Con KPlot vemos que la mayoría de entrevistados de este cluster tienen 20 años, así como que no tienen hijos y que su pareja apenas cobra dinero. Podemos llamar a este cluster como mujeres universitarias, es decir, jóvenes que no tienen hijos y que apenas gastan en el pago de la vivienda, pues ese dinero viene de sus padres.

Por último tenemos el cluster 4, el cuál es el más grande de los 5 generados (1081 (29.68 %) instancias), que contendrá las mujeres de 48 años que tienen entre 1 y 2 hijos, que su pareja cobra entre 1000 y 1500 euros, con una casa no muy grande y que la tienen casi pagada. Aunque la media de hijos o de ingresos de su pareja sean 1.645 y 3.625, se dan muchos otros valores como podemos ver en el KPlot, por lo que podremos etiquetar este cluster como mujeres amas de casa, con una vivienda pequeña-mediana y con la casa pagada.

En los cluster 3 y 4 podemos ver que en el número de hijos y en el pago de la vivienda hay picos muy altos. Esto es debido a valores atípicos en la serie, instancias que se han colado en dichos clusters pero que no corresponden exactamente a la media de éste. Esto que comentamos se puede comprobar en el Box Plot, que representa los valores máximo y mínimo junto a los cuartiles de los datos contenidos en el cluster estudiado.

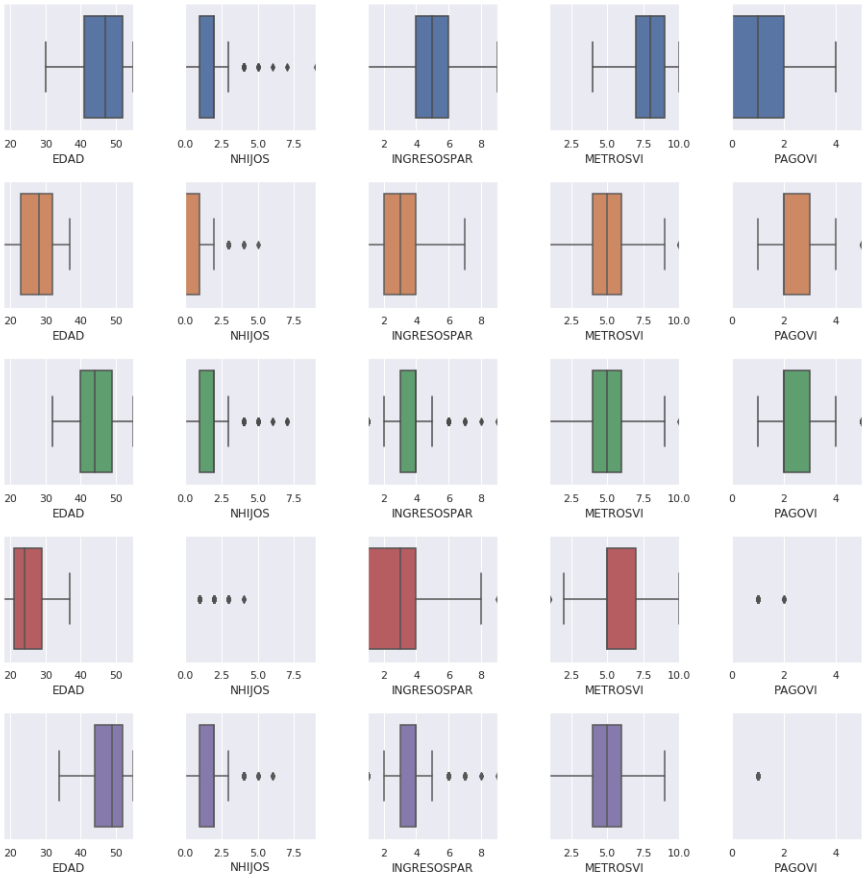

Figura 2.4: Box Plot para KMeans sobre el caso de estudio fijado.

