

# Práctica 2

# Análisis Predictivo

# Mediante Clasificación

Paula Villanueva Núñez

49314567Z

[pvillanunez@correo.ugr.es](mailto:pvillanunez@correo.ugr.es)

Cuarto Curso del Grado en Ingeniería Informática Curso 2021-2022 Grupo 1

Universidad de Granada

# Índice

<b>1</b>	<b>Introducción</b>	<b>3</b>
<b>2</b>	<b>Casos de estudio</b>	<b>5</b>
2.1	Caso de estudio 1 . . . . .	5
2.1.1	k-Means . . . . .	6
2.1.2	Birch . . . . .	8
2.1.3	Interpretación de la segmentación . . . . .	17
2.2	Caso de estudio 2 . . . . .	18
2.2.1	k-Means . . . . .	19
2.2.2	AgglomerativeClustering . . . . .	21
2.2.3	Interpretación de la segmentación . . . . .	33
2.3	Caso de estudio 3 . . . . .	34
2.3.1	k-Means . . . . .	35
2.3.2	Agglomerative Clustering . . . . .	45
2.3.3	Interpretación de la segmentación . . . . .	46
<b>3</b>	<b>Contenido adicional</b>	<b>48</b>
3.1	Boxplot . . . . .	48
3.2	Dendrograma (sin y con mapa de calor) . . . . .	49
<b>4</b>	<b>Bibliografía</b>	<b>50</b>

# 1 Introducción

En esta práctica abordaremos un problema real de clasificación mediante el uso de técnicas de aprendizaje no supervisado para realizar un análisis relacional mediante segmentación. Sobre el conjunto de datos con el que trabajaremos, se aplicarán distintos algoritmos de agrupamiento o *clustering*.

Para ello, partiremos de los microdatos publicados por el Instituto Nacional de Estadística (INE) sobre la última encuesta de condiciones de vida (año 2020), se dispone de un conjunto de 15.043 respuestas y unas 200 variables sobre datos básicos del hogar, la vivienda, exclusión social, renta, carencia, consumo... Estas variables se estructuran se la siguiente forma:

- **Datos básicos del hogar:** contiene información del hogar, para localizarlo geográficamente, el grado de urbanización de la zona en la que se encuentra, el año de la encuesta y los factores de ponderación.
- **Datos básicos de la persona:** se encuentra información para identificar a la persona y a su hogar, factores de ponderación, información demográfica, situación de presente o ausente, situación respecto a la actividad, etc.
- **Datos detallados del hogar:** se pueden encontrar datos básicos del hogar, sobre la vivienda, acerca de la exclusión social, acerca de la renta, sobre carencia material, etc.
- **Datos detallados de los adultos:** contiene información básica de la persona, de educación, salud, vida laboral, sobre la renta, etc.

Para aplicar un análisis de agrupamiento, realizaremos tres casos de estudio diferentes en los que seleccionaremos unas variables numéricas u ordinales para *clustering* y otras variables categóricas, que serán útiles para fijar los casos de estudio donde centrar el análisis. Los casos de estudio que se han elegido son de interés, esto es, se han fijado condiciones en algunas variables. De esta forma, aplicaremos distintos algoritmos de *clustering*, analizaremos la calidad de las soluciones obtenidas e interpretaremos los resultados con el objetivo de encontrar relaciones entre las variables.

Este trabajo se ha realizado empleando bibliotecas y paquetes de Python, principalmente `numpy`, `pandas`, `scikit-learn`, `matplotlib` y `seaborn`.

Los algoritmos de agrupamiento elegidos para realizar este estudio son los siguientes.

- **k-Means.** Es un algoritmo iterativo en el que las instancias se van moviendo entre los clusters hasta que se alcanza el conjunto de clusters deseado. Esto es, agrupa los ejemplos en tantos grupos como se le indique. Cabe destacar que este algoritmo genera clusters convexo.  
Los parámetros de este algoritmo son `init='k-means++'`, `n_clusters=5`, `n_init=5`, `random_state=123456`.
- **MeanShift.** Este algoritmo fija un radio (*bandwidth*), en vez del número de clusters, y va desplazando los centroides hasta las regiones más densas. Los clusters generados también serán convexas.  
Los parámetros de este algoritmo son `bandwidth=0.3`
- **Birch.** Es un algoritmo de clustering incremental. Agrupa los objetos conforme se van recibiendo y mantiene las características de los clusters para resumir los objetos que van llegando.

Cada vez que llega un objeto, desciende por el árbol escogiéndose en cada nodo el CF más cercano. Cuando llega a una hoja, si puede ser absorbido por algún CF existente se agrega y, si no, se crea un CF nuevo. Si se alcanza el máximo, se divide la hoja en dos con el par de CF más lejanos de la hoja anterior.

Los parámetros de este algoritmo son `branching_factor=25, threshold=0.25, n_clusters =5`.

- **AgglomerativeClustering.** Es un método jerárquico, es decir, genera una jerarquía en la que, dependiendo del nivel de corte, obtendremos un clustering distinto. Es un método aglomerativo que se basa en medir la distancia entre clusters y en cada paso se fusionan los dos clusters más cercanos. También se conoce como *Ward*.

Los parámetros de este algoritmo son `n_clusters=5, linkage="ward"`.

- **DBSCAN.** A partir de un punto, va buscando otros puntos en su vecindad y uniéndolos al cluster hasta que no se alcancen más puntos. Puede encontrar clusters con distintas formas. Utiliza dos parámetros para medir la densidad: *eps* (radio) y *minPts* (tamaño mínimo). Además, este algoritmo cuando encuentra objetos que no puede agrupar, genera un cluster  $-1$  y los agrupa en él.

Los parámetros de este algoritmo son `eps=0.14, min_samples=5`.

Para comparar estos algoritmos, en cada uno de ellos se ha obtenido las siguientes medidas.

- **Tiempo** de ejecución del algoritmo.
- **Coeficiente Silhouette.** mide cómo de similares son los objetos de un mismo cluster (cohesión) comparado con otros clusters (separación). De esta forma, mide la calidad global del agrupamiento. Esta medida toma valores en el intervalo  $[-1, 1]$ , siendo mejor cuanto más cercano a 1 está. Si se acerca a 0 significa que el objeto está en el borde de dos clusters y si se acerca a  $-1$ , indica que el agrupamiento es incorrecto.
- **Índice Calinski-Harabasz.** Indica la razón entre la dispersión intra-clusters y la dispersión inter-clusters. Cuanto mayor, mejor es el agrupamiento.

Además, en al menos 2 algoritmos para cada caso de estudio se analizará el efecto de algunos parámetros determinantes.

Este análisis realizado también se apoyará en visualizaciones tales como nubes de puntos (*scatter matrix*), dendogramas (en agrupamiento jerárquico), mapas de temperatura (*heatmap*), gráficos de burbujas con la distancia relativa entre los centros de los clústers mediante *multidimensional scaling*, etc.

## 2 Casos de estudio

### 2.1 Caso de estudio 1

El primer caso de estudio se basará en las personas que tienen una renta neta entre 0 y 150.000. Las variables que nos ayudarán a llevar a cabo el agrupamiento son las siguientes.

