Práctica 2

Análisis relacional mediante segmentación

Miguel Ángel Fernández Gutiérrez

Inteligencia de Negocio
5º Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada

mianfg.me

Índice

1. Introducción

En esta segunda práctica pasamos del aprendizaje supervisado al no supervisado, mediante el estudio de un problema de clustering. Realizaremos un análisis relacional usando diversos algoritmos de clustering a partir de los microdatos publicados por el Instituto Nacional de Estadística (INE) sobre la última encuesta de condiciones de vida (del año 2020). Este *dataset* dispone de 15043 respuestas y aproximadamente 200 variables sobre datos básicos del hogar, vivienda, exclusión social, renta, carencia, sobreendeudamiento, consumo y riqueza, entre otros. Esta encuesta se realiza con criterios armonizados a nivel de la Unión Europea (EU-SILC), lo que permitiría la comparación con datos de otros países miembros.

Nos dispondremos a elegir tres casos de estudio, que describiremos pertinentemente en cada apartado.

1.1. Algoritmos usados

En cada uno de ellos, realizaremos una comparación de diversos algoritmos de clustering. Analizaremos en más profundidad algunos de ellos, así como intentaremos refinar algunos parámetros. Estos algoritmos son:

- **K-Means.** Este algoritmo realiza un agrupamiento en un número prefijado de *clusters*. Partiendo de unos centros iniciales aleatorios, calcula los datos que se encuentran más cerca (a partir de una medida, en general la norma L_2) y recalcula los centros, repitiendo este proceso hasta que no haya más cambios. Debido a este uso de la distancia, genera clusters convexos. Es un algoritmo básico, muy sensible a *outliers*, por lo que esperamos que el resto de algoritmos mejoren los resultados ofrecidos por este.
- **Birch.** Es un algoritmo de clustering incremental, que crea un árbol con las características de los clusters guardando únicamente la información necesaria para disgregarlos. De cada nodo cuelgan ciertos *subclústers*, cuyo número es acotado por un factor de ramificación y un umbral (que determina si una instancia está lo suficientemente cerca del centro). Cuando llega una instancia, va descendiendo por el árbol hasta llegar a un cluster. Si no llega a éste y no ha superado el factor de ramificación, se crea un nuevo *subclúster*. Finalmente, prefijamos el número de clusters con el que quedarnos.
- **DBSCAN.** Es un algoritmo que no usa los centroides, lo cual diversifica las formas que adquiere el algoritmo. Asigna los clusters a las instancias de forma incremental, de acuerdo a dos parámetros. El primer parámetro, ϵ, determina cuándo un objeto es densamente alcanzable a partir de otro. El segundo es un número mínimo de objetos por los que debemos alcanzar a otro para que pertenezcan al mismo cluster. Habrá elementos que no resulten en ningún cluster, que el algoritmo incluye en el "cluster" `-1`.
- **MeanShift.** Del mismo modo que el anterior, este algoritmo usa el concepto de densidad. A partir de un radio prefijado, determina un número de clusters y desplaza los centros hacia regiones con mayor densidad. Por esto, los clusters generados también serán convexos.
- **Ward.** Este método será el único jerárquico que estudiemos. Según el nivel de corte, la jerarquía generada nos dará distintos agrupamientos. Este algoritmo va agrupando dos clusters en cada paso, tratando de minimizar la varianza.

1.2. Métricas usadas

Para poder medir el rendimiento de los algoritmos, usaremos las siguientes métricas:

- **Tiempo de ejecución** del algoritmo, en segundos.
- **Índice de Calinski-Harabasz.** Es un cociente entre la dispersión intra-cluster e inter-cluster. Un valor alto nos indica que los grupos son densos y están bien separados.
- **Índice de Davies-Bouldin.** Es un índice que compara la "similaridad" media entre los clusters, siendo la similaridad una medida que compara la distancia entre los clusters con el tamaño de los propios clusters. Un menor valor de este índice indica una mejor separación entre los clusters.
- **Coeficiente Silhouette.** Mide, en comparación con los de otros clusters, cómo de similares son los objetos de un mismo cluster. Toma valores en $[-1, 1]$, siendo un valor de 1 indicativo de que los clusters están muy concentrados y distantes entre sí, un valor de 0 que están superpuestos y -1 que el agrupamiento es incorrecto.
- **Número de clústers**, especialmente útil en el caso de los algoritmos que no lo tienen fijado *a priori*.

1.3. Implementación

Para la implementación, recurrimos al código proporcionado, que hace uso de las implementaciones de *SciKit-Learn*. El código está organizado en tres ficheros:

- `cluster.py` contiene las funciones principales de clustering. En especial, contiene la clase `ClusterAlgorithm`, que encapsula todas las operaciones de entrenamiento y métricas, y que permite almacenar de forma organizada varias instancias (con diversos parámetros) del mismo algoritmo, así como todos los resultados de la ejecución por cada instancia.
- `visualization.py` contiene todas las funciones de visualización.
- El notebook `case_i.ipynb` contiene todas las salidas de todas las ejecuciones que hemos realizado para el *i*-ésimo caso de esta práctica, organizadas en apartados. En caso de no disponer de Jupyter, el notebook ha sido exportado a un script `case_i.py`.

2. Casos de estudio

2.1. Ayudas a hogares andaluces

En la encuesta podemos ver varios valores referidos a las ayudas económicas que reciben los hogares. En primer lugar, realizaremos un **filtrado**, quedándonos únicamente con los hogares andaluces. Para ello, vemos aquellas instancias cuyo valor `DB040` sea `ES61`. Tras este filtrado nos quedamos con 1608 instancias.

Además, en nuestro estudio usaremos las siguientes variables:

- **`renta_disponible`** (`HY020`): renta disponible total del hogar en el año anterior al de encuesta. Esta variable es esencial pues nos permite apreciar la capacidad monetaria de las familias. Al realizar el agrupamiento, veremos cómo ciertos rangos en las ayudas se asocian a ciertos niveles de patrimonio.
- **`ayuda_familia_hijos`** (`HY050N`): ayuda por familia/hijos en el año anterior al de la encuesta.
- **`asistencia_social`** (`HY060N`): ingresos por asistencia social en el año anterior al de la encuesta.
- **`ayuda_vivienda`** (`HY070N`): ayuda para vivienda en el año anterior al de la encuesta.

Observando las variables, vemos cómo muchas son cero. Esto es algo normal, dado que el número de ayudas que se otorgan es bajo, y la muestra es pequeña (por ejemplo, observando los valores de `ayuda_vivienda` para nuestro *subset*, vemos que únicamente hay 20 instancias positivas). Independientemente de esto, realizaremos el análisis con estos valores, pues nos permitirá comprender cómo funcionan los algoritmos de clustering. Además, si la calidad de los datos fuese suficiente, podríamos extraer algunas conclusiones interesantes al realizar el mismo clustering en otras Comunidades Autónomas.[1]

Por otra parte, no debemos hacer tratamiento de valores perdidos para ninguna de las variables. Una vez comprobado esto, procedemos a ejecutar los algoritmos con las instancias que aparecen a continuación:

```
<ClusterAlgorithm [K-Means], 1 instance:
    init=k-means++, n_clusters=5, n_init=5, random_state=common.RANDOM_SEED>
<ClusterAlgorithm [Birch], 1 instance:
    branching_factor=25, threshold=0.15, n_clusters=5>
<ClusterAlgorithm [DBSCAN], 1 instance:
    eps=0.15, min_samples=5>
<ClusterAlgorithm [MeanShift], 1 instance:
    (no parameters)>
<ClusterAlgorithm [Ward], 1 instance:
    n_clusters=5, linkage=ward>
```

[1] Esta posibilidad se exploró pero, desgraciadamente, no se obtuvieron datos muy reveladores con ninguno de los algoritmos. En todas las comunidades autónomas los datos son escasos, con una gran cantidad de valores nulos.

Obtenemos las medidas que se reflejan en la Tabla 1.

	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette	Número de clusters
K-Means	0.102721	1348.639476	0.637565	0.481568	5
Birch	0.086947	256.611475	0.678905	0.550564	5
DBSCAN	0.118925	159.220980	1.043619	0.814450	2
MeanShift	24.117940	179.696538	0.433546	0.393259	22
Ward	0.142816	1081.540804	0.775540	0.406924	5

Tabla 1: Medidas para caso de estudio 1

En primer lugar, nos fijaremos en las medidas de tiempo. El algoritmo más exigente ha sido MeanShift, que casi consume medio minuto de ejecución. Si observamos la documentación de la implementación, vemos que, al no pasarle un parámetro `bandwith`, el propio algoritmo hace una estimación con complejidad cuadrática respecto al número de instancias. El resto han podido obtener los resultados en menos de un segundo, y obteniendo mejores medidas que MeanShift.

Podemos ver que, en el caso de MeanShift, se han obtenido muchos clusters, lo cual conlleva a que el coeficiente Silhouette sea el peor de todos (al estar más cerca de 0, nos indica que es el algoritmo que ha dado los clusters más superpuestos). El resto de medidas, sin embargo, penalizan este algoritmo notablemente.

Por otra parte, vemos cómo DBSCAN genera un número muy bajo de clusters: en concreto, solo es capaz de obtener un cluster con más del 99 % de los elementos, siendo el resto *outliers*. Esto hace que los objetos de este único cluster estén muy separados de los del resto, lo cual explica por qué el coeficiente Silhouette es el más elevado de todos, mientras que el coeficiente Calinski-Harabasz es muy bajo. Además, el índice Davies-Bouldin es el más alto de todos, lo que nos indica de nuevo que hay mucha separación entre ambos clusters. El hecho de que todos los ejemplos se hayan clasificado dentro de un cluster mayoritario nos indica que hemos escogido mal los parámetros del algoritmo – más adelante veremos cómo ajustarlos correctamente.

El resto de algoritmos han obtenido unos resultados similares, siendo K-Means el algoritmo con mayor índice Calinski-Harabasz. Esto nos indica que los grupos obtenidos son densos y están bien separados. Sin embargo, este valor no se ratifica con el resto de métricas. A continuación veremos con más detalle cómo ha realizado este algoritmo el agrupamiento.

Finalmente, vemos cómo Birch también obtiene buenos resultados: es por ello por lo que también lo vamos a estudiar en más detalle.

2.1.1. Estudio de K-Means

Vamos a centrarnos en el algoritmo K-Means para este caso, y validaremos si el número de clusters de 5 que hemos fijado en la mayoría de los algoritmos es el mejor posible para el agrupamiento.

En primer lugar, observaremos con más detalle cómo ha funcionado el algoritmo con nuestros resultados. En la Figura 1 podemos ver cómo tenemos mucha disparidad en el tamaño de los clusters. Tenemos un primer cluster con muchas instancias, seguido del cluster 4, y con un número considerablemente menor tenemos el resto.

Figura 1: Número de instancias clasificadas en cada cluster para K-Means (caso 1)

En la Figura 2 podemos ver con más detalle la dispersión de estos agrupamientos. Vemos cómo la variable `renta_disponible` es la que más separa los grupos. Esto es esperable, dado que es la que tiene más variabilidad de todas las variables del *dataset*. La variable `ayuda_familia_hijos` también sirve para realizar la separación, aunque en menor medida. Vemos cómo los clusters están medianamente bien definidos, aunque es notable el efecto de que muchos de los valores sean nulos en ciertas variables.

Podremos también comprender mejor qué elementos se encuentran en cada cluster con la ayuda de la Figura 3. En efecto, se evidencia cómo muchas de las variables son cero, es decir, cómo de la muestra se otorgan pocas ayudas. Sin embargo, podemos observar comportamientos ciertamente esperables. Vemos cómo las ayudas por los hijos se otorgan principalmente en las instancias del cluster 2, que es el segundo que tiene las rentas más bajas. Sin embargo, vemos cómo el cluster 0 tiene rentas aún más bajas pero no tiene prácticamente ayudas percibidas. Esto puede ser un indicador de que el número de clusters no es adecuado, y de que el algoritmo ha agrupado separando si las ayudas se perciben o no. En el resto de los grupos podemos apreciar cómo K-Means ha agrupado principalmente por renta, y cómo las ayudas que se dan son muy escasas en todos ellos. En la Tabla 2 aparecen las características de los grupos, en el formato $Q_1 - Q_3$ (primer cuartil-tercer cuartil), equivalentemente los cuantiles .25 y .75, respectivamente.