3. Caso de estudio 2: Pensamiento liberal

3.1. Descripción del caso

Para este caso de estudio hemos intentado separar nuestro conjunto de datos en las mujeres con pensamiento liberal (en el siguiente caso de estudio lo haremos con el pensamiento conservador) utilizando los valores de las instancias concernientes a las preguntas sobre creencias. Para ello, elegiremos la gente que está a favor de que las parejas homosexuales

tengan los mismos derechos que las parejas heterosexuales, piense que una mujer no debe priorizar su familia frente a su carrera profesional y abogue porque los hombres deben participar en las tareas domésticas en la misma medida que las mujeres. Para aplicar clustering sobre este subconjunto utilizaremos lo valores de `EDAD`, `NHIJOS`, `INGREHOG_INTER` (ingresos en el hogar en intervalos) y `ESTUDIOSA` (estudios alcanzados). Este subconjunto contendrá un total de 4981 instancias. Lo obtendremos con el siguiente código:

```
subset_izda = datos.loc[(datos['V_HOMOSEXUAL']==1) & (datos['V_PRIORIDADM']
    ]==3) & (datos['V_TARDOM']==1)]
usadas_ideas = ['EDAD', 'NHIJOS', 'INGREHOG_INTER', 'ESTUDIOSA']
X_izda = subset_izda[usadas_ideas]
```

Veamos ahora los resultados que hemos obtenido de la ejecución de nuestros algoritmos:

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
KMeans	5	0.09	2314.444	0.29182
Minibatch	5	0.09	2276.795	0.28912
AGG	5	0.74	1843.196	0.22974
Birch	5	0.27	1409.362	0.19493
Mean-shift	4	91.32	1546.573	0.26717

Tabla 3.1: Resultados obtenidos para el subset de pensamiento liberal.

Al igual que en el caso de estudio anterior, los tiempos son muy pequeños para los cuatro primeros algoritmos mientras que Mean Shift, al tener que calcular el número de clusters, tarda más que el resto. En este caso, se han generado 4 clusters con Mean Shift, pero el cuarto de ellos solo contiene un 0.32 % de los datos (16 instancias).

El algoritmo KMeans vuelve a ser el mejor en las medidas de calidad del algoritmo (esta vez obteniendo también el mayor valor de Silhouette) y con Mean Shift vuelve a pasar lo mismo que anteriormente: al tener un cluster con tan pocos elementos, el valor de Silhouette aumenta respecto al resto de algoritmos. Para este subconjunto, Birch es el peor algoritmo a ejecutar (malos resultados en CH y Silhouette).

Veamos ahora como se comporta nuestro dataset frente a la modificación de parámetros iniciales en KMeans y Agglomerative Clustering. Los cambios realizados son los mismos que en el caso de estudio anterior.

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
Default (K=5)	5	0.09	2314.444	0.29182
K=3	3	0.06	2424.056	0.30231
K=8	8	0.15	1976.842	0.26562
init='random'	5	0.07	2315.042	0.29200

En este subconjunto volvemos a notar que el número de clusters a utilizar debe de ser menor que 5, pues las métricas son mejores para un número de clusters menor, mientras que si aumentamos el valor de K se empeorarán los resultados. Otra vez más, inicializar el algoritmo con "random" proporciona los mismos resultados que inicializarlo con "k-means++".

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
Ward	5	0.74	1843.196	0.22974
Complete	5	0.83	1182.047	0.15585
Average	5	0.86	999.959	0.20511

Modificando en `linkage` de Agg. Clustering volvemos a obtener el mejor resultado para *Ward*, y un buen valor de Silhouette para *Average*, que nos genera dos clusters con menos del 1 % de los datos.

3.2. Interpretación de la segmentación

Añadiremos los distintos gráficos obtenidos para el algoritmo KMeans de 5 clusters, y más adelante compararemos los resultados de este subconjunto con los resultados del pensamiento liberal.