- **HY020.** Renta disponible total del hogar en el año anterior al de la encuesta. Toma valores entre -99999999.99 y 99999999.99
- **HY140G.** Impuesto sobre la renta y cotizaciones sociales. Toma valores entre -99999999.99 y 99999999.99
- **HH070.** Gastos de la vivienda: alquiler (si la vivienda se encuentra en régimen de alquiler), intereses de la hipoteca (para viviendas en propiedad con pagos pendientes) y otros gastos asociados (comunidad, agua, electricidad, gas, etc.). Toma valores entre 1 y 99999999.99
- **HH031.** Año del contrato o de la compra o de instalación. Toma valores entre 1900 y 2020.
- **HC010.** Durante el mes pasado, importe que el hogar gastó en alimentos y bebidas no alcohólicas para ser consumidas en casa. Toma valores entre 0 y 99999999.99

Los resultados obtenidos al ejecutar los cinco algoritmos antes mencionados con este caso de estudio y estas variables se recogen en la siguiente tabla.

Tabla 1: Resultados caso de estudio 1

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
k-Means	0.686	4858.284	0.25998	5
MeanShift	157.964	513.046	0.23616	4
Birch	0.310	2859.992	0.19544	5
Agglomerative Clustering	5.014	3672.636	0.19569	5
DBSCAN	0.545	217.979	0.29528	3

Con respecto al **tiempo de ejecución**, destaca el de MeanShift por ser notablemente superior al del resto, esto se debe a que su complejidad tiende a  $O(T * n * \log(n))$ , con  $n$  el número de muestras y  $T$  el número de puntos. En cuanto a los otros algoritmos, el algoritmo AgglomerativeClustering tarda 5 segundos, mientras que el resto apenas llega a 1 segundo.

El índice **Calinski-Harabasz** de los algoritmos k-Means, Agglomerative Clustering y Birch es bastante alto, a diferencia del de los algoritmos MeanShift y DBSCAN. Esto puede deberse a que estos dos últimos algoritmos no agrupan bien los objetos en los clusters, pues analizando los clusters que generan, siempre hay un cluster con la mayoría de los objetos y otro cluster con tamaño muy pequeño con los objetos que no puede agrupar, por lo que no se realiza un buen agrupamiento.

El índice **Silhouette** es bastante similar en todos los algoritmos. Los algoritmos MeanShift y DBSCAN proporcionan un índice Silhouette más elevado, pues por lo ya comentado antes, tenemos que los objetos del cluster con tamaño más grande estén muy separados del otro cluster.

Con respecto al **número de clusters**, los algoritmos k-Means, Birch y AgglomerativeClustering generan 5 clusters pues así se lo indicamos en los parámetros. Los algoritmos MeanShift y DBSCAN

generan menos número de clusters, pero realmente un cluster tiene la mayoría de los elementos y el resto apenas tienen elementos.

En resumen, para este caso los algoritmos k-Means y Birch, que son los que estudiaremos más adelante, son los que mejor resultados obtienen, teniendo en cuenta el tiempo de ejecución y los índices Silhouette y Calinski-Harabasz.

### 2.1.1 k-Means

**Análisis de parámetros** En este apartado se analizará el parámetro decisivo de k-Means, el número de clusters. Por defecto, se ha ejecutado con 5 clusters. En este apartado variaremos este número de tal forma que estará comprendido entre 2 y 20, pues no nos interesa que sea demasiado elevado pues no obtendríamos apenas información. De esta forma, en la siguiente tabla se muestran los resultados para cada valor de este parámetro.

Tabla 2: Análisis de parámetros con k-Means del caso de estudio 1

Número de clusters	Tiempo (s)	Calinski-Harabasz	Silhouette
2	0.296	4195.746	0.21572
3	0.459	5168.548	0.26319
4	0.770	5066.365	0.26425
5	0.877	4858.284	0.25998
6	1.108	4513.355	0.23725
7	1.265	4157.473	0.21781
8	1.969	3904.830	0.20875
9	3.183	3687.288	0.20835
10	1.769	3523.596	0.20926
11	2.339	3408.080	0.19433
12	2.500	3275.173	0.19162
13	2.640	3145.769	0.19356
14	4.485	3043.729	0.19069
15	7.501	2955.974	0.18403
16	5.609	2880.827	0.18398
17	4.676	2797.239	0.18273
18	4.375	2728.150	0.18234
19	5.353	2662.171	0.18043
20	5.259	2601.605	0.17745

Destaca que al aumentar el número de clusters, aumenta el tiempo de ejecución considerablemente. Esto se debe a que si aumenta el número de clusters, aumenta el número de operaciones a realizar. Con respecto a los índices Silhouette y Calinski-Harabasz, en las siguientes figuras se muestra su gráfica para cada número de clusters.

## 2 Casos de estudio

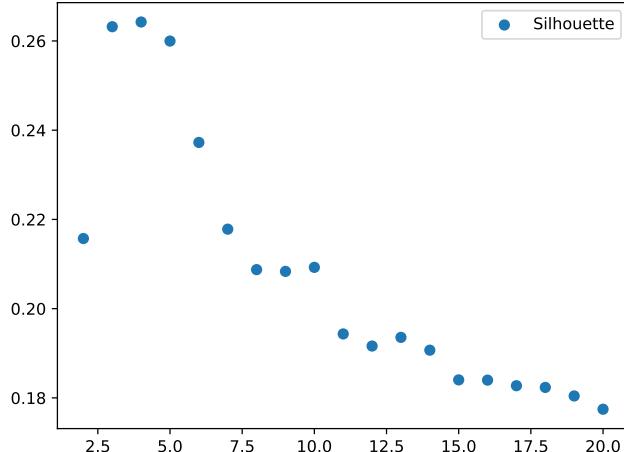


Figura 1: Silhouette de k-Means en el caso 1

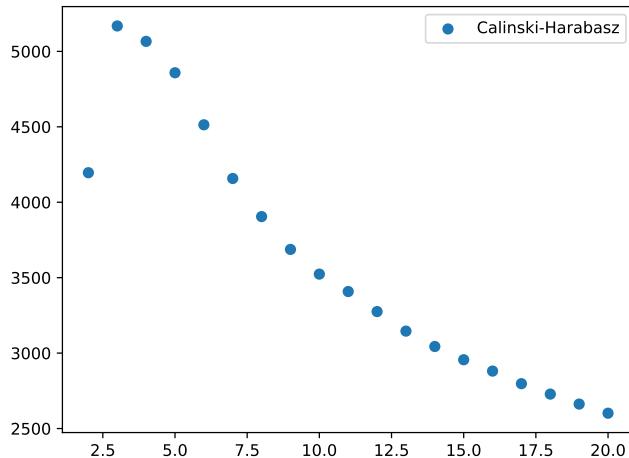


Figura 2: Calinski-Harabasz de k-Means en el caso 1

Observamos que, para un número pequeño de clusters, obtenemos mejores índices Silhouette y Calinski-Harabasz. Esto se debe a que, al haber pocos clusters posibles, los objetos se quedan más cerca de su centro. Además, observamos que el decremento en cada medida es distinta. Esto puede deberse a que, al aumentar el número de clusters, haya menos dispersión intra-clusters por lo que el índice Calinski-Harabasz disminuye. En cuanto al índice Silhouette, al aumentar el número de clusters, no habrá tanta diferencia entre los clusters por lo que disminuye con menos rapidez.

En conclusión, para este caso de estudio con este algoritmo, deberíamos establecer el número de clusters a 3 o 4, que es cuando obtenemos un mejores valores de ambas medidas.

### 2.1.2 Birch

Este algoritmo genera 5 clusters, cuya organización se puede observar en la siguiente figura.