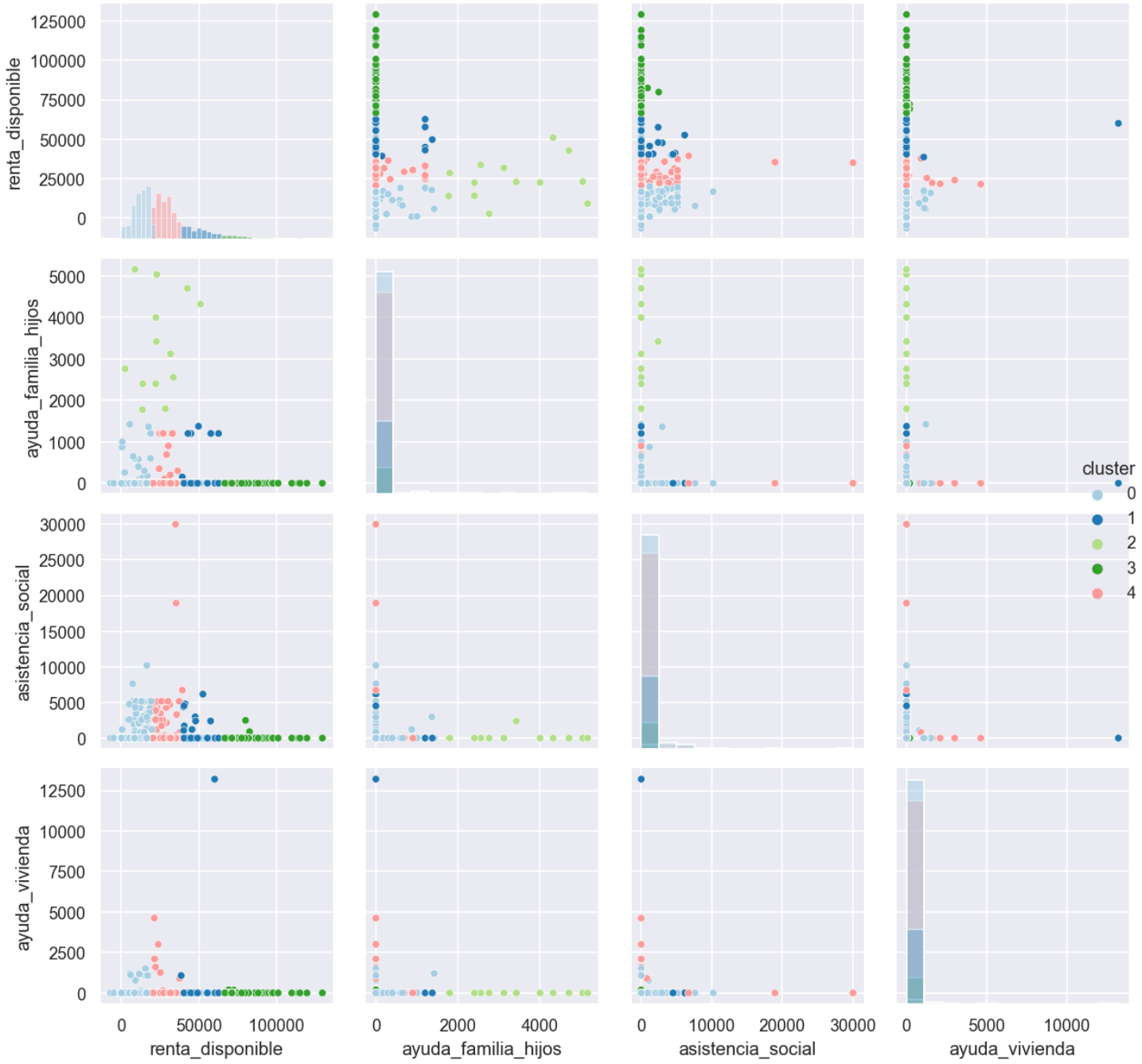


Figura 2: Matriz de dispersión para K-Means (caso 1)

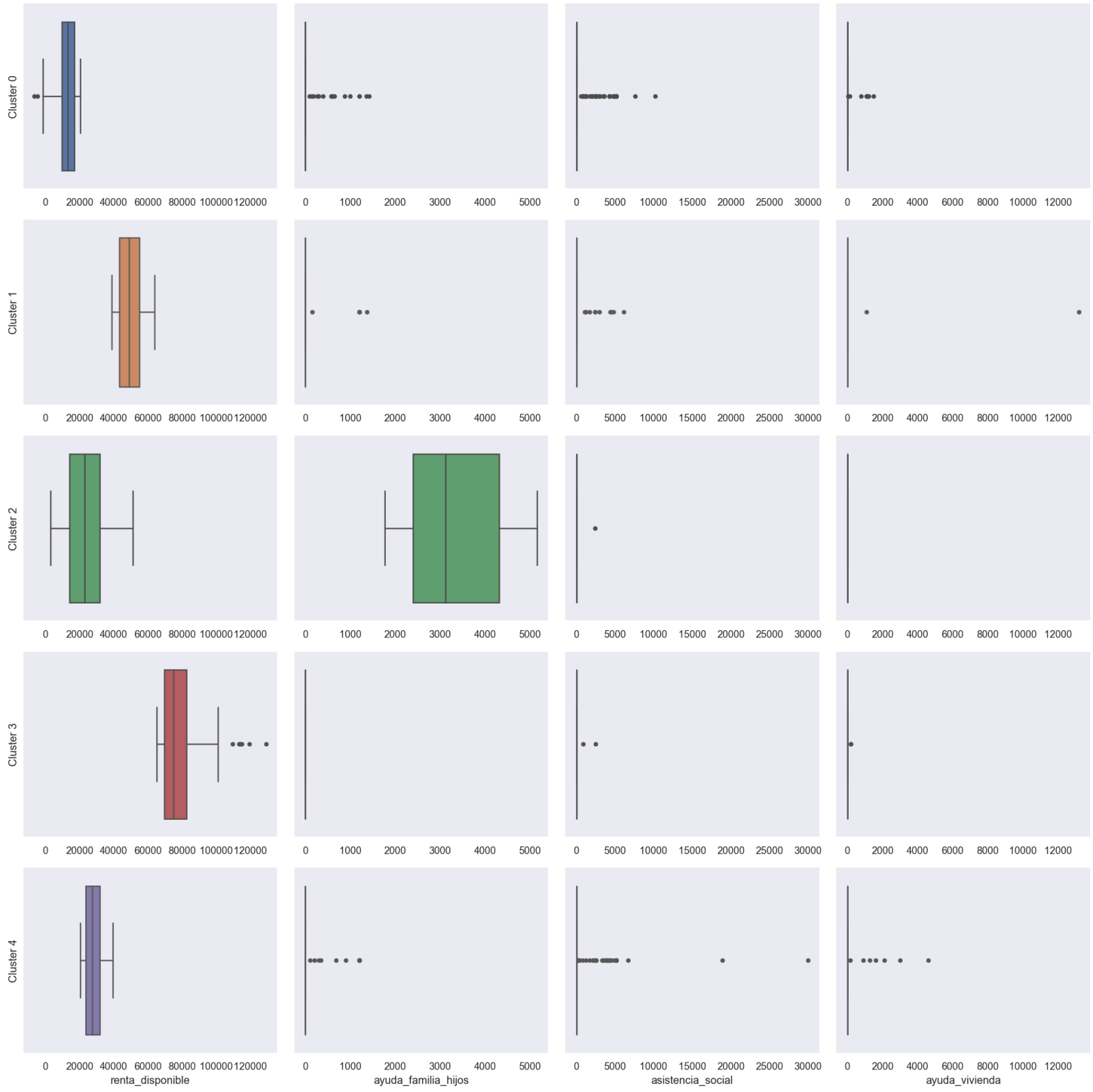


Figura 3: Boxplots para K-Means (caso 1)

Cluster	renta_disponible	ayuda_familia_hijos	asistencia_social	ayuda_vivienda
0	9458.05-16916.11	0.00-0.00	0.00-0.00	0.00-0.00
1	43180.67-55023.05	0.00-0.00	0.00-0.00	0.00-0.00
2	14132.70-31845.70	2400.00-4320.00	0.00-0.00	0.00-0.00
3	69605.30-82486.25	0.00-0.00	0.00-0.00	0.00-0.00
4	23686.33-31690.17	0.00-0.00	0.00-0.00	0.00-0.00

Tabla 2: Características de los clusters para K-Means (caso 1)

Vamos a intentar mejorar los resultados realizando un análisis de los parámetros. Ejecutaremos el algoritmo K-Means variando el número de clusters (`n_clusters`), e intentaremos ver cuál es el mejor.

n_clusters	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette
2	0.050041	1387.420297	0.659314	0.596836
3	0.071441	1176.399368	0.553835	0.609442
4	0.075313	1363.068374	0.622538	0.490041
5	0.093952	1348.639476	0.637565	0.481568
6	0.073983	1349.199440	0.630125	0.509544
7	0.090999	1334.106918	0.621179	0.520835
8	0.097112	1365.634272	0.627227	0.487691
9	0.117514	1490.303765	0.558773	0.490711
10	0.168061	1654.426284	0.524222	0.496788
11	0.160662	1795.093919	0.534119	0.492870
12	0.159459	1870.898131	0.538531	0.469177
13	0.187739	1927.760489	0.556353	0.484606
14	0.137825	1992.434192	0.569114	0.482392
15	0.145319	1917.237648	0.601724	0.496336
16	0.120897	2046.656088	0.579125	0.487899
17	0.156361	2065.189224	0.595579	0.495646
18	0.154650	2081.564262	0.581216	0.487371
19	0.136484	2117.479914	0.600233	0.489492

Tabla 3: Variación en parámetros para K-Means (caso 1)

(a) Calinski-Harabasz **(b)** Davies-Bouldin **(c)** Silhouette

Figura 4: Medidas ante variación en parámetros para K-Means (caso 1)

En la Tabla 3 vemos los valores exactos de las métricas al variar los parámetros, pero la Figura 4 es mucho más ilustrativa.

En este caso, es complicado obtener conclusiones a partir de los valores obtenidos, pues el comportamiento del algoritmo es poco predecible. Por una parte, vemos cómo el índice Calinski-Harabasz va aumentando (con un mínimo local en torno a los 15 clusters), mientras que el índice de Davies-Bouldin tiene un claro mínimo en torno a los 10 clusters. Esto nos indica que en los 10 clusters tenemos la mejor separación entre los clusters. Sin embargo, en otro a este entorno se reduce el coeficiente Silhouette. En definitiva, no vemos un número claro de clusters óptimo para nuestro problema. Esto nos indica que los datos que hemos obtenido no son muy buenos, siendo claro el efecto del gran número de parámetros nulos.

Una apreciación respecto a este caso de estudio. Evidentemente, no hemos obtenido unos resultados buenos, pero me resultó interesante incluirlo en la memoria para demostrar dos aspectos:

- En primer lugar, ver qué es lo que ocurre cuando los valores de ciertos parámetros son muy parecidos (en nuestro caso, muchas instancias tienen valores nulos). Vemos cómo si prefijamos el número de clusters, los agrupamientos tienden a separar las instancias no nulas de las nulas.
- En segundo lugar, ver cómo independientemente del *tuning* de los parámetros, hay casos en el que el algoritmo no obtiene resultados buenos. Esto puede deberse a que la separación óptima no consistiría en clusters convexos.

En definitiva, vemos cómo el algoritmo K-Means no nos da resultados buenos en este caso. Veamos qué ocurre con el algoritmo Birch, que dio resultados aceptables en la primera ejecución.

2.1.2. Estudio de Birch

Veamos cómo podemos mejorar los parámetros de Birch, pero para ello estudiaremos con más detalle los resultados ya obtenidos. Como podemos apreciar en la Figura 5, los clusters son muy desiguales en tamaño.