La distribución de los clusters será:

```
1:  1090 (21.88%)
3:  1061 (21.30%)
0:  1006 (20.20%)
4:   976 (19.59%)
2:   848 (17.02%)
```

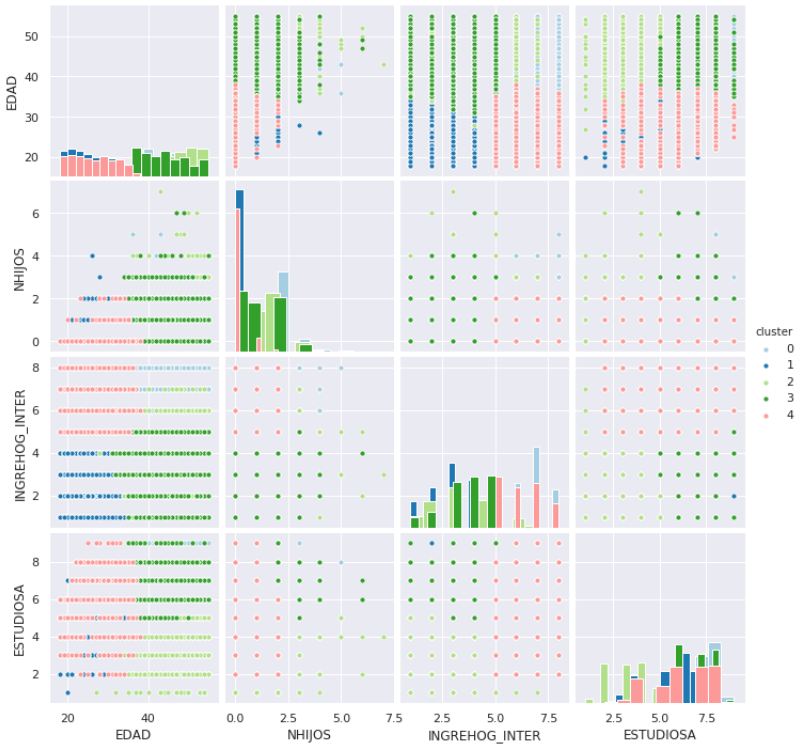

Figura 3.1: Scatter Plot para KMeans sobre el caso de estudio fijado.

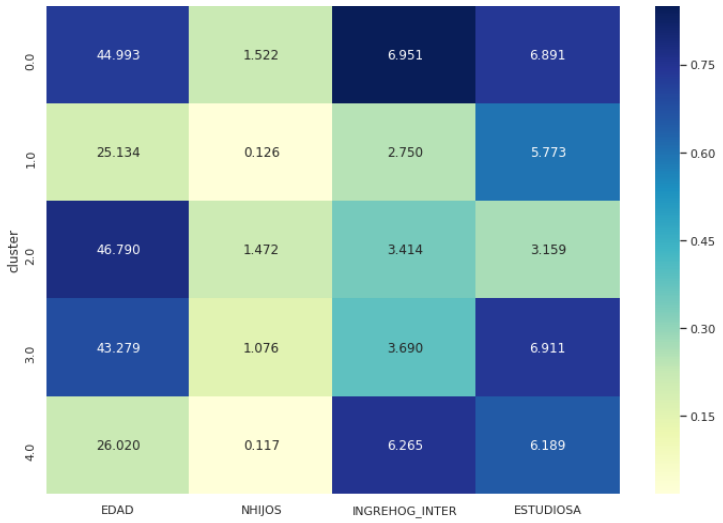

Figura 3.2: Heatmap para KMeans sobre el caso de estudio fijado.