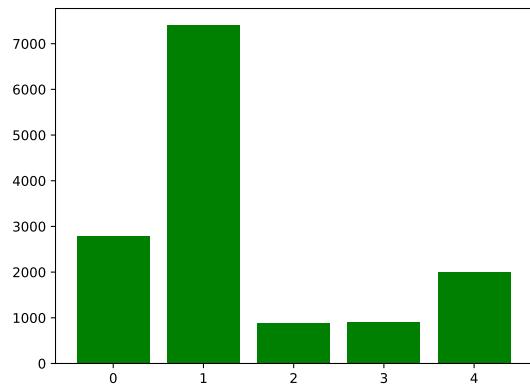


Figura 3: Tamaño de los clusters de Birch en el caso 1

Observamos que el cluster número 1 tiene el mayor número de objetos (más de la mitad, el 53%) y el cluster número 2, el que menos (un 6.33%). Los clusters 0, 3 y 4 tienen, más o menos, un número más similar.

En la siguiente figura podemos observar el Heatmap de este algoritmo. Esta gráfica puede ayudarnos a determinar las características de cada cluster. Esto es, cuanto más alto sea el valor, mayor determinación habrá.

## 2 Casos de estudio



Figura 4: Heatmap de Birch en el caso 1

El valor que tiene cada cluster en cada variable indica el centroide que tiene. Esto es, el cluster 1 tiene el centroide en 20124.503 con respecto a la variable `renta_neta`. Además, el color indica que cuanto más azul sea, más determinante será esa variable a la hora de agrupar los objetos. Por ejemplo, el año de compra de los objetos del cluster 2 o del cluster 4 es determinante, o los gastos en alimentación del cluster 5.

En la siguiente figura se muestra la matriz de dispersión de las variables dos a dos y en la diagonal, el histograma de cada variable.

## 2 Casos de estudio

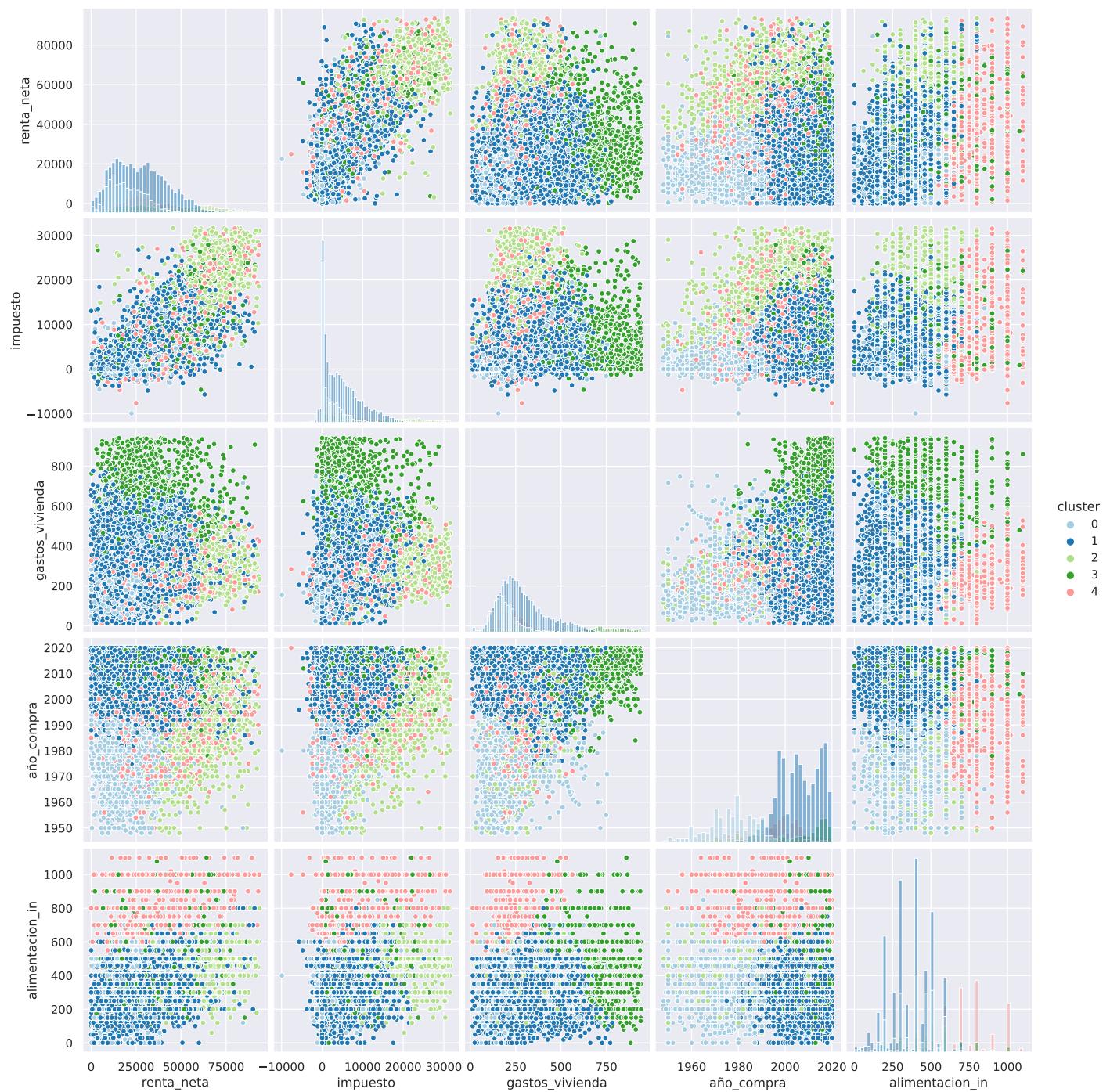


Figura 5: Scatter matrix de Birch en el caso 1

Observamos que la variable `año_compra` ayuda a separar los objetos en cuatro de los clusters. Las variables `gastos_vivienda`, `alimentacion_in` e `impuesto` también aportan información relevante para agrupar los objetos en al menos 4 clusters.

La siguiente figura muestra los diagramas de cajas de cada variable. Este diagrama será útil también para distinguir las características de cada grupo, pues nos da información del rango de valores que toman las variables.