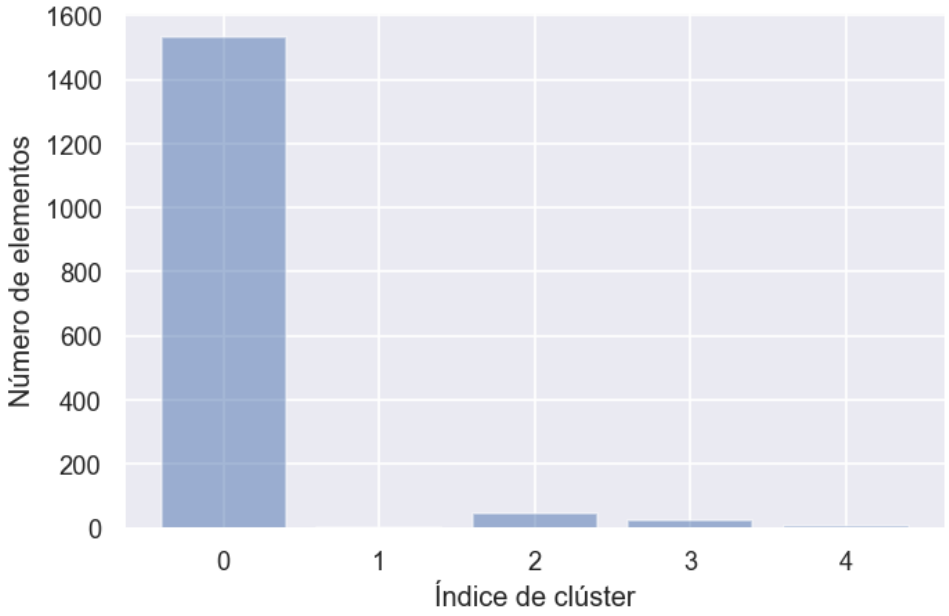


Figura 5: Número de instancias clasificadas en cada cluster para Birch (caso 1)

El análisis de la dispersión de la Figura 6 nos da más información sobre la forma de estos clusters. En este caso, `ayuda_familia_hijos` es la variable decisiva para la separación. Comparativamente, vemos una agrupación más clara, aunque es perceptible el cambio en el número de instancias en cada cluster.

En la Figura 7 y en la Tabla 4 podemos apreciar cómo los agrupamientos son mucho más acertados: si observamos los valores que toma `ayuda_familia_hijos`, vemos cuatro grupos diferenciados: los que perciben más ayudas (cluster 1), que coinciden con los que tienen un `renta_disponible` más bajo; los que perciben ayudas de menor cuantía (cluster 4), que tienen un `renta_disponible` algo más alto que el anterior; los que perciben ayudas de aún menor cuantía (cluster 3), que tienen un `renta_disponible` un tanto más alto; y finalmente tenemos los grupos que no perciben prácticamente ayuda (clusters 0 y 2), siendo el cluster 0 el de menor renta de ellos dos y el cluster 2 el de menor renta de todos los agrupamientos.

Viendo los *outliers* en estos dos últimos clusters, vemos cómo en el cluster 2, de mayor renta, no se perciben prácticamente ayudas, mientras que en el cluster 0 hay varias instancias que perciben ayudas.

En todos los casos los valores de `asistencia_social` y `ayuda_vivienda` son prácticamente inexistentes, pero sí que apreciamos que en el cluster 0, de baja renta, es donde se encuentran las instancias con mayor valor de estas variables. Esto nos puede dar una pista de por qué el algoritmo ha realizado la distinción entre los clusters 0 y 2: aunque ambos no perciban mucha `ayuda_familia_hijos`, el cluster 0 sí lo hace en `asistencia_social` y `ayuda_vivienda`, mientras que el cluster 3 no.

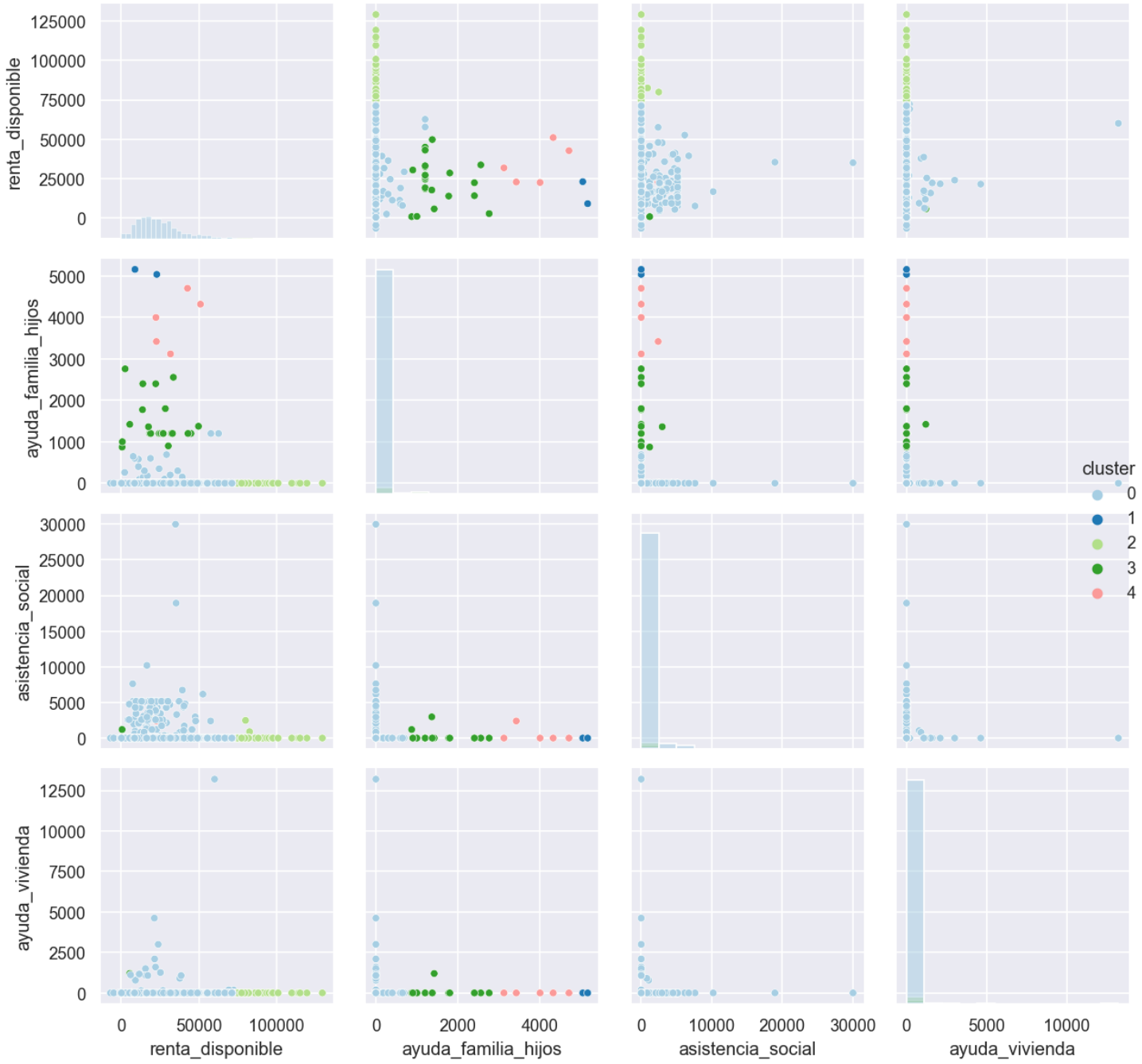


Figura 6: Matriz de dispersión para Birch (caso 1)

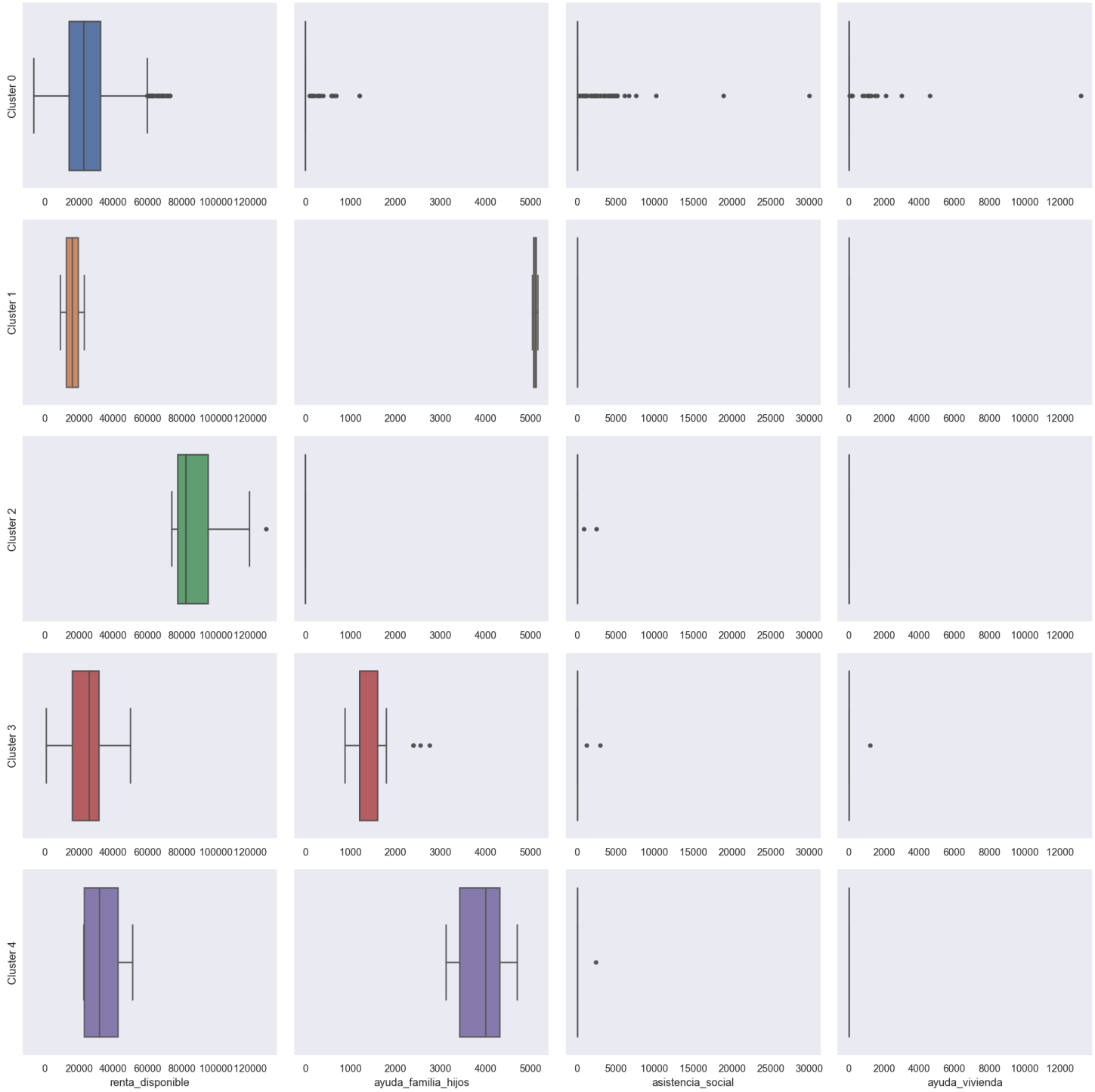


Figura 7: Boxplots para Birch (caso 1)

Cluster	renta_disponible	ayuda_familia_hijos	asistencia_social	ayuda_vivienda
0	9458.05-16916.11	0.00-0.00	0.00-0.00	0.00-0.00
1	43180.67-55023.05	0.00-0.00	0.00-0.00	0.00-0.00
2	14132.70-31845.70	2400.00-4320.00	0.00-0.00	0.00-0.00
3	69605.30-82486.25	0.00-0.00	0.00-0.00	0.00-0.00
4	23686.33-31690.17	0.00-0.00	0.00-0.00	0.00-0.00