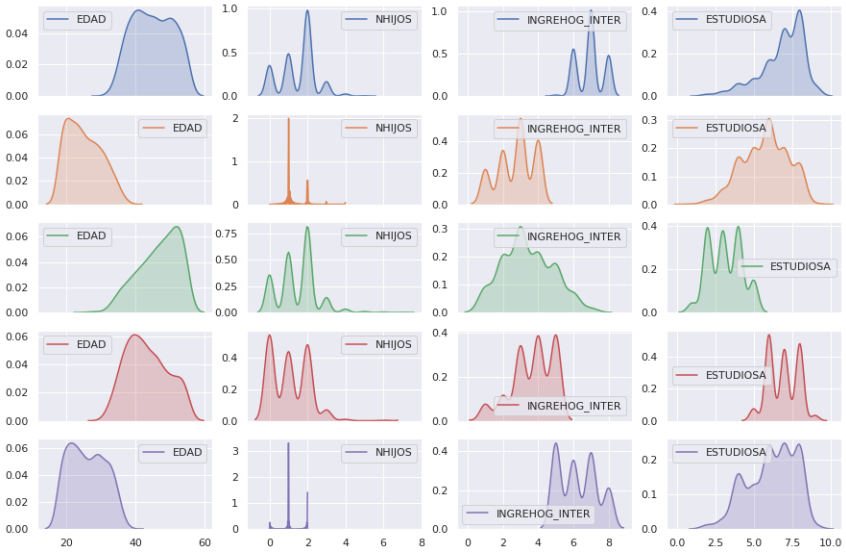

Figura 3.3: KPlot para KMeans sobre el caso de estudio fijado.

El cluster número 0 estará compuesto por las personas de (aproximadamente) 45 años, con 1 o 2 hijos, que tienen unos ingresos en el hogar de entre 2500 y 4000 euros que han alcanzado cursado estudios universitarios. Según la gráfica KPlot, este cluster contendrá mujeres de entre 40 y 50 años que la mayoría tienen 2 hijos, y cuyos ingresos en el hogar son generalmente altos. Además, los estudios realizados se encuentran entre los valores 7 y 9, que corresponden a títulos universitarios y doctorados. Podemos llamar a este cluster mujeres mayores de 40 con estudios universitarios.

El segundo cluster corresponde a gente joven, de aproximadamente 20-25 años, que no tiene hijos y que los ingresos en el hogar (que en su mayoría corresponderán a los de sus padres) son medios-bajos, con estudios superiores al bachillerato. En este caso vuelve a haber picos en la gráfica, que se corresponderán a valores atípicos como comentamos en el caso de estudio anterior. Podemos definir este cluster como mujeres universitarias que necesitan beca.

Continuamos con el cluster 2, que contienen mujeres de 46 años con dos hijos mayoritariamente, aunque muchas también con uno o ninguno. En este caso los ingresos se distribuyen en un rango más amplio, siendo la media de entre 1000 y 2000 euros al mes, y los estudios se distribuyen en torno a secundaria y grado medio.

Para el cluster número 3, tenemos mujeres entre 35 y 55 años con ninguno, uno o dos hijos, habiendo una cantidad similar de cada una de ellas. Éstas tienen unos ingresos en el hogar de entre 1000 y 2000 euros al mes y estudios universitarios, por lo que podemos nombrar a este cluster como madres con estudios universitarios.

En el último cluster tenemos valores similares al cluster número 1, aunque un poco mayores, lo que determinará un nivel mayor de estudios, pues ya habrán superado la carrera universitaria. No tienen hijos y tienen unos ingresos de entre 2000 y 2500 euros, que volvemos a suponer que se corresponden a los ingresos de sus padres. Nombraremos a este cluster como mujeres universitarias que no necesitan beca y postuniversitarias en busca de trabajo.

Más adelante podremos comparar estos resultados con los obtenidos por el pensamiento conservador.

4. Caso de estudio 3: Pensamiento conservador

4.1. Descripción del caso

Al igual que con el pensamiento liberal, intentamos separar las personas con ideas conservadoras utilizando las respuestas sobre los temas de valores, creencias y actitudes. Utilizaremos el valor de `V_CRECERFELIZ`, que discute si un niño necesita un hogar con su padre y su madre para crecer felizmente, y volveremos a utilizar la opinión sobre las parejas homosexuales, pero esta vez en contra de que tengan los mismos derechos que los heterosexuales.