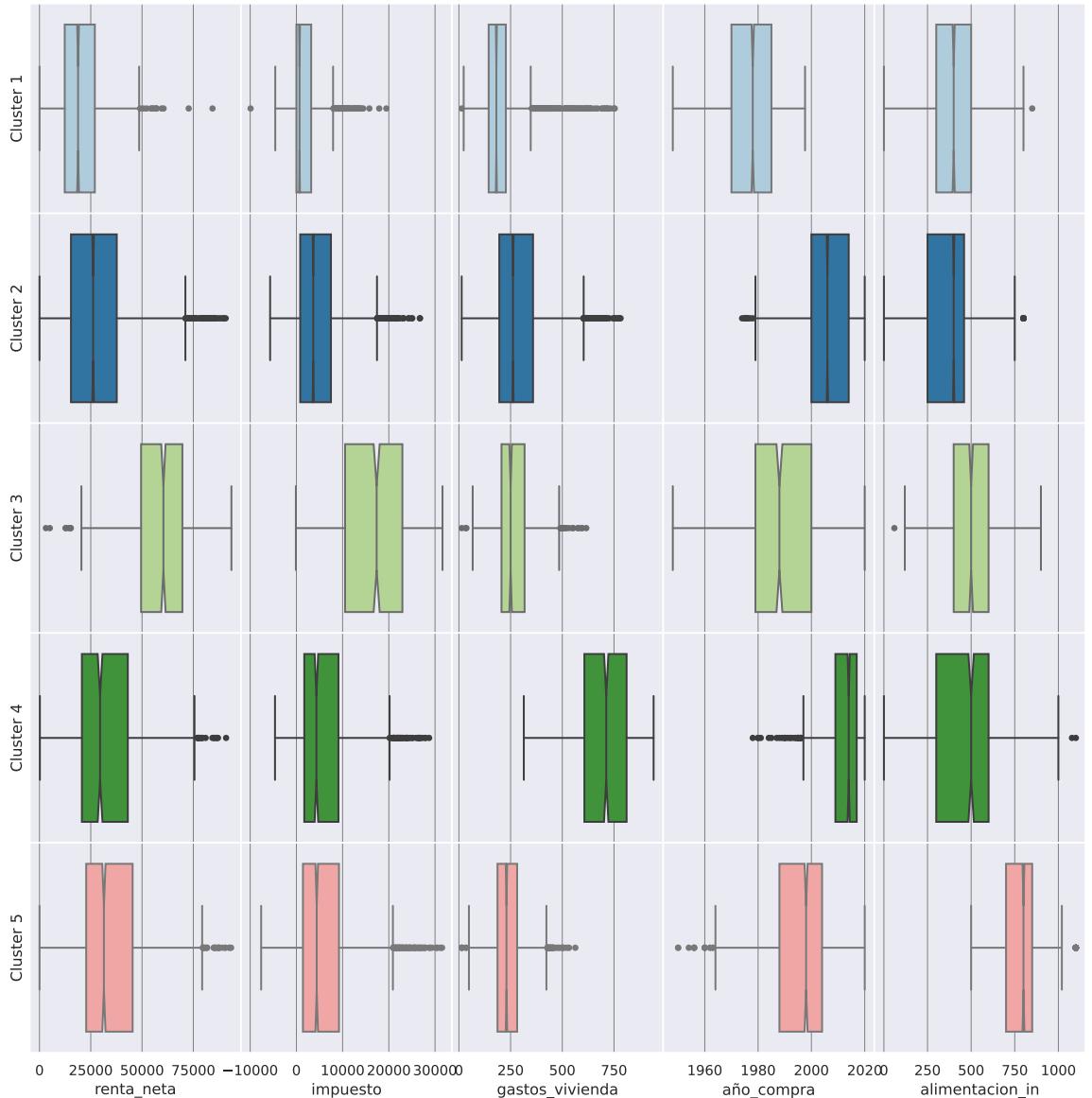


Figura 6: Boxplot de Birch en el caso 1

En el cluster 1 tenemos que la **renta\_neta** toma valores entre 13.000 y 26.000, el **impuesto** toma valores entre 0 y 3.000, los **gatos\_vivienda** toma valores entre 140 y 240, el **año\_compra** toma valores entre 1970 y 1985 y la **alimentacion\_in** toma valores entre 260 y 500. Este cluster tiene la renta neta más baja, paga pocos impuestos, tiene pocos gastos de vivienda, compró la vivienda hace unos 40 años y también gasta poco en alimentación. Por lo que este cluster contiene a la gente que tiene poco dinero y gasta muy poco, puede que en este cluster la mayoría de la gente sean personas mayores sobre todo por el año de compra, o también podrá haber gente que haya heredado la casa de sus padres y al tener poco dinero, no habrá podido permitirse comprar una casa más nueva.

## 2 Casos de estudio

En el cluster 2 tenemos que la **renta\_neta** toma valores entre 16.000 y 37.500, el **impuesto** toma valores entre 100 y 8.000, los **gatos\_vivienda** toma valores entre 200 y 400, el **año\_compra** toma valores entre 2000 y 2013 y la **alimentacion\_in** toma valores entre 250 y 450. Este cluster tiene la renta algo más elevada que el anterior, paga algo más de impuestos y gastos de vivienda, los gastos en alimentación también son bajos pero, con diferencia, el año de compra es mucho más reciente.

En el cluster 3 tenemos que la **renta\_neta** toma valores entre 49.000 y 70.000, el **impuesto** toma valores entre 10.500 y 22.500, los **gatos\_vivienda** toma valores entre 220 y 300, el **año\_compra** toma valores entre 1979 y 2000 y la **alimentacion\_in** toma valores entre 400 y 600. Este cluster destaca por tener la renta más elevada que el resto, y en consecuencia paga más impuestos. Los gastos de la vivienda son similares a los de los anteriores clusters, aunque los gastos en alimentación son algo más elevados.

En el cluster 4 tenemos que la **renta\_neta** toma valores entre 20.000 y 40.500, el **impuesto** toma valores entre 200 y 9.000, los **gatos\_vivienda** toma valores entre 630 y 800, el **año\_compra** toma valores entre 2010 y 2017 y la **alimentacion\_in** toma valores entre 270 y 600. En este cluster destaca que los gastos de la vivienda son más elevados que el resto, esto puede deberse a que el año de compra es mucho más reciente y en consecuencia, los precios asociados a la vivienda son más elevados que antes. El resto de variables son bastante similares a las ya comentadas.

En el cluster 5 tenemos que la **renta\_neta** toma valores entre 22.000 y 45.000, el **impuesto** toma valores entre 200 y 9.500, los **gatos\_vivienda** toma valores entre 150 y 270, el **año\_compra** toma valores entre 1987 y 2003 y la **alimentacion\_in** toma valores entre 700 y 900. Este cluster destaca porque los gastos de alimentación son más elevados que el resto, también tienen una renta neta algo elevada y el resto de variables son similares a las ya comentadas. Que haya tantos gastos en alimentación puede deberse a que tengan hijos o personas dependientes a las que tengan que alimentar.

En la siguiente imagen se representan los distintos clusters y cómo de cerca o lejos están unos de otros y el tamaño de cada uno.

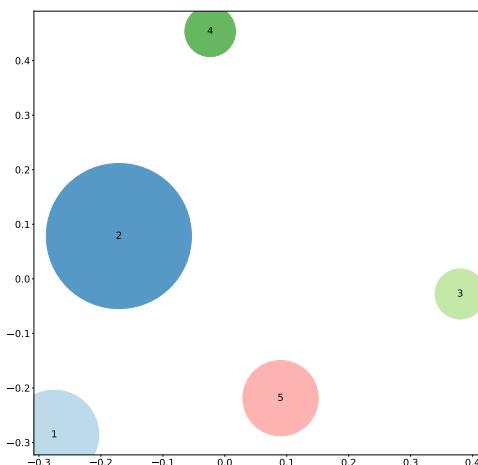


Figura 7: Multidimensional scaling de Birch en el caso 1

Y en la siguiente imagen, se puede ver la distancia intercluster.

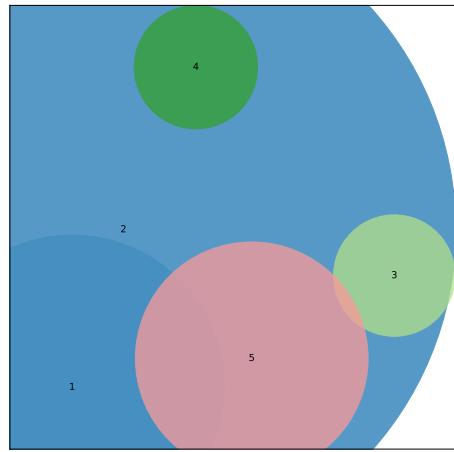


Figura 8: Distancia intercluster de Birch en el caso 1

A continuación, se muestra la distribución por variable y cluster.