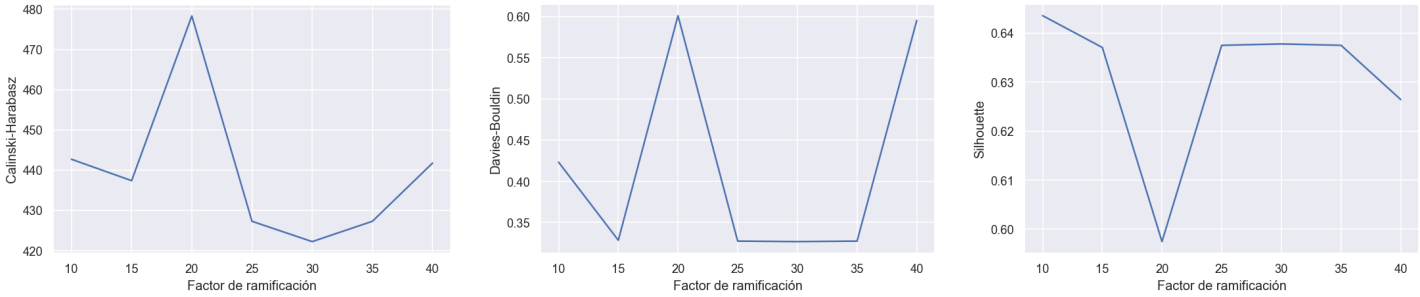
Tabla 4: Características de los clusters para Birch (caso 1)

Veamos el efecto de los parámetros `threshold` (umbral) y `branching_factor` (factor de ramificación). No hemos sometido a estudio el parámetro `n_clusters` porque se estudia exhaustivamente a lo largo de la memoria con K-Means.

threshold	branching_factor	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette	Núm. clusters
0.01	10	0.270074	961.051484	0.534130	0.509498	5
0.05	10	0.320712	442.707246	0.423138	0.643526	5
0.10	10	0.143869	527.890941	0.344149	0.631890	5
0.15	10	0.118283	256.611475	0.678905	0.550564	5
0.01	15	0.229861	550.519834	0.375209	0.636391	5
0.05	15	0.191177	437.381727	0.328507	0.637046	5
0.10	15	0.105700	520.374113	0.342738	0.632595	5
0.15	15	0.110058	256.611475	0.678905	0.550564	5
0.01	20	0.184067	1065.859659	0.531999	0.513194	5
0.05	20	0.136053	478.306766	0.600981	0.597465	5
0.10	20	0.105633	520.374113	0.342738	0.632595	5
0.15	20	0.088617	256.611475	0.678905	0.550564	5
0.01	25	0.218785	411.526294	0.602090	0.588249	5
0.05	25	0.153672	427.287965	0.327431	0.637484	5
0.10	25	0.102199	520.374113	0.342738	0.632595	5
0.15	25	0.098106	256.611475	0.678905	0.550564	5
0.01	30	0.190597	1011.455214	0.538655	0.498434	5
0.05	30	0.139531	422.239514	0.326848	0.637760	5
0.10	30	0.144001	520.374113	0.342738	0.632595	5
0.15	30	0.146250	256.611475	0.678905	0.550564	5
0.01	35	0.340183	1047.800908	0.530728	0.509316	5
0.05	35	0.256705	427.287965	0.327431	0.637484	5
0.10	35	0.109358	520.374113	0.342738	0.632595	5
0.15	35	0.114929	256.611475	0.678905	0.550564	5
0.01	40	0.201877	408.054659	0.602800	0.579821	5
0.05	40	0.105936	441.763333	0.595093	0.626419	5
0.10	40	0.108200	520.374113	0.342738	0.632595	5
0.15	40	0.110184	256.611475	0.678905	0.550564	5

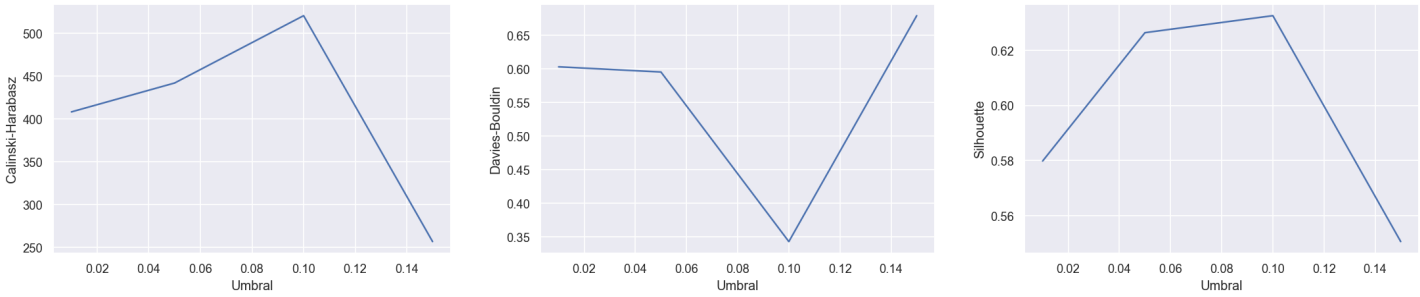
Tabla 5: Variación en parámetros para Birch (caso 1)

En las Figuras 8 y 9 podemos ver claramente el efecto de modificar estos parámetros.



(a) Calinski-Harabasz **(b)** Davies-Bouldin **(c)** Silhouette

Figura 8: Medidas ante variación en `branching_factor` (fijado `threshold`=0.05) para Birch (caso 1)



(a) Calinski-Harabasz **(b)** Davies-Bouldin **(c)** Silhouette

Figura 9: Medidas ante variación en `threshold` (fijado `branching_factor`=40) para Birch (caso 1)

Es interesante apreciar cómo, fijado un factor de ramificación, el tiempo decrece a medida que aumenta el valor del umbral. La explicación es clara: a mayor umbral, mayor probabilidad de que un objeto pertenezca a un cluster, y menos nodos del árbol hay que recorrer.

Fijado un cierto umbral, vemos cómo las métricas tienen un comportamiento muy consistente: tanto Calinski-Harabasz como Davies-Bouldin alcanzan un máximo local en torno al factor de ramificación 20, y es en este mismo entorno donde Silhouette alcanza un mínimo. Estos resultados nos indican que en torno a este factor obtenemos un peor agrupamiento. Si nos fijamos en el coeficiente de Silhouette, vemos cómo para un factor de ramificación menor obtenemos un mejor agrupamiento. Sin embargo, es en torno al factor 30 donde obtenemos las mejores métricas.

El caso del umbral es mucho más regular: fijado un factor de ramificación 40, vemos que las mejores métricas están en torno al umbral 0.10, que es donde los índices Calinski-Harabasz y Silhouette son mayores y Davies-Bouldin es menor, lo que nos indica un mejor agrupamiento.

Este comportamiento nos indica la importancia de hacer *hyperparameter tuning* en el modelo Birch.

2.1.3. Interpretación de la segmentación

Mediante el agrupamiento de este apartado, hemos encontrado las siguientes tendencias:

- Los encuestados que tienen mayor renta no reciben ayudas.
- Los encuestados de menor renta han sido separados en dos grupos: unos que reciben ayudas y otros que no. Por lo general, el hecho de que el encuestado tenga menor renta no implica que vaya a percibir ninguna ayuda.
- Las ayudas son percibidas por una parte muy baja de la población andaluza.
- Las ayudas que más perciben los encuestados son a la familia, seguidas de las ayudas a la vivienda.
- Las ayudas por asistencia social son las más dispersas de todas, con una mayor variabilidad en las cuantías.

2.2. Renta, transferencias e impuestos

Un caso interesante es conocer qué hacen los encuestados con su dinero. Para ello, usaremos las siguientes variables:

- **`renta_disponible`** (`HY020`): renta disponible total del hogar en el año anterior al de encuesta. Esta variable es esencial pues nos permite apreciar la capacidad monetaria de las familias.
- **`transferencias_percibidas`** (`HY080N`): transferencias periódicas monetarias percibidas de otros hogares en el año anterior al de encuesta.
- **`transferencias_abonadas`** (`HY130N`): transferencias periódicas monetarias abonadas a otros hogares en el año anterior al de encuesta.
- **`impuesto_renta`** (`HY140G`): impuesto sobre la renta y cotizaciones sociales.

En este caso, no tenemos que tratar con valores perdidos, y las variables tienen mucha más información que en el caso anterior, teniendo menos valores nulos. Ejecutaremos los algoritmos con las siguientes instancias:

```
<ClusterAlgorithm [K-Means], 1 instance:
    init=k-means++, n_clusters=5, n_init=5, random_state=common.RANDOM_SEED>
<ClusterAlgorithm [Birch], 1 instance:
    branching_factor=25, threshold=0.08, n_clusters=5>
<ClusterAlgorithm [DBSCAN], 1 instance:
    eps=0.15, min_samples=5>
<ClusterAlgorithm [MeanShift], 1 instance:
    (no parameters)>
<ClusterAlgorithm [Ward], 1 instance:
    n_clusters=5, linkage=ward>
```

Obtenemos los resultados de la Tabla 6, que son muy similares a los del caso 1 (Tabla 1). La única diferencia destacable con el caso anterior es que los índices son, en general, mejores. Esto puede indicarnos que estamos ante un conjunto de datos que es más agrupable. Además, vemos cómo los tiempos de ejecución son mayores debido a que estamos usando todo el *dataset*.

	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette	Número de clusters
K-Means	0.246937	8272.116087	0.779611	0.420262	5
Birch	1.439632	3629.008906	0.531974	0.627073	5
DBSCAN	9.908936	741.932487	0.980895	0.885794	2
MeanShift	368.519296	581.174243	0.599671	0.457880	71
Ward	21.938677	7181.290490	0.702107	0.544362	5

Tabla 6: Medidas para caso de estudio 2

Deberemos recurrir al estudio individualizado de algunos de estos algoritmos para obtener más conclusiones. En concreto, estudiaremos los algoritmos K-Means y Ward.

2.2.1. Estudio de K-Means

La segmentación obtenida con este algoritmo puede apreciarse en la Figura 10. De nuevo, tenemos bastante disparidad entre los clusters obtenidos, teniendo los clusters 0 y 4 una cantidad considerablemente mayor de instancias.

Figura 10: Número de instancias clasificadas en cada cluster para K-Means (caso 2)

De nuevo, mediante la figura 12 concluimos que `renta_disponible` e `impuesto_renta` son las variables que más separan los datos. Además, vemos cómo ambos valores son proporcionales: a mayor renta disponible, mayores impuestos. Veamos gracias a esta Figura 12 y a la Tabla 7 cómo se distribuyen estos clusters.

Vemos que el poder adquisitivo puede organizarse en orden ascendente mediante los clusters 0, 4, 1, 3 y 2. Además, vemos que, por lo general, la gente con menor renta percibe más transferencias de las que abona. Además, el importe de las transferencias abonadas aumenta con la renta (y con los impuestos), y el de las transferencias recibidas, por el contrario, va disminuyendo.

Esta información concuerda perfectamente con la que nos otorgan los centroides de los clusters, cuyos valores podemos ver en la Figura 13.

Adicionalmente, de los diagramas boxplot también podemos ver cómo los clusters son mucho más compactos que en el caso 1, lo cual concuerda con los mejores índices obtenidos.