Para aplicar clustering volveremos a utilizar los mismos atributos (edad, número de hijos, ingresos y estudios alcanzados), y con todo esto obtendremos un subconjunto de 653 elementos. Para conseguirlo utilizaremos este código:

```
subset_dcha = datos.loc[(datos['V_CRECERFELIZ']==1) & (datos['V_HOMOSEXUAL']
    ]==3)]
usadas_ideas = ['EDAD', 'NHIJOS', 'INGREHOG_INTER', 'ESTUDIOSA']
X_dcha = subset_dcha[usadas_ideas]
```

Los resultados obtenidos en la ejecución de nuestros cinco algoritmos son:

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
KMeans	5	0.05	250.722	0.25276
Minibatch	5	0.05	246.713	0.25259
AGG	5	0.01	213.094	0.21233
Birch	5	0.04	195.923	0.20144
Mean-shift	5	5.52	134.238	0.19630

Tabla 4.1: Resultados obtenidos para el subset de mujeres con ideas conservadoras.

Para todos los algoritmos se han generado 5 clusters, donde los 4 primeros tendrán aproximadamente los mismos elementos en cada cluster pero Mean Shift tendrá dos clusters con muy pocos elementos. Aún así, los datos contenidos en estas agrupaciones pueden ser interesantes dentro del caso de estudio.

El tiempo es muy bajo en todos los algoritmos, lo que es de esperar con los tiempos obtenidos en anteriores subconjuntos; además, al ser un subconjunto de datos relativamente pequeño, el tiempo de ejecución de Mean Shift también se ve reducido.

Una vez más volvemos a tener como claro ganador el algoritmo KMeans, que en este caso ha obtenido valores prácticamente iguales a Minibatch KMeans. Por otra parte, esta vez Mean Shift, aparte de tener un valor muy malo de CH, también lo tiene de Silhouette, por lo que la aplicación de este algoritmo sobre el subset fijado no es una buena idea.

Cabe destacar que, aunque los tiempos sean ínfimos, Agg. Clustering queda como el más rápido de todos, lo que es debido a la baja cantidad de datos a agrupar.

Comprobemos ahora como cambian los resultados respecto a la modificación de los parámetros de los algoritmos:

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
Default (K=5)	5	0.05	250.722	0.25276
K=3	3	0.04	288.314	0.26147
K=8	8	0.06	218.947	0.23408
init='random'	5	0.03	250.587	0.25510

Una vez más tenemos un resultado mucho mejor al escoger un valor menor de K, lo que nos hace comenzar a pensar que si tenemos que definir un nuevo subconjunto y aplicarle distintos algoritmos, estudiaremos el valor óptimo de KMeans con valores pequeños, menores que 5. Inicializar con "random" vuelve a servir para nada.

	Nº clusters	**Tiempo (seg)**	**C-H**	**S-C**
Ward	5	0.01	213.094	0.21233
Complete	5	0.01	170.308	0.19467
Average	5	0.01	88.323	0.21625

Para Agg. Clustering tenemos que *Ward* sigue siendo el mejor `linkage` a utilizar, y que *Complete* nunca da mejores resultados que el resto de vinculaciones (sobre nuestro dataset). Para *Average* volvemos a obtener un buen valor Silhouette, mejor que el de *Ward*, pero hay que tener en cuenta que nos ha generado clusters con 14, 2 y 1 elementos.

4.2. Interpretación de la segmentación

La distribución de las instancias en los distintos clusters es la siguiente:

```
0:    161 (24.66%)
4:    142 (21.75%)
1:    142 (21.75%)
3:    124 (18.99%)
2:     84 (12.86%)
```

Veamos las gráficas generadas por KMeans sobre nuestro conjunto de datos:

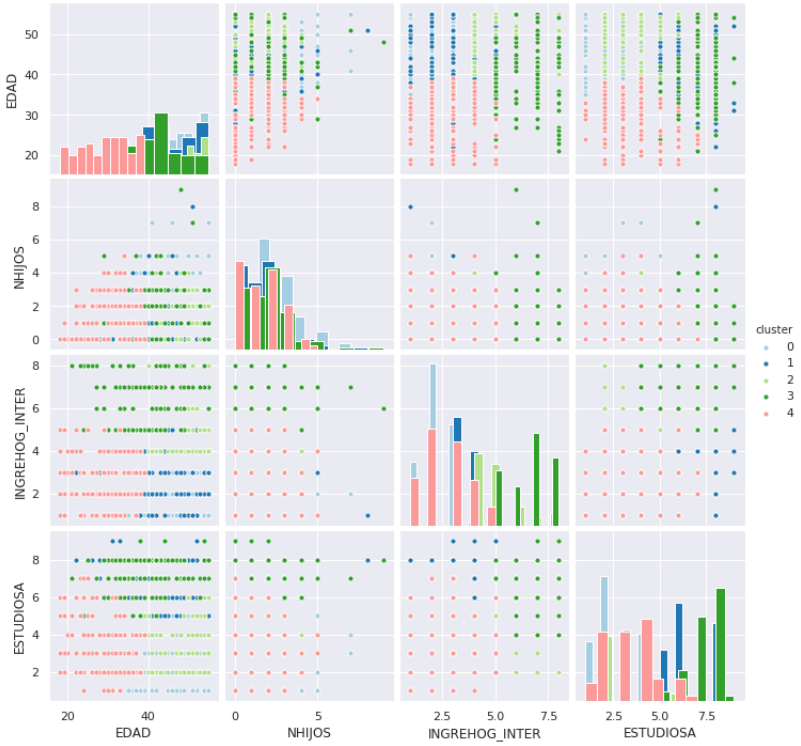

Figura 4.1: Scatter Plot para KMeans sobre el caso de estudio fijado.

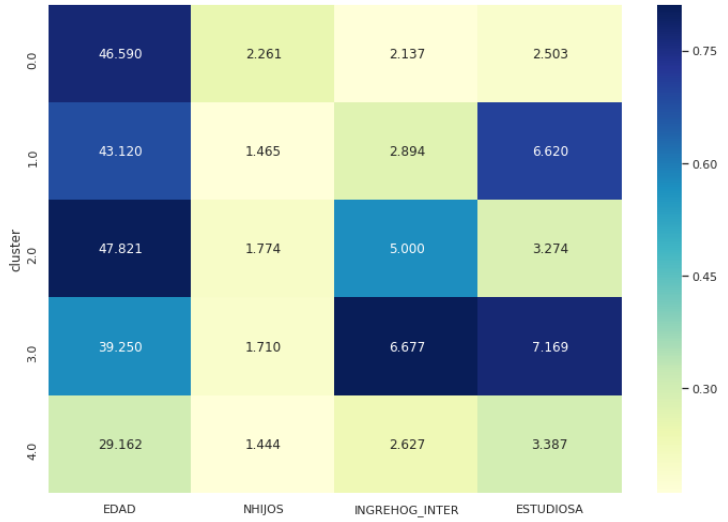

Figura 4.2: Heatmap para KMeans sobre el caso de estudio fijado.

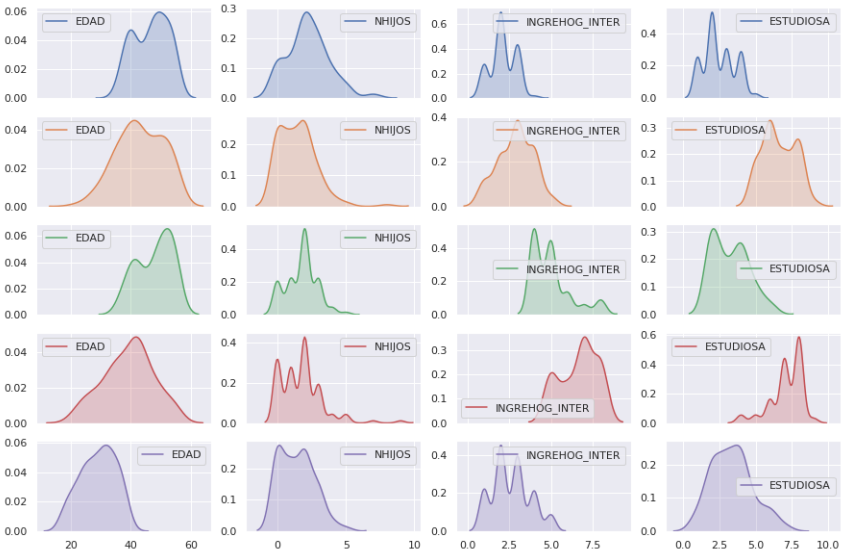

Figura 4.3: KPlot para KMeans sobre el caso de estudio fijado.