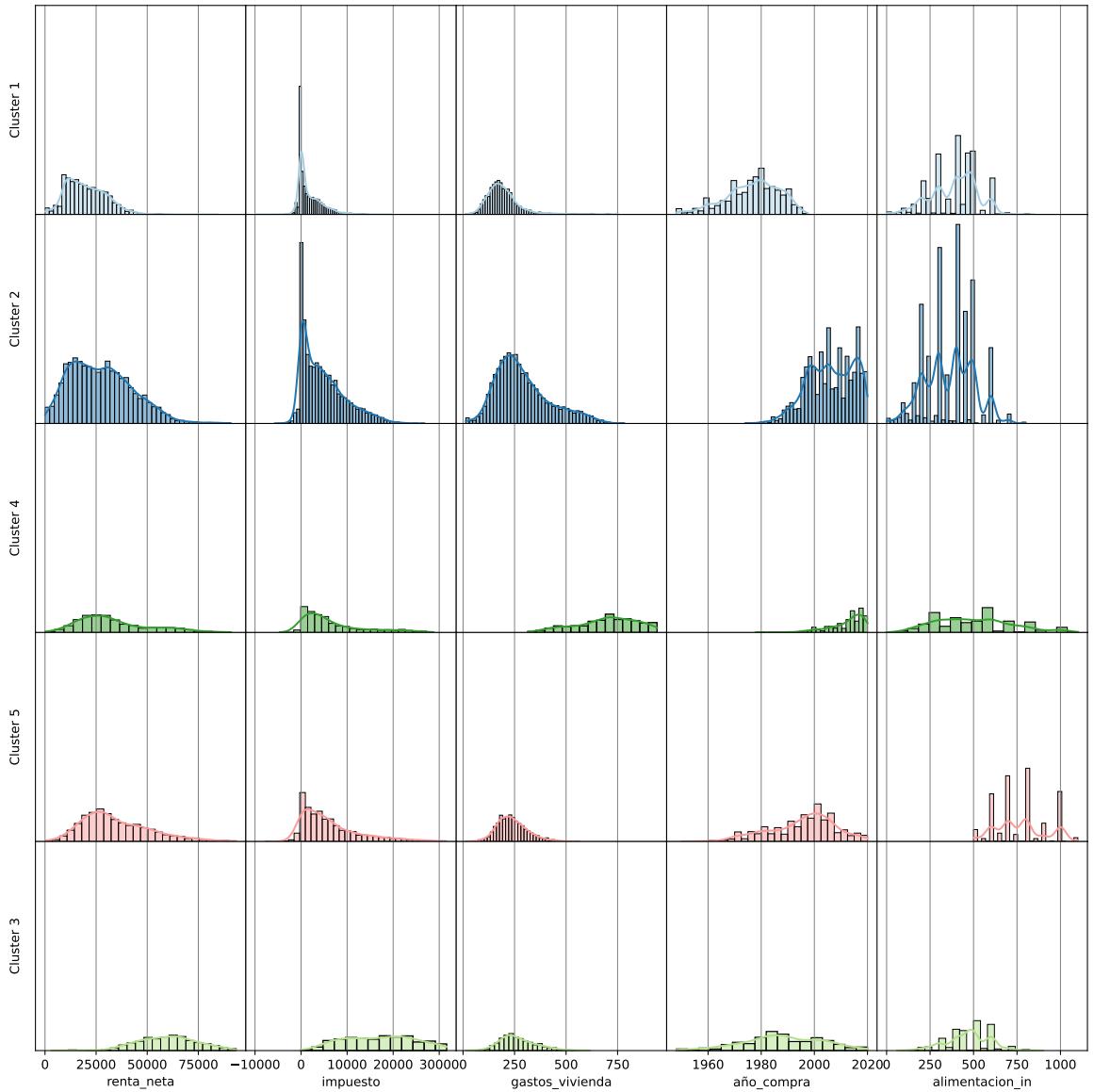


Figura 9: Distribución por variable y cluster de Birch en el caso 1

Ya sabíamos los rangos de valores que podían tomar las variables, pues ya lo comentamos en la gráfica del Boxplot. Sin embargo, con esta gráfica obtenemos más información sobre qué elementos toman valores mayomente.

En el cluster 1 observamos que las personas suelen tener una renta muy baja y los impuestos que pagan son mayormente 0.

En el cluster 2 destaca que los impuestos que pagan son mayormente 0, aunque el resto de variables toman valores con frecuencia similar.

En el resto de clusters y variables, cabe destacar que casi todos siguen una normal. No hay valores que destaque por ser más frecuentes que los ya comentados.

## 2 Casos de estudio

A continuación, se muestra una gráfica con las coordenadas paralelas para visualizar y analizar los clusters.

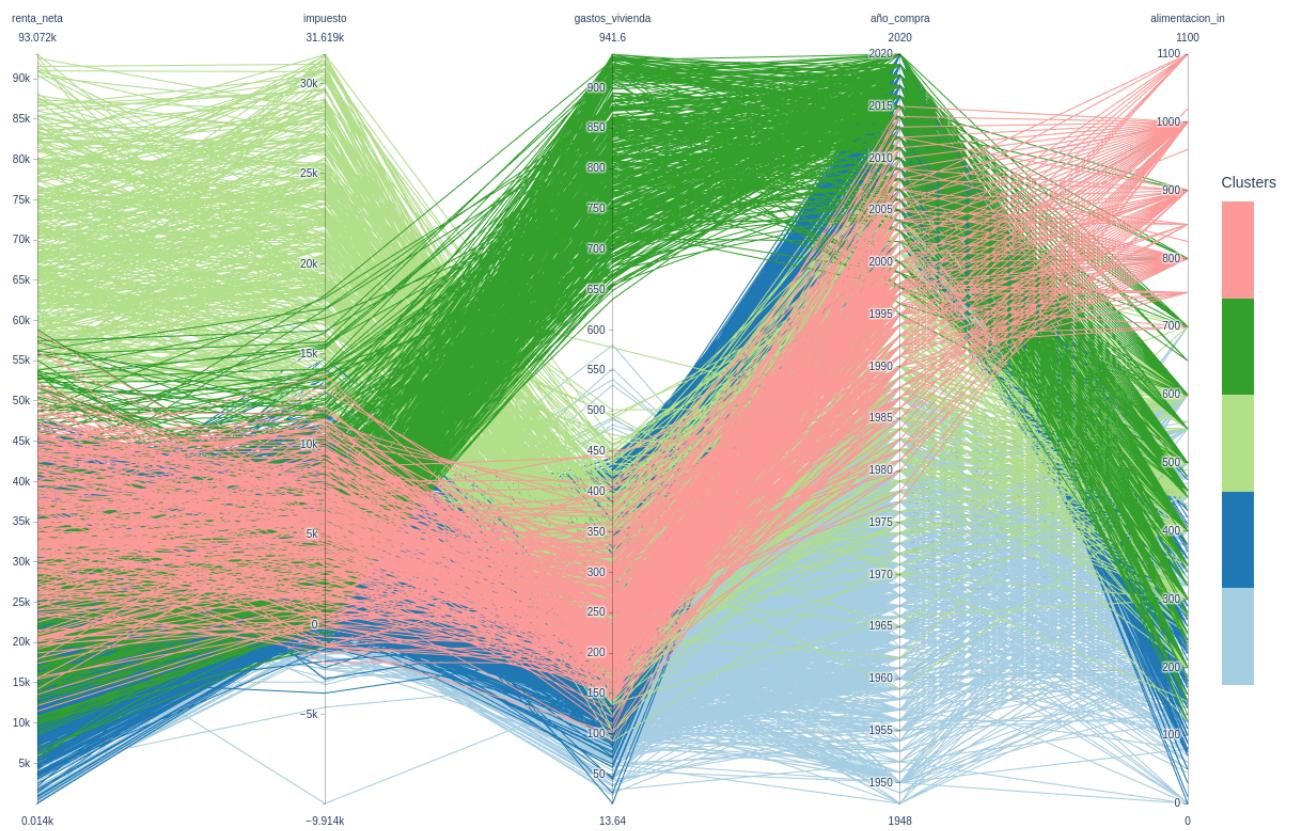


Figura 10: Parallel coordinates de Birch en el caso 1

**Análisis de parámetros** En este apartado se analizarán los parámetros `threshold` y `branching_factor` de Birch. Por defecto, se ha ejecutado con 5 clusters, `branching_factor=25` y `threshold=0.25`. En este apartado variaremos los valores de `branching_factor` entre 10 y 40 y los valores de `threshold` entre 0.1 y 0.4. De esta forma, en la siguiente tabla se muestran los resultados para cada valor de este parámetro.