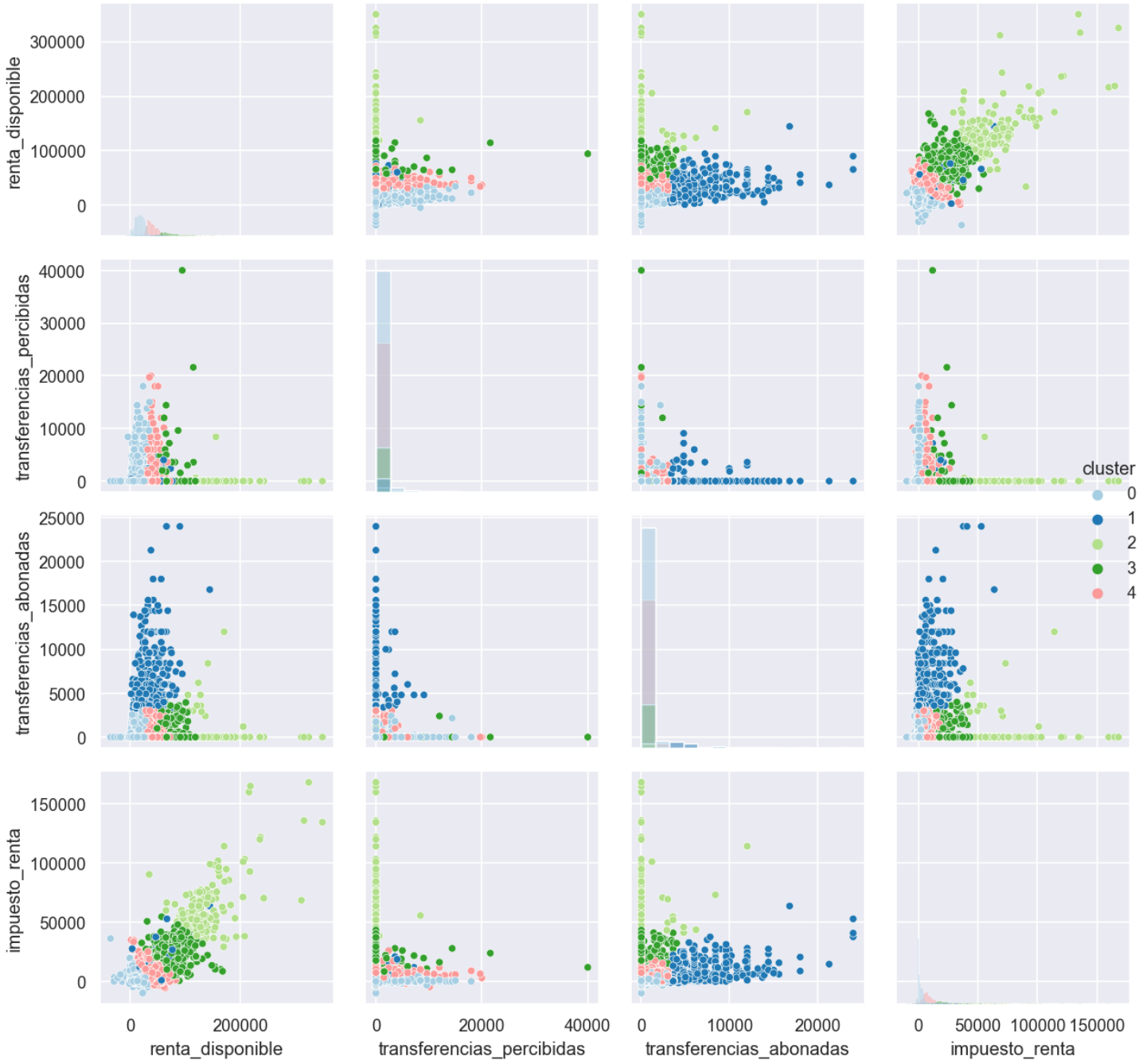


Figura 11: Matriz de dispersión para K-Means (caso 2)

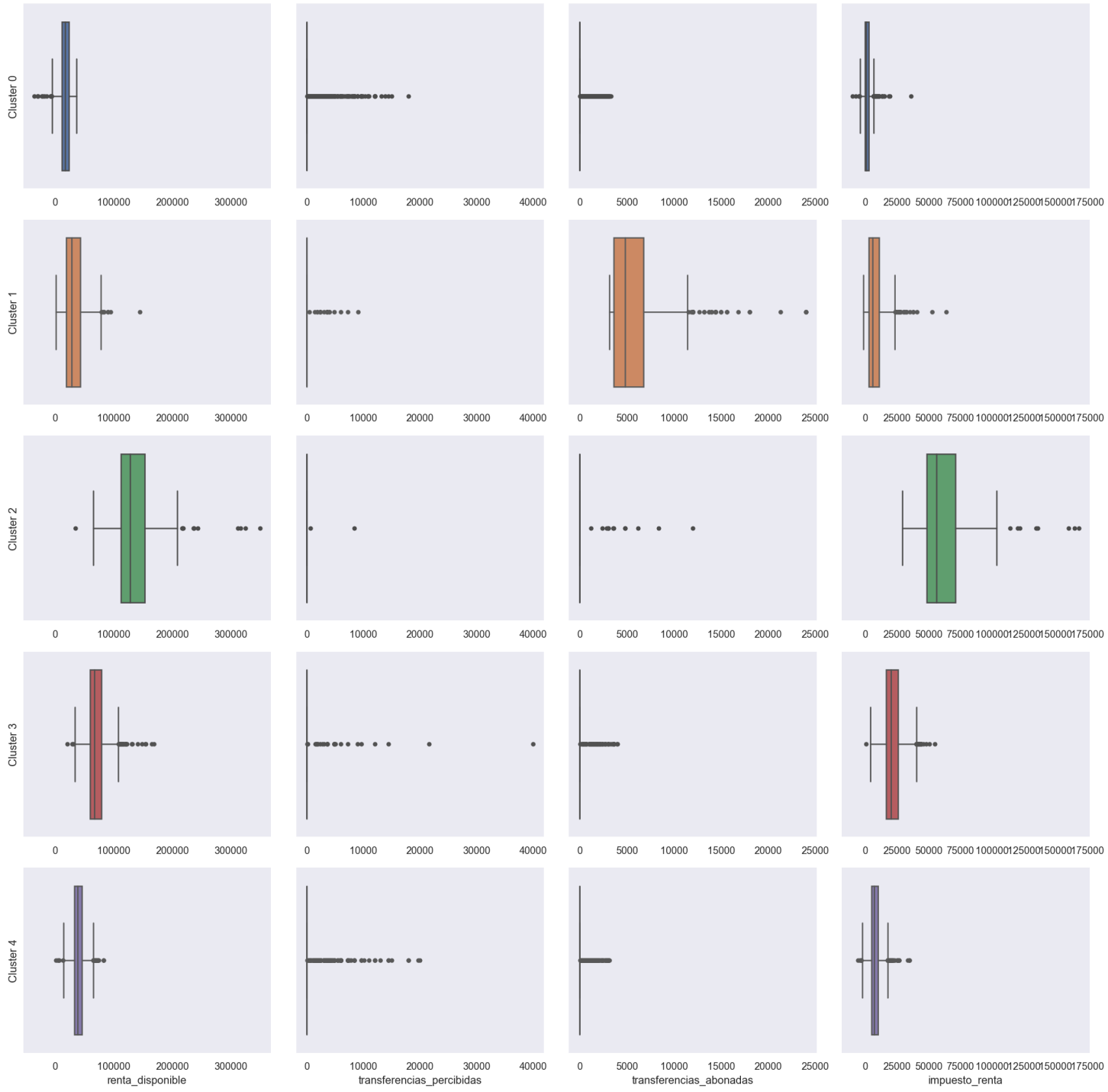


Figura 12: Boxplots para K-Means (caso 2)

Cluster	renta_disponible	transferencias_percibidas	transferencias_abonadas	impuesto_renta
0	11595.40-23002.00	0.00-0.00	0.00-0.00	0.00-2690.99
1	18666.90-42438.65	0.00-0.00	3600.00-6750.00	2737.55-10993.55
2	112067.12-152385.30	0.00-0.00	0.00-0.00	48309.40-70891.98
3	59818.18-79018.68	0.00-0.00	0.00-0.00	16294.75-25874.92
4	32864.95-45668.28	0.00-0.00	0.00-0.00	4944.28-10067.45

Tabla 7: Características de los clusters para K-Means (caso 2)

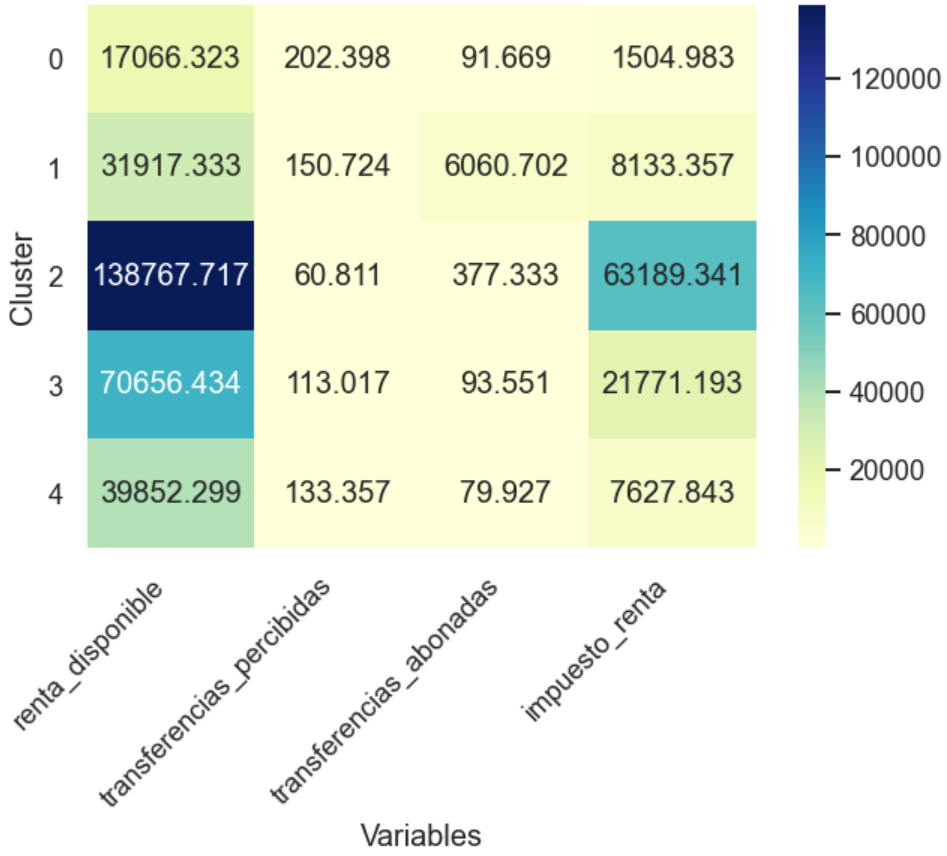


Figura 13: Heatmap de centroides para K-Means (caso 2)

A pesar de que hemos obtenido unos grupos bastante buenos, volveremos a analizar el comportamiento de K-Means con la variación del número de clusters (`n_clusters`). Dado que los datos de este caso tienen mucha más variabilidad, esperamos poder obtener conclusiones más significativas.

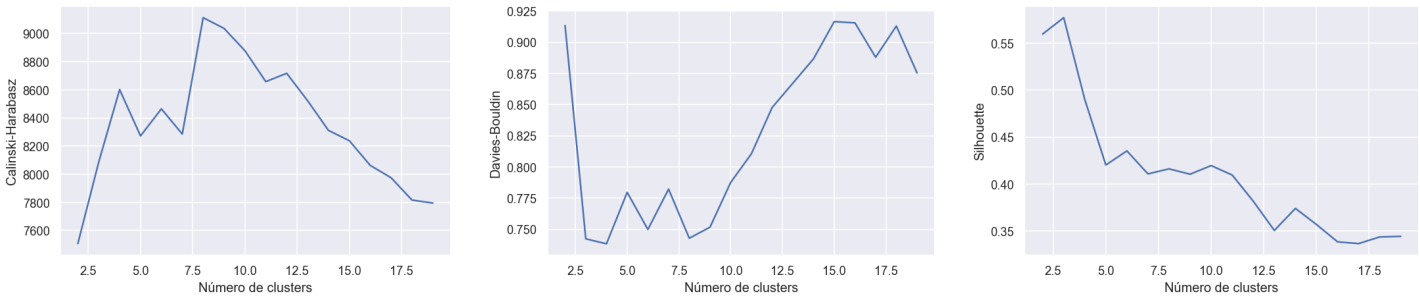
En la Tabla 8 podemos ver el cambio en las métricas en función del cambio del número de parámetros. Vemos cómo la tendencia en tiempo es a aumentar conforme aumenta el número de clusters. Esto puede deberse a que, al tener más clusters, son necesarias más iteraciones, dado que el cálculo de centroides y distancias tiene la misma complejidad independientemente del número de clusters prefijado.