El primer cluster lo componen mujeres con una media de 46 años, 2 hijos y unos ingresos en el hogar bastante bajos. Además, el nivel de estudios es bastante bajo. Si nos vamos al KPlot, vemos que la mayoría de mujeres tienen aproximadamente 40 o 50 años (mayoritariamente 50) y que, aunque la media sea nivel de estudios bajo, hay varios casos de un nivel de estudios más avanzado.

En el cluster número 1 se encontrarán las mujeres de 43 años con uno o dos hijos y unos

ingresos algo mayores a los del anterior cluster, entre 1000 y 1500 euros al mes. Por otra parte, el nivel de estudios en este conjunto es mayor, moviéndose en valores de estudios universitarios.

Para el cluster número 2 vemos que hay una edad mayor a los anteriores casos y que es más frecuente tener 2 hijos que en los clusters 0 y 1. Además, los ingresos en el hogar ya son mayores para este grupo de gente (entre 2000 y 2500 euros). Aún así, estos ingresos deben ser de su pareja, pues el nivel de estudios vuelve a ser bajo, por debajo del bachillerato.

El cuarto cluster (número 3) distribuye las edades en un rango mayor, siendo la media 40 años aproximadamente. Lo conforman mujeres con entre 0 y 3 hijos, dándose más el valor de dos hijos, aunque muchas de ellas mujeres sin hijos. Los ingresos en el hogar son muy altos, lo que puede estar influido por el hecho de que su nivel de estudios también tiende a ser más alto que el resto, según vemos en el KPlot.

Por último, el cluster número 4 lo conforman las mujeres jóvenes, en torno a 25 y 35 años, que generalemente no tienen hijos (según el KPlot) y cuyos ingresos en el hogar tienden a ser medios, con una media de entre 500 y 1500 euros. La media de estudios alcanzados está por debajo del bachiller, aunque también hay casos de estudios más avanzados.

Veamos ahora la interpretación del dendograma obtenido para este caso de estudio mediante Agg. Clustering Ward. Pero antes de todo, hemos de entender como funciona Agglomerative Clustering para entender así lo que representa en dendograma.

Agg. Clust. comienza considerando cada instancia del conjunto como un cluster que contiene a esta instancia como único elemento. En cada iteración de ejecución del algoritmo, se determinará qué dos clusters son más similares entre ellos utilizando el criterio definido en el parámetro `linkage`. A continuación estos dos clusters se unirán en uno solo; este proceso se repetirá hasta formar un único cluster que contenga a todas las instancias de nuestro subconjunto. Ahora bien, si nosotros especificamos como parámetro que queremos que el algoritmo devuelve N clusters, Agg. Clustering dejará de buscar similitud entre clusters cuando llegue a la cantidad N.

Una vez entendido esto podemos ver el dendograma de Agg. Clust. Ward en el subconjunto de pensamiento conservador:

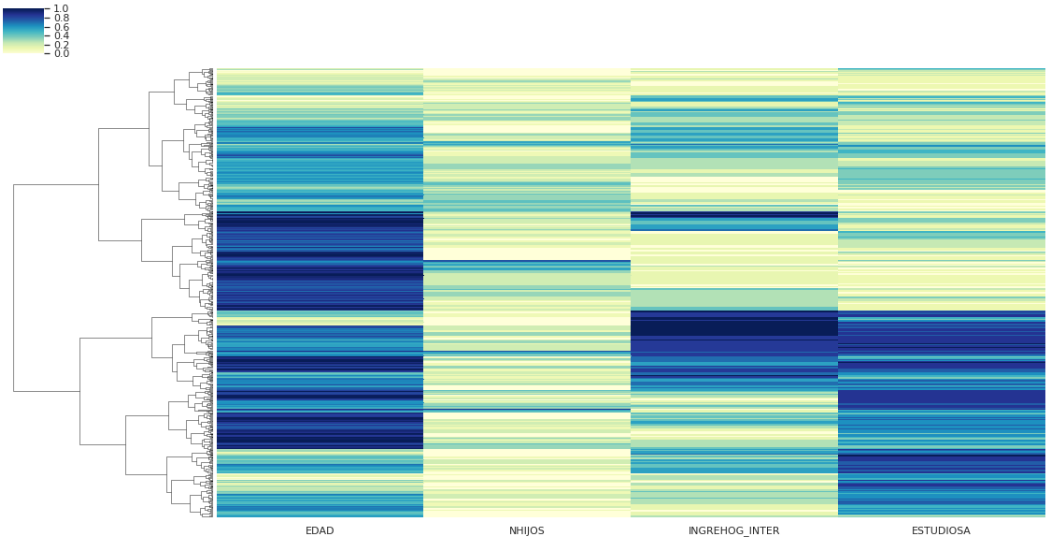

Figura 4.4: Dendograma para Agg. Clustering Ward sobre el caso de estudio fijado.

Como podemos ver, en el dendograma se representan todas las instancias del caso de estudio fijado y se muestra de qué manera se han ido *fusionando* entre ellas, hasta formar un único cluster. En nuestro caso, que hemos elegido `n_clusters=5`, se cortará el árbol de manera que solo tengamos 5 hojas.

Comparemos ahora este dendograma con su heatmap asociado:

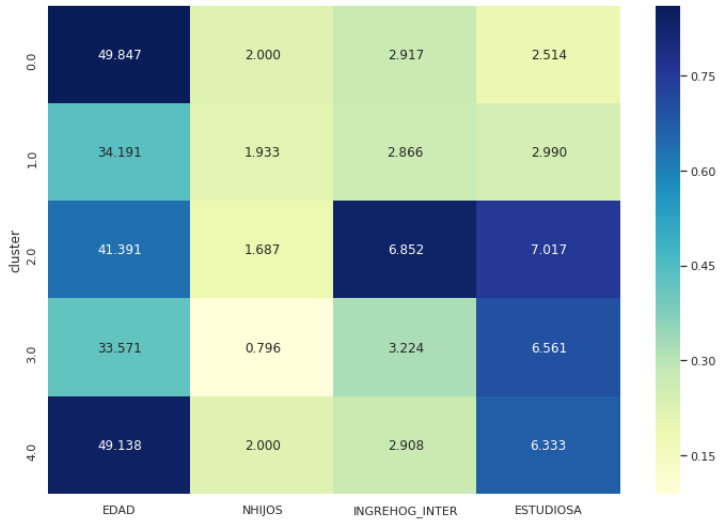

Figura 4.5: Heatmap para Agg. Clustering sobre el caso de estudio fijado.

Si comparamos los colores que aparecen en el dendograma y en el heatmap, vemos que las

instancias con mayor valor de `ESTUDIOSA` se encontraban en la parte baja del dendograma y se han juntado para formar tres clusters diferentes. Además, utilizando el dendograma, podemos ver que aunque la media de hijos se mueva en torno 1-2 hijos, hay instancias en el que el valor es mucho mayor (o mucho menor).

5. Bibliografía

- Transparencias de clase de teoría y prácticas junto a los tutoriales de Youtube subidos por Jorge Casillas.
- Calinski-Harabasz Index and Boostrap Evaluation with Clustering Methods, http://ethen8181.github.io/machine-learning/clustering_old/clustering/clustering.html
- Silhouette (clustering), https://en.wikipedia.org/wiki/Silhouette_%28clustering%29
- http://scikit-learn.org/stable/modules/clustering.html
- http://www.learndatasci.com/k-means-clustering-algorithms-python-intro/
- http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html
- https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tut
- http://seaborn.pydata.org/generated/seaborn.clustermap.html