## 2 Casos de estudio

Tabla 3: Análisis de parámetros de Birch del caso de estudio 2

Threshold	Branching factor	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
0.10	10	1.027	3498.555	0.19713	5
0.15	10	0.605	3519.153	0.20392	5
0.20	10	0.534	2616.076	0.17188	5
0.25	10	0.398	1774.264	0.14333	5
0.30	10	0.351	3197.041	0.21825	5
0.35	10	0.204	3921.442	0.22725	5
0.40	10	0.210	3441.325	0.18812	5
0.10	15	0.877	3077.791	0.17767	5
0.15	15	0.599	2599.854	0.15824	5
0.20	15	0.384	2879.127	0.16305	5
0.25	15	0.378	2652.612	0.19058	5
0.30	15	0.218	3157.312	0.19457	5
0.35	15	0.210	3921.442	0.22725	5
0.40	15	0.212	3441.325	0.18812	5
0.10	20	0.918	3417.259	0.21092	5
0.15	20	0.541	2840.413	0.18644	5
0.20	20	0.383	2391.108	0.18724	5
0.25	20	0.350	2501.323	0.18757	5
0.30	20	0.212	2192.199	0.18340	5
0.35	20	0.210	3921.442	0.22725	5
0.40	20	0.233	3441.325	0.18812	5
0.10	25	0.976	2829.884	0.13944	5
0.15	25	0.548	2925.992	0.16127	5
0.20	25	0.458	2787.955	0.17580	5
0.25	25	0.371	2859.992	0.19544	5
0.30	25	0.238	2192.199	0.18340	5
0.35	25	0.230	3921.442	0.22725	5
0.40	25	0.242	3441.325	0.18812	5
0.10	30	0.881	3244.722	0.17953	5
0.15	30	0.540	2751.411	0.21589	5
0.20	30	0.404	2798.647	0.15722	5
0.25	30	0.286	2604.142	0.16791	5
0.30	30	0.221	2192.199	0.18340	5
0.35	30	0.215	3921.442	0.22725	5
0.40	30	0.223	3441.325	0.18812	5
0.10	35	0.910	3648.834	0.19483	5
0.15	35	0.688	3653.365	0.21741	5
0.20	35	0.509	3340.550	0.20469	5
0.25	35	0.311	1273.308	0.18069	5
0.30	35	0.240	2192.199	0.18340	5
0.35	35	0.247	3921.442	0.22725	5
0.40	35	0.242	3441.325	0.18812	5

Tabla 4: Análisis de parámetros de Birch del caso de estudio 2 (continuación)

Threshold	Branching factor	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
0.10	40	0.848	3699.565	0.17864	5
0.15	40	0.611	3339.548	0.20292	5
0.20	40	0.455	2896.375	0.19118	5
0.25	40	0.281	1933.432	0.11453	5
0.30	40	0.247	2192.199	0.18340	5
0.35	40	0.251	3921.442	0.22725	5
0.40	40	0.244	3441.325	0.18812	5

A diferencia del algoritmo k-Means, al aumentar el Threshold (umbral), con branching factor (factor de ramificación) fijo, el tiempo de ejecución disminuye. Esto se debe a que también aumenta la probabilidad de que un objeto pertenezca a un cluster y así disminuye el número de nodos a recorrer del árbol. Con respecto a los índices Calinski-Harabasz y Silhouette, al aumentar el umbral y/o el factor de ramificación fijo, no siguen una tendencia creciente o decreciente. Por lo que es difícil determinar los mejores parámetros para este algoritmo.

### 2.1.3 Interpretación de la segmentación

A partir de los resultados obtenidos, concluimos que la variable `año_compra` determina los distintos clusters. Pues las personas que hace más de unos 40 años (aproximadamente) que no compran una vivienda, tienen poco dinero y pocos gastos (vivienda, alimentación, etc.), pues no pueden permitirse apenas nada más. Las personas que tienen más renta, pudieron permitirse comprar una vivienda en los últimos años y por lo tanto también tienen más gastos (vivienda, alimentación, etc.). Incluso a mayor renta, más impuestos pagan.

## 2.2 Caso de estudio 2

El segundo caso de estudio se basará en las personas que reciben asistencia social. Las variables que nos ayudarán a llevar a cabo el agrupamiento son las siguientes.

- **HY060N.** Ingresos por asistencia social en el año anterior de encuesta. Toma valores entre 1 y 99999999.99
- **HY020.** Renta disponible total del hogar en el año anterior al de la encuesta. Toma valores entre -99999999.99 y 99999999.99
- **HC010.** Durante el mes pasado, importe que el hogar gastó en alimentos y bebidas no alcohólicas para ser consumidas en casa. Toma valores entre 0 y 99999999.99
- **HS130.** Ingresos mínimos para llegar a final de mes. Toma valores entre 1 y 999999.

Los resultados obtenidos al ejecutar los cinco algoritmos antes mencionados con este caso de estudio y estas variables se recogen en la siguiente tabla.

Tabla 5: Resultados caso de estudio 2

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
k-Means	0.037	244.750	0.23638	5
MeanShift	3.384	67.954	0.26489	6
Birch	0.015	157.065	0.21638	5
Agglomerative Clustering	0.011	201.209	0.21281	5
DBSCAN	0.007	46.426	0.19531	3

Con respecto al **tiempo de ejecución**, destaca el de MeanShift por ser notablemente superior al del resto, esto se debe a que su complejidad tiende a  $O(T * n * \log(n))$ , con  $n$  el número de muestras y  $T$  el número de puntos. En cuanto a los otros algoritmos, apenas llega cada uno a 1 segundo de tiempo de ejecución.

El índice **Calinski-Harabasz** de los algoritmos k-Means y Agglomerative Clustering es bastante alto, a diferencia del de los algoritmos MeanShift y DBSCAN. Esto puede deberse a que estos dos últimos algoritmos no agrupan bien los objetos en los clusters, pues analizando los clusters que generan, siempre hay un cluster con la mayoría de los objetos y otros clusters con tamaño muy pequeño con los objetos que no puede agrupar, por lo que no se realiza un buen agrupamiento.

El índice **Silhouette** es bastante similar en todos los algoritmos. El algoritmo MeanShift proporciona un índice Silhouette más elevado, pues por lo ya comentado antes, tenemos que los objetos del cluster con tamaño más grande estén muy separados del otro cluster.