Veamos la Figura 14 para analizar el comportamiento de las métricas conforme variamos el número de clusters.

n_clusters	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette
2	0.108243	7507.056710	0.913400	0.559495
3	0.140831	8086.837960	0.742330	0.576909
4	0.196893	8601.486356	0.738363	0.489982
5	0.251995	8272.116087	0.779611	0.420262
6	0.298472	8464.371956	0.749822	0.435259
7	0.261529	8285.461542	0.782178	0.410796
8	0.336392	9112.340022	0.742828	0.416179
9	0.439366	9036.100963	0.751683	0.410458
10	0.698227	8877.702445	0.787594	0.419639
11	0.519608	8657.914225	0.810525	0.409562
12	0.628989	8717.032698	0.847609	0.381739
13	0.827932	8522.513386	0.867111	0.350654
14	0.705091	8311.442049	0.886660	0.374102
15	0.812976	8237.994102	0.916468	0.356977
16	0.591599	8062.688817	0.915495	0.338569
17	0.675214	7974.891617	0.887920	0.336678
18	0.669967	7816.506119	0.912933	0.343768
19	0.713114	7795.066855	0.875314	0.344380

Tabla 8: Variación en parámetros para K-Means (caso 2)

Observamos cómo el comportamiento en este caso es más regular que en el caso anterior. Además, vemos cómo el índice Calinski-Harabasz va disminuyendo conforme aumenta el número de clusters, así como el coeficiente Silhouette. Por su parte, el índice Davies-Bouldin va aumentando. Este es un comportamiento más estable y esperable: conforme aumentamos el número de clusters, tendremos clusters con menos instancias pero más densos (por ello disminuye el índice Calinski-Harabasz), obteniendo a su vez clusters más similares (lo cual explica el incremento del índice Davies-Bouldin), y teniendo un mayor solapamiento (lo cual lo ratifica el decremento del coeficiente Silhouette). Además, con estos resultados vemos cómo la mejor elección del número de clusters está en torno a los 5 clusters: con menos no obtenemos una separación suficiente y con más obtenemos clusters demasiado similares. Por tanto, nuestra elección inicial de 5 clusters fue medianamente acertada para este caso – en efecto, pudimos distinguir grupos claros con ciertas características.



(a) Calinski-Harabasz **(b)** Davies-Bouldin **(c)** Silhouette

Figura 14: Medidas ante variación en número de clusters para K-Means (caso 2)

2.2.2. Estudio de Ward

En el caso de este algoritmo, nos interesa mucho más no qué clusters se han generado, sino cómo lo han hecho. Esto es porque estos algoritmos generan una jerarquía, y a partir de ella podemos especificar un cierto número de clusters para obtener un agrupamiento. Para estudiar este proceso usaremos un dendrograma, que podemos ver en la Figura 15.

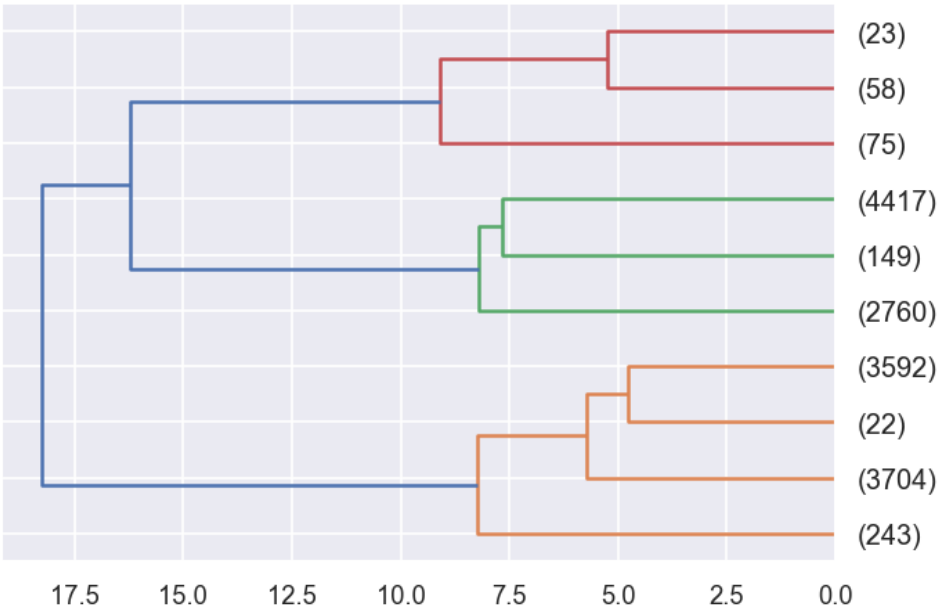


Figura 15: Dendrograma en Ward, truncado a 10 hojas (caso 2)

El dendrograma representado se interpreta como sigue: los colores simbolizan los diferentes niveles, y los valores en el eje de abscisas nos indican la distancia a la que se encuentran los enlaces. En el eje de ordenadas tenemos el número de elementos del subcluster correspondiente.

Viendo dicho diagrama desde arriba, vemos que la primera hoja tiene 23 elementos, y como está a una distancia alta, no se junta todavía con la siguiente hoja. Los primeros agrupamientos ocurren entre las hojas con 3592 elementos y 22 elementos, cuando están a una distancia cercana a 5. Las uniones de los grupos inferiores son más cercanas que las de los superiores. A partir del dendrograma, podemos ver cómo, si quisiésemos obtener 5 clusters, obtendríamos justamente la distribución que aparece en la Figura 16.

Para comprender con más detalle cómo se realiza el agrupamiento, combinaremos el dendrograma con un mapa de calor en la Figura 17.

En primer lugar, vemos cómo las divisiones de `impuesto_renta` separa las instancias en dos grandes grupos, de tamaño similar. A continuación, uno de los grupos se sigue subdividiendo en `transferencias_abonadas`, mientras que el otro lo hace en `transferencias_percibidas`. Esto tiene sentido con lo interpretado antes: las personas con mayor renta (y por tanto mayores impuestos sobre la renta) tienen más transferencias abonadas y menos transferencias percibidas. Finalmente, vemos cómo la variable `renta_disponible` no realiza subdivisiones significativas. Esto se debe a que esta variable está muy correlacionada con `impuesto_renta`, que ya ha establecido las divisiones al principio.

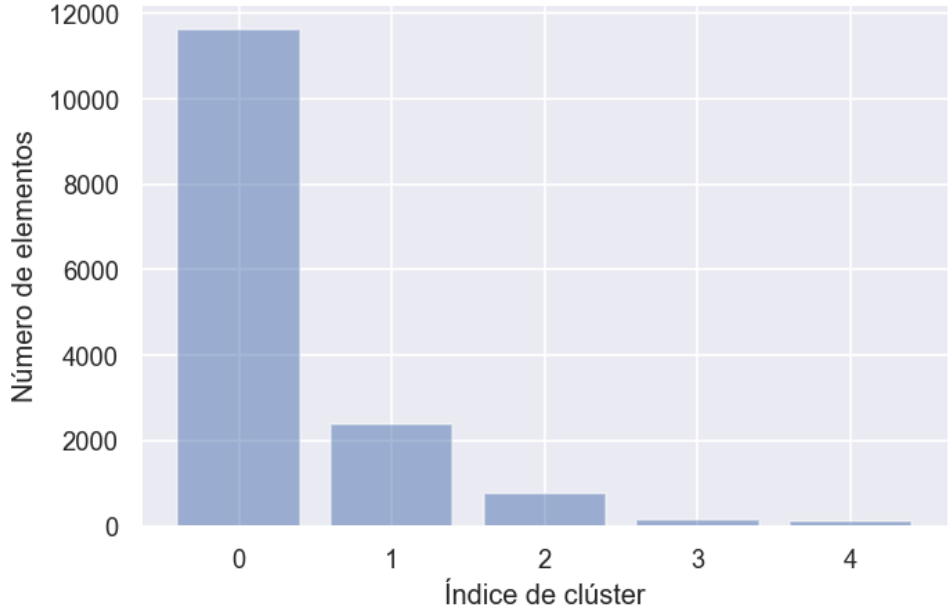


Figura 16: Número de instancias clasificadas en cada cluster para Ward (caso 2)

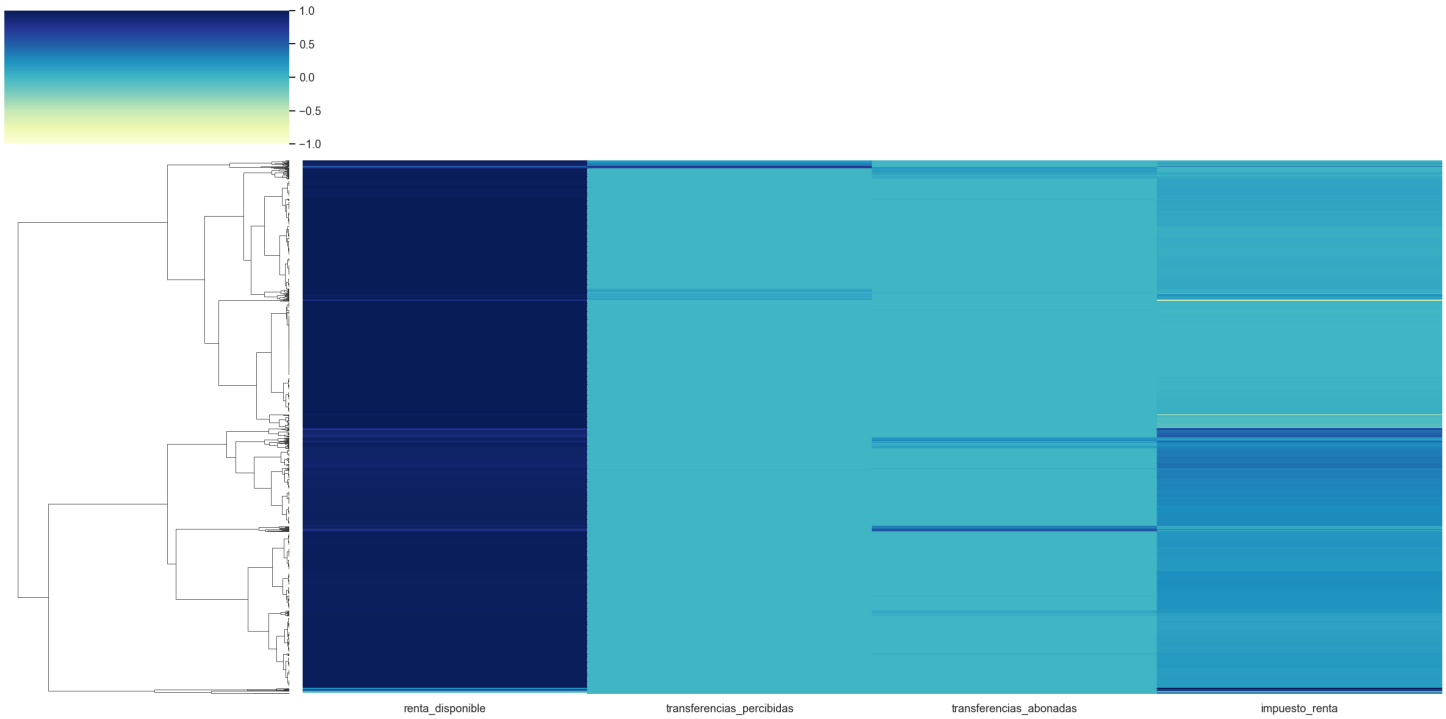


Figura 17: Dendrograma con mapa de calor en Ward (caso 2)

2.2.3. Interpretación de la segmentación

Mediante los agrupamientos de estos apartados hemos observado las siguientes tendencias:

- Los impuestos pagados y el valor de la renta están directamente correlados.
- A más impuestos pagados, se abonan más transferencias, mientras que a menos, los encuestados tienen más transferencias percibidas que abonadas.

2.3. Alquiler de segundas viviendas y poder adquisitivo

Finalmente veremos la relación entre los beneficios del alquiler de viviendas y el poder adquisitivo de los encuestados (que analizaremos mediante su renta y los ingresos mínimos para llegar a fin de mes). Usaremos las siguientes variables:

- **`renta_disponible`** (`HY020`): renta disponible total del hogar en el año anterior al de encuesta. Esta variable es esencial pues nos permite apreciar la capacidad monetaria de las familias.
- **`renta_alquiler`** (`HY040N`): renta neta procedente del alquiler de una propiedad o terreno en el año anterior al de encuesta.
- **`valor_vivienda`** (`HV010`): valor actual de la vivienda principal.
- **`ingresos_minimos`** (`HS130`): ingresos mínimos para llegar a final de mes.

Realizaremos un cierto filtrado y preprocesado de estos datos. En primer lugar, sólo usaremos datos de los encuestados con segunda vivienda (basta ver que el valor `HV020` sea igual a `1`). Tras esto, vemos cómo hay muchos valores perdidos, en concreto en `valor_vivienda` y en `ingresos_minimos`. Para disponer de datos reales, y dado que la muestra tiene suficientes instancias, he dedicido prescindir de las instancias con valores perdidos. Obtenemos tras esto un total de 3383 instancias.