Con respecto al **número de clusters**, los algoritmos k-Means, Birch y AgglomerativeClustering generan 5 clusters pues así se lo indicamos en los parámetros. Los algoritmos MeanShift y DBSCAN generan 6 y 3 clusters, respectivamente, pero realmente un cluster tiene la mayoría de los elementos y el resto apenas tienen elementos.

En resumen, para este caso los algoritmos k-Means y Agglomerative Clustering, que son los que estudiaremos más adelante, son los que mejor resultados obtienen, teniendo en cuenta el tiempo de ejecución y los índices Silhouette y Calinski-Harabasz.

### 2.2.1 k-Means

**Análisis de parámetros** En este apartado se analizará el parámetro decisivo de k-Means, el número de clusters. Por defecto, se ha ejecutado con 5 clusters. En este apartado variaremos este número de tal forma que estará comprendido entre 2 y 20, pues no nos interesa que sea demasiado elevado pues no obtendríamos apenas información. De esta forma, en la siguiente tabla se muestran los resultados para cada valor de este parámetro.

Tabla 6: Resultados caso de estudio 2

Número de clusters	Tiempo (s)	Calinski-Harabasz	Silhouette
2	0.037	417.813	0.35573
3	0.110	304.499	0.25105
4	0.139	267.716	0.25530
5	0.611	244.750	0.23638
6	0.667	227.409	0.21755
7	0.704	216.004	0.22078
8	0.922	206.435	0.21141
9	0.966	200.035	0.21435
10	0.150	191.960	0.21235
11	0.167	182.802	0.20545
12	0.179	180.168	0.21030
13	0.210	173.847	0.21118
14	0.210	170.042	0.21633
15	0.225	165.850	0.20872
16	0.225	162.312	0.21374
17	0.253	156.911	0.21439
18	0.252	155.326	0.21556
19	0.322	150.728	0.21314
20	0.302	149.536	0.21255

Destaca que al aumentar el número de clusters, aumenta el tiempo de ejecución considerablemente. Esto se debe a que si aumenta el número de clusters, aumenta el número de operaciones a realizar. Con respecto a los índices Silhouette y Calinski-Harabasz, en las siguientes figuras se muestra su gráfica para cada número de clusters.

## 2 Casos de estudio

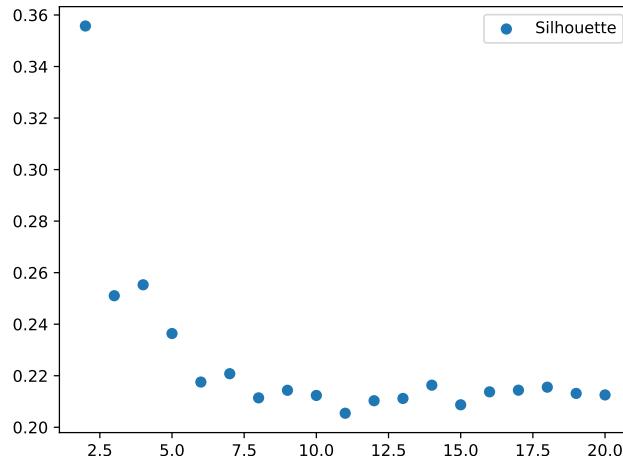


Figura 11: Silhouette de k-Means en el caso 2

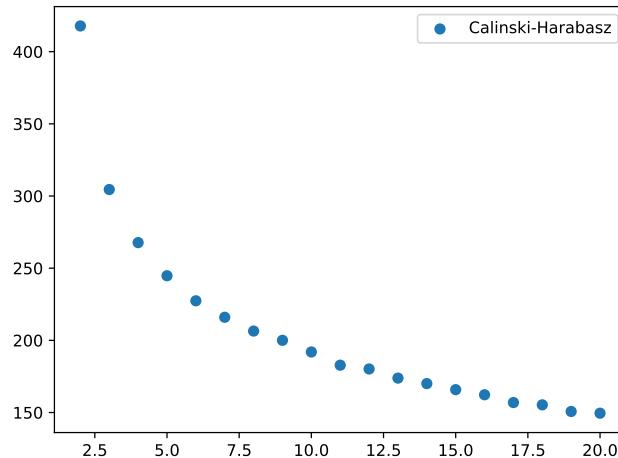


Figura 12: Calinski-Harabasz de k-Means en el caso 2

Observamos que, para un número pequeño de clusters, obtenemos mejores índices Silhouette y Calinski-Harabasz. Esto se debe a que, al haber pocos clusters posibles, los objetos se quedan más cerca de su centro. Además, observamos que el decremento en cada medida es distinta. Esto puede deberse a que, al aumentar el número de clusters, haya menos dispersión intra-clusters por lo que el índice Calinski-Harabasz disminuye. En cuanto al índice Silhouette, al aumentar el número de clusters, no habrá tanta diferencia entre los clusters por lo que disminuye con menos rapidez.

En conclusión, para este caso de estudio con este algoritmo, deberíamos establecer el número de clusters a 3 o 4, que es cuando obtenemos un mejores valores de ambas medidas.

### 2.2.2 Agglomerative Clustering

Este algoritmo genera 5 clusters, cuya organización se puede observar en la siguiente figura.

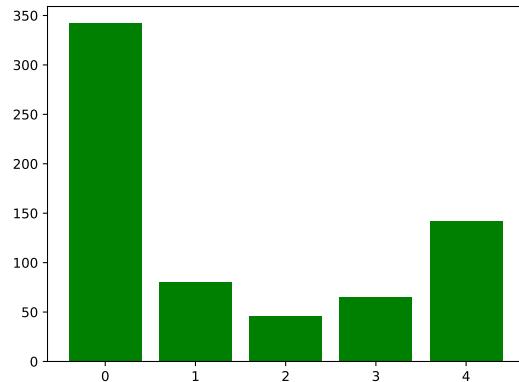


Figura 13: Tamaño de los clusters de Agglomerative Clustering en el caso 2

Observamos que el cluster número 0 tiene el mayor número de objetos (la mitad, el 50%) y el cluster número 2, el que menos (un 6.81%). Los clusters 1, 3 y 4 tienen, más o menos, un número más similar.

En la siguiente figura podemos observar el Heatmap de este algoritmo. Esta gráfica puede ayudarnos a determinar las características de cada cluster. Esto es, cuanto más alto sea el valor, mayor determinación habrá.