Usaremos los algoritmos como sigue:

```
<ClusterAlgorithm [K-Means], 1 instance:
    init=k-means++, n_clusters=5, n_init=5, random_state=common.RANDOM_SEED>
<ClusterAlgorithm [Birch], 1 instance:
    branching_factor=25, threshold=0.1, n_clusters=5>
<ClusterAlgorithm [DBSCAN], 1 instance:
    eps=0.01, min_samples=15>
<ClusterAlgorithm [MeanShift], 1 instance:
    (no parameters)>
<ClusterAlgorithm [Ward], 1 instance:
    n_clusters=5, linkage=ward>
```

Tras ejecutar todas las instancias, obtenemos las medidas de la Tabla 9.

	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette	Número de clusters
K-Means	0.060921	1085.753364	1.059004	0.347075	5
Birch	0.092774	290.038885	0.537357	0.867256	5
DBSCAN	0.051473	133.877231	2.149241	-0.134353	5
MeanShift	16.567714	145.487809	0.560435	0.312649	38
Ward	0.440466	905.613209	1.056496	0.386116	5

Tabla 9: Medidas para caso de estudio 3

De nuevo, obtenemos resultados similares a los de las Tablas 1 y 6. Nuevamente, Birch es el algoritmo que tiene mejores resultados, y MeanShift el que peor. Además, vemos cómo

DBSCAN tiene un coeficiente de Silhouette negativo, a pesar de haber realizado un clustering más significativo (ha logrado identificar tres clusters más de lo que ha hecho en apartados anteriores): lo analizaremos en detenimiento a continuación.

He decidido prescindir del estudio de K-Means para este caso de estudio, dado que ya se estudió de forma exhaustiva en los casos anteriores.

2.3.1. Estudio de DBSCAN

Comenzamos analizando los resultados obtenidos: intentaremos explicar por qué hemos obtenido un coeficiente de Silhouette negativo. En primer lugar, vemos en la Figura **??** cómo, esencialmente, este algoritmo está creando un cluster principal, el 0, y de forma secundaria está creando clusters con muy pocos elementos – sin embargo, tenemos muchos *outliers*, que son los elementos del cluster -1. De hecho, es sorprendente ver cómo hay más elementos en el cluster -1 que en el 0.

Figura 18: Número de instancias clasificadas en cada cluster para DBSCAN (caso 3)

Para entender cómo ha resultado el agrupamiento, nos ayudaremos de las Figura 19 y 20, así como de la Tabla 10. Podemos ver cómo el agrupamiento realizado es muy poco preciso, no marcándose grupos diferenciados en niguno de los casos. Además, los boxplots nos revelan que se ha realizado cierta segmentación, en concreto usando principalmente el valor de `renta_disponible`. Podemos ver cómo, en general, a mayor renta, el valor de la vivienda es mayor, así como los ingresos. El cluster 4, por su parte, no nos da ninguna información relevante.

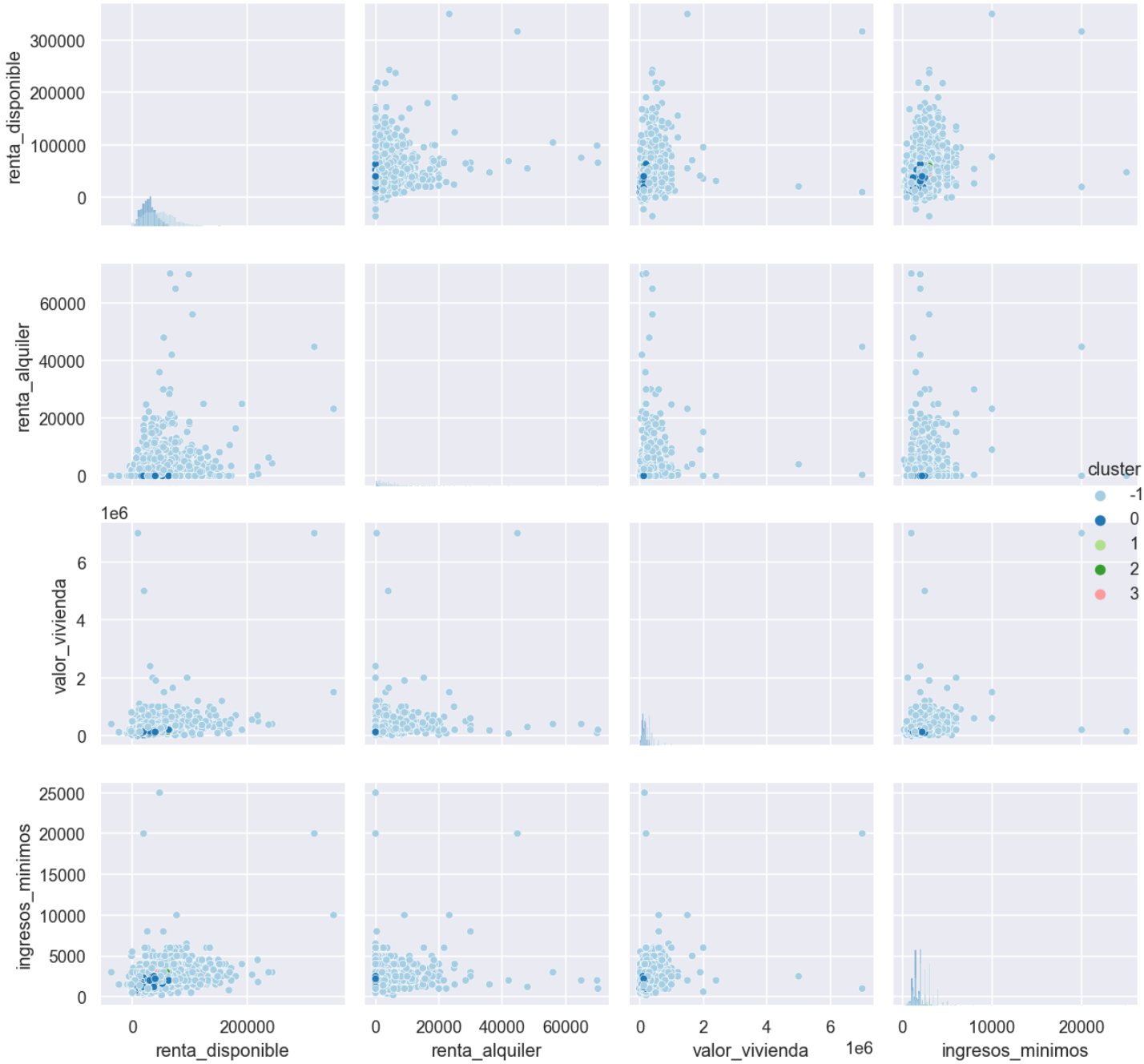


Figura 19: Matriz de dispersión para DBSCAN (caso 3)

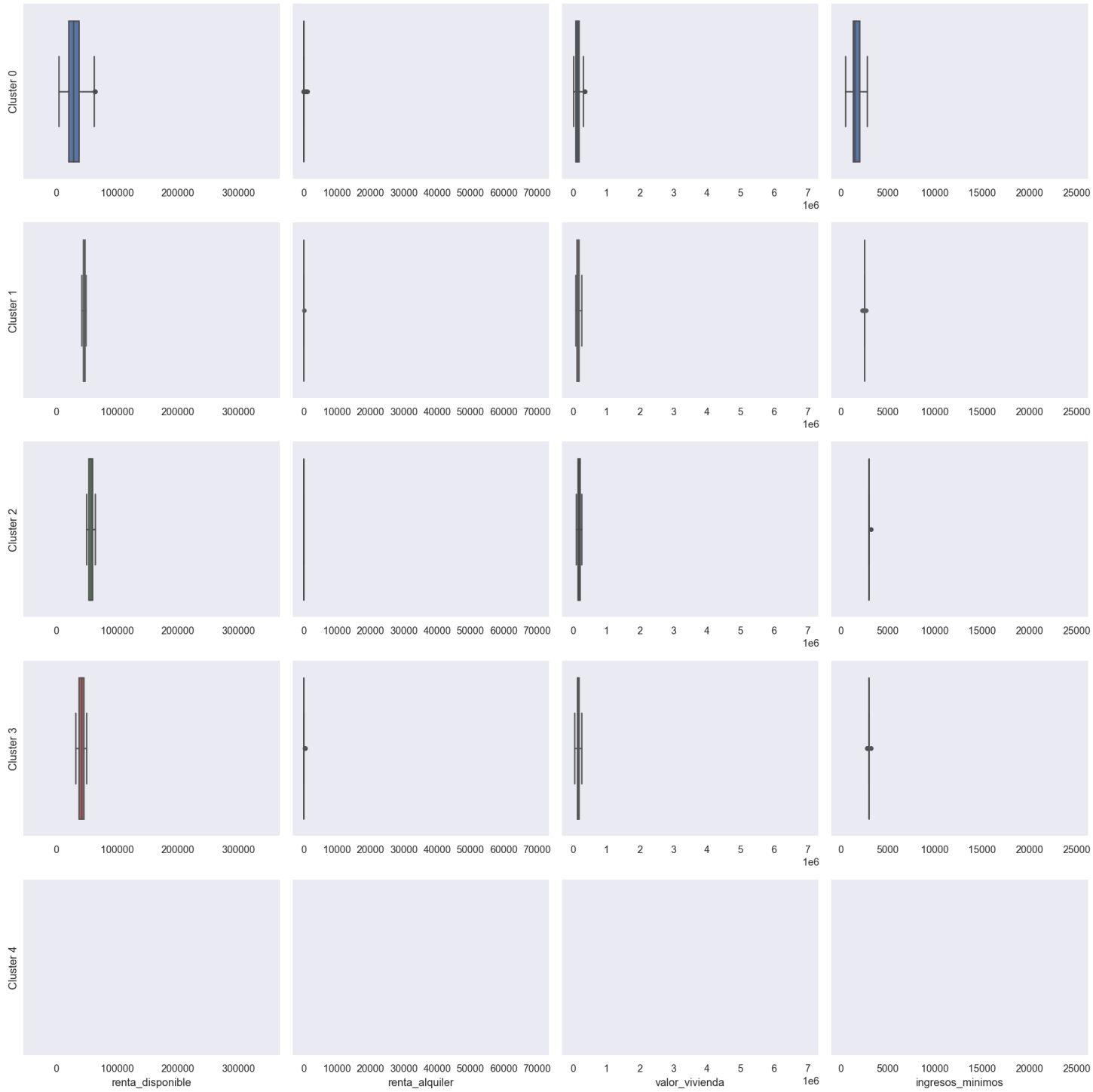


Figura 20: Boxplots para BSCAN (caso 3)

Cluster	renta_disponible	renta_alquiler	valor_vivienda	ingresos_minimos
-1	28743.35-65921.25	0.00-3496.00	150000.00-350000.00	1500.00-3000.00
0	19817.95-36794.05	0.00-0.00	80000.00-180000.00	1290.00-2000.00
1	44362.70-46499.35	0.00-0.00	115000.00-181500.00	2500.00-2500.00
2	53186.38-59742.97	0.00-0.00	150000.00-200000.00	3000.00-3000.00
3	36885.55-45019.35	0.00-0.00	120000.00-182500.00	3000.00-3000.00

Tabla 10: Características de los clusters para DBSCAN (caso 3)

Esto nos indica que los parámetros que usamos no eran convenientes, para ello realizaremos un estudio modificando los valores de `eps` (ϵ) y `min_samples` (mínimo de samples). Vemos los resultados de las ejecuciones de todas las instancias en la Tabla 11.

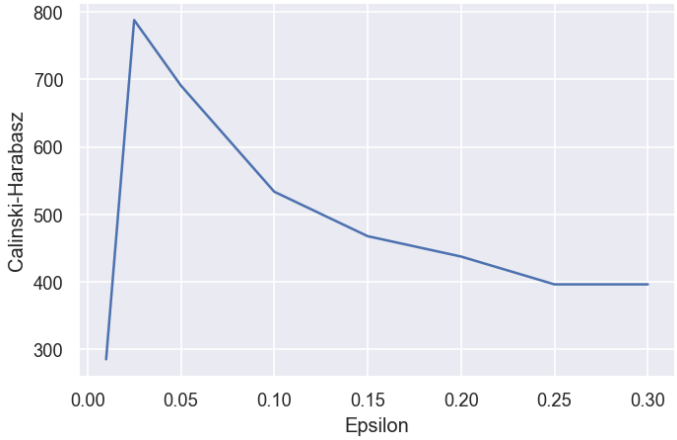
eps	min_samples	Tiempo	Calinski-Harabasz	Davies-Bouldin	Silhouette	Número de clusters
0.010	5	0.057119	37.714815	1.929662	-0.441902	17
0.010	10	0.044803	284.761368	2.276998	0.027153	3
0.010	15	0.040622	133.877231	2.149241	-0.134353	5
0.010	20	0.042163	98.963152	1.984486	-0.185535	6
0.025	5	0.084432	370.283688	1.620323	0.398102	3
0.025	10	0.083957	788.209068	1.562690	0.517122	2
0.025	15	0.087013	791.534490	1.573986	0.486548	2
0.025	20	0.081952	786.720871	1.596716	0.461422	2
0.050	5	0.160941	232.880642	1.838346	0.600787	4
0.050	10	0.168896	691.101831	1.387485	0.703619	2
0.050	15	0.194511	720.525070	1.380781	0.689387	2
0.050	20	0.166561	751.383484	1.370616	0.678435	2
0.100	5	0.247112	533.481751	1.301251	0.822024	2
0.100	10	0.252783	533.481751	1.301251	0.822024	2
0.100	15	0.260234	553.473158	1.296794	0.816505	2
0.100	20	0.249785	562.825613	1.281658	0.814130	2
0.150	5	0.281548	443.829590	1.240508	0.867902	2
0.150	10	0.275363	467.480924	1.219174	0.865624	2
0.150	15	0.273916	501.096629	1.236474	0.854727	2
0.150	20	0.273862	501.096629	1.236474	0.854727	2
0.200	5	0.256259	395.986363	1.275806	0.886030	2
0.200	10	0.260829	437.255279	1.238237	0.878732	2
0.200	15	0.258081	437.255279	1.238237	0.878732	2
0.200	20	0.251237	437.255279	1.238237	0.878732	2
0.250	5	0.250530	287.596912	1.122572	0.887685	2
0.250	10	0.247636	395.986363	1.275806	0.886030	2
0.250	15	0.253368	423.004065	1.249631	0.883125	2
0.250	20	0.262025	423.004065	1.249631	0.883125	2
0.300	5	0.235831	287.596912	1.122572	0.887685	2
0.300	10	0.244994	395.986363	1.275806	0.886030	2
0.300	15	0.241191	395.986363	1.275806	0.886030	2
0.300	20	0.241849	395.986363	1.275806	0.886030	2

Tabla 11: Variación en parámetros para DBSCAN (caso 3)

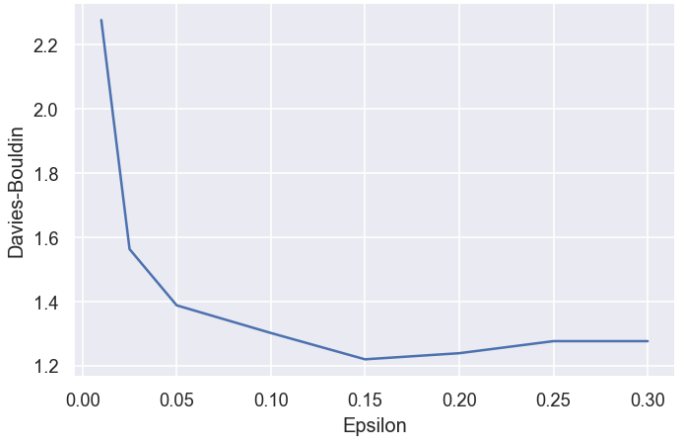
En primer lugar, vemos cómo con el valor =0.01 obtenemos un coeficiente de Silhouette negativo, es decir, no obtenemos una agrupación correcta. Esto puede deberse a que, cuando la distancia que determina si una instancia es densamente alcanzable a partir de otra es muy pequeña, se pueden alcanzar menos instancias a partir de una dada, lo cual aumenta el número de clusters y la clasificación incorrecta.

En general, se acaban obteniendo dos grupos: un único cluster y otro de *outliers*. Esto nos indica que DBSCAN quizás no sea el mejor algoritmo para estos datos, dado que no hay combinaciones de parámetros que obtengan buenos resultados.

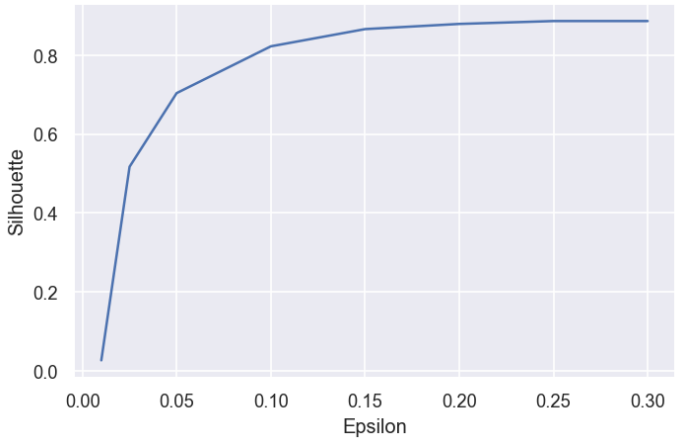
Independientemente de estos malos resultados, analizaremos el comportamiento de las métricas al variar estos parámetros. Primero, en la Figura 21 fijamos `min_samples` y variamos `eps`. Vemos cómo el coeficiente Silhouette crece con `eps`, mientras que el índice Calinski-Harabasz y Davies-Bouldin van decreciendo. Por otra parte, vemos cómo el número de clusters siempre se reduce a 2, es decir, sólo obtenemos un cluster y otro de *outliers*.



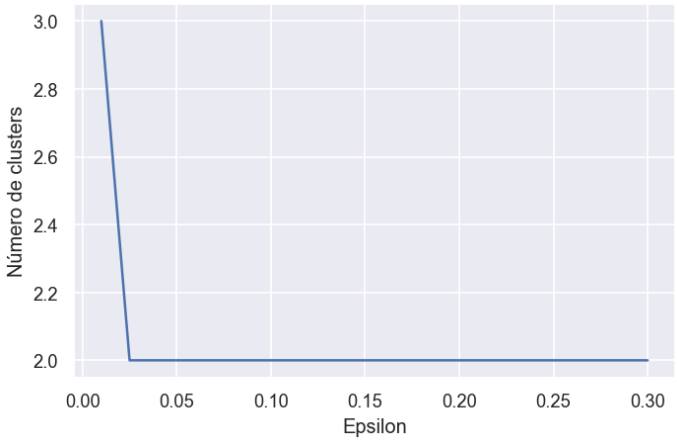
(a) Calinski-Harabasz



(b) Davies-Bouldin



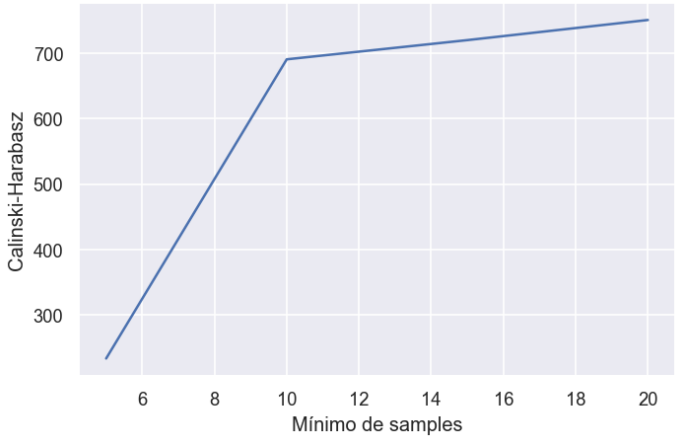
(c) Silhouette



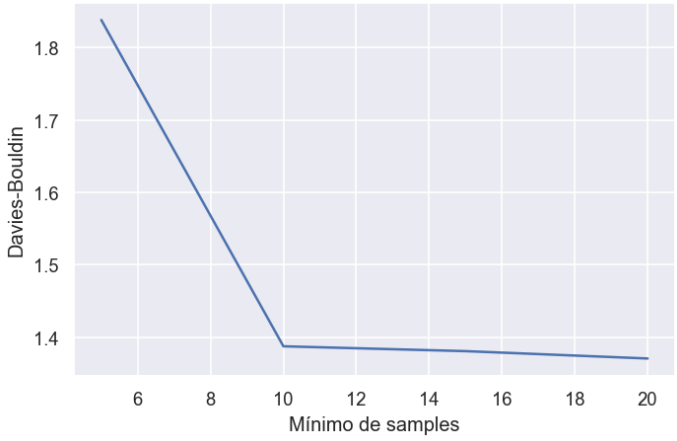
(d) Número de clusters

Figura 21: Medidas ante variación en `eps` para DBSCAN, fijado `min_samples`=10 (caso 3)

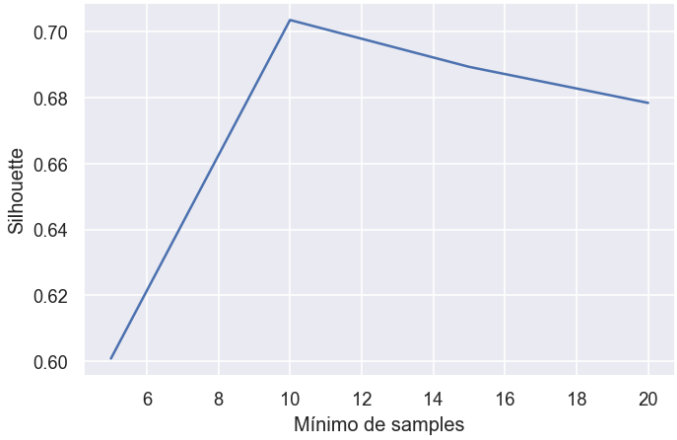
Fijado `eps` y variando `min_samples` tampoco obtenemos resultados muy reveladores (Figura 22): vemos cómo en `min_samples` se alcanza un punto de inflexión en el que varían las derivadas de las distintas medidas, y en donde se maximiza el coeficiente de Silhouette, también con 2 clusters (incluyendo el de *outliers*).



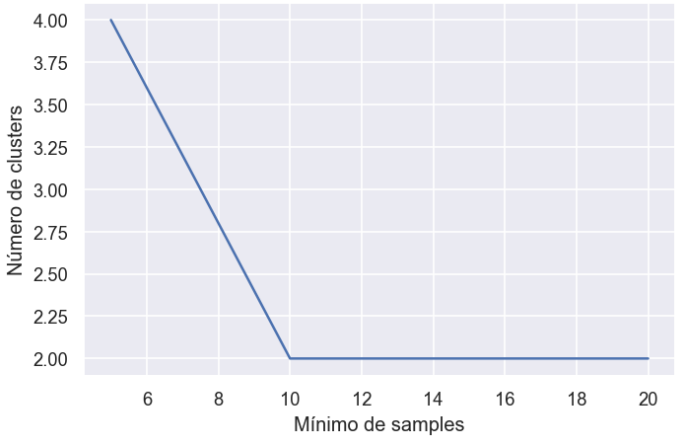
(a) Calinski-Harabasz



(b) Davies-Bouldin



(c) Silhouette



(d) Número de clusters

Figura 22: Medidas ante variación en `min_samples` para DBSCAN, fijado `eps`=.05 (caso 3)

2.3.2. Interpretación de la segmentación

Mediante este análisis, hemos podido ver cómo las agrupaciones que se obtienen siguen las siguientes tendencias:

- Los encuestados con mayor renta tienen unos mayores ingresos mínimos para llegar a final de mes.
- Los encuestados con mayor renta no necesariamente tienen una mayor renta de alquiler.
- Por lo general, a mayor renta, mayor valor de la vivienda principal.

3. Bibliografía

scikit-learn, *Clustering*. https://scikit-learn.org/stable/modules/clustering.html

Towards Data Science, *DBSCAN Python Example: The Optimal Value For Epsilon (EPS)*. https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc

Haitian Wei, *How to measure clustering performances when there are no ground truth?*. https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c

scikit-learn, *sklearn.cluster.estimate-bandwith*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.estimate_bandwidth.html