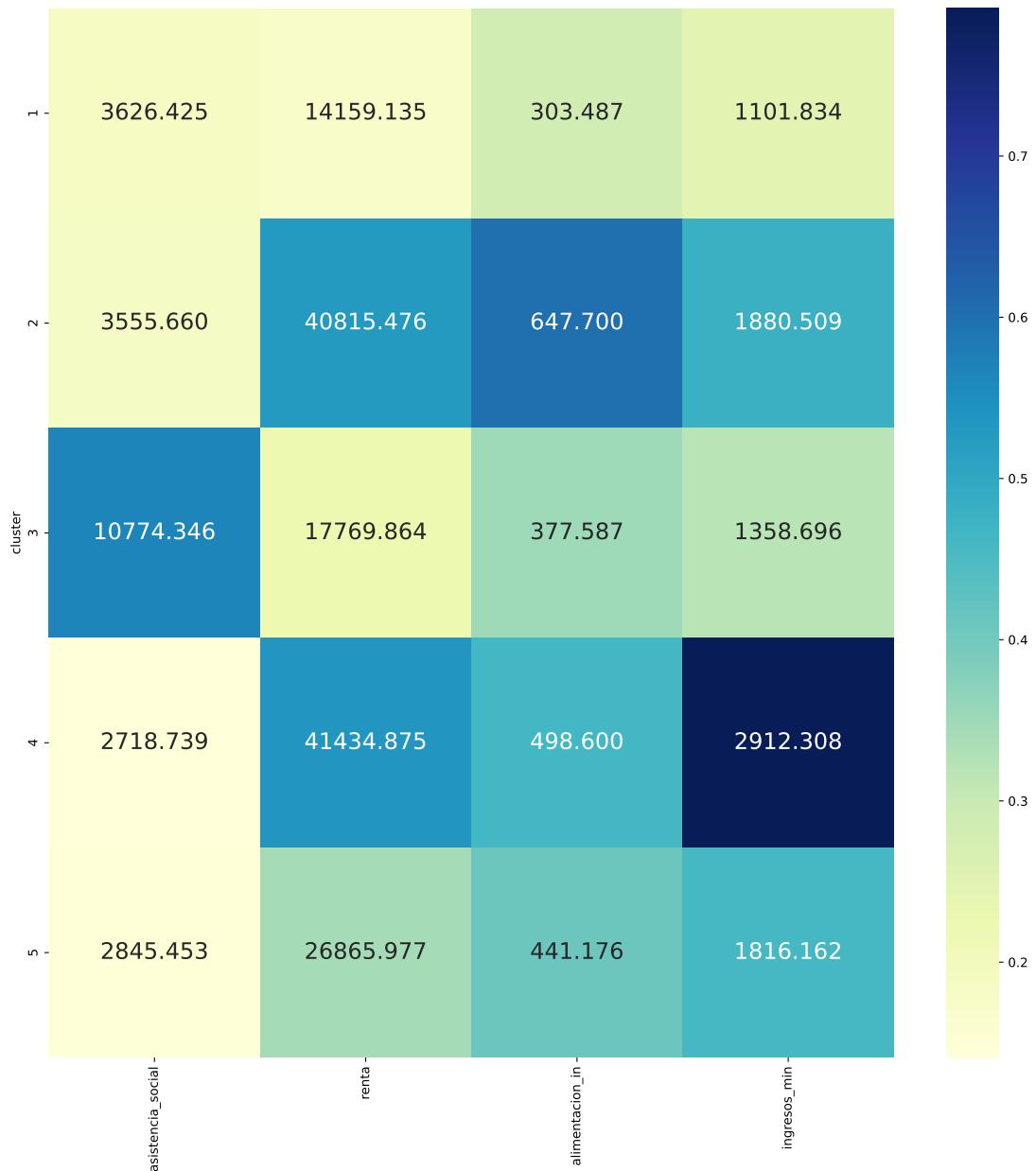


Figura 14: Heatmap de Agglomerative Clustering en el caso 2

El valor que tiene cada cluster en cada variable indica el centroide que tiene. Esto es, el cluster 1 tiene el centroide en 3626.425 con respecto a la variable `asistencia_social`. Además, el color indica que cuanto más azul sea, más determinante será esa variable a la hora de agrupar los objetos. Por ejemplo, los ingresos mínimos de los objetos del cluster 4 es determinante.

En la siguiente figura se muestra la matriz de dispersión de las variables dos a dos y en la diagonal, el histograma de cada variable.

## 2 Casos de estudio

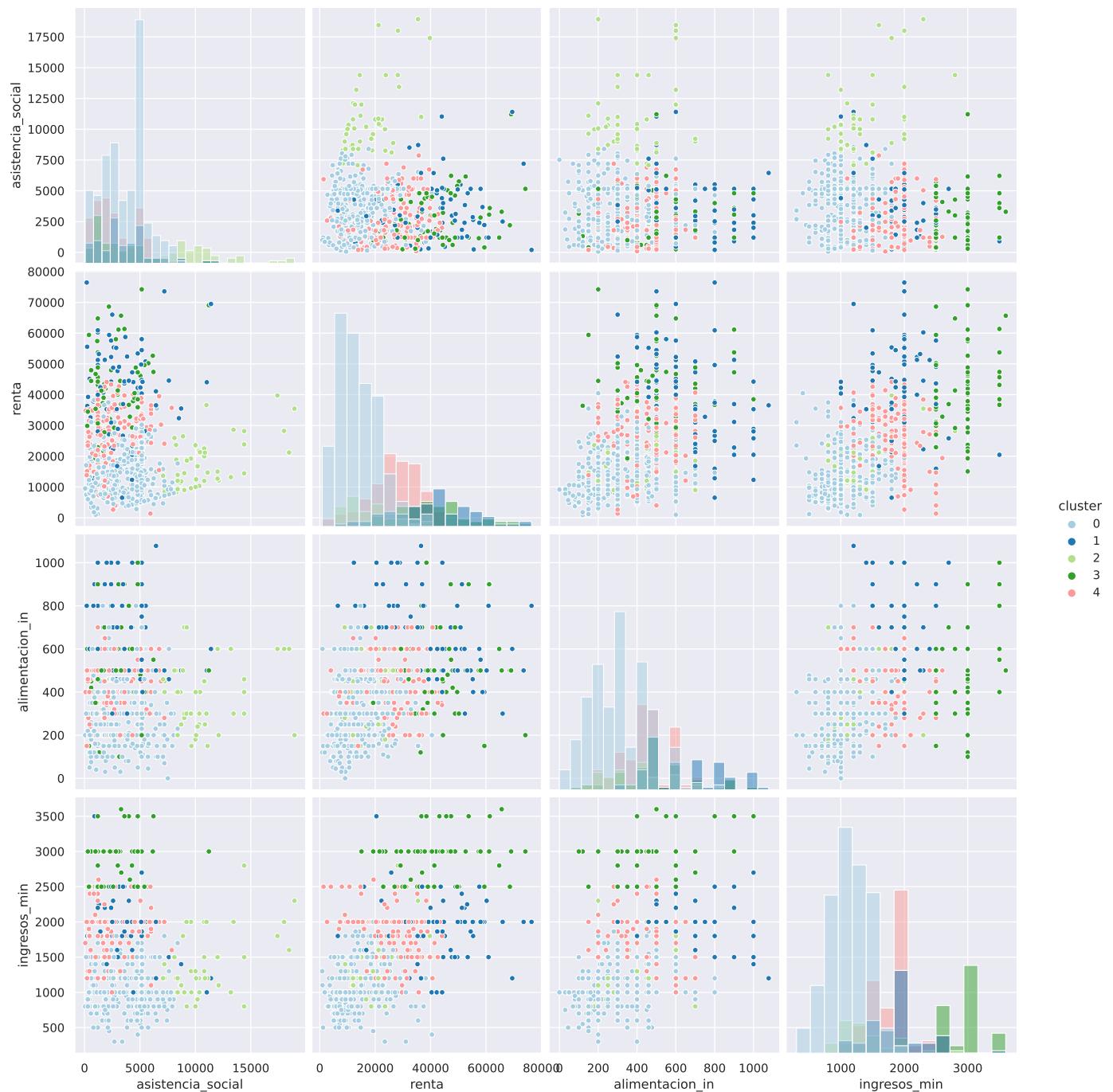


Figura 15: Scatter matrix de Agglomerative Clustering en el caso 2

Observamos que la variable `infresos_min` ayuda a separar los objetos en cuatro de los clusters. Las variables `alimentacion_in` y `renta` tambien aportan información relevante para agrupar los objetos.

La siguiente figura muestra los diagramas de cajas de cada variable. Este diagrama será útil también para distinguir las características de cada grupo, pues nos da información del rango de valores que toman las variables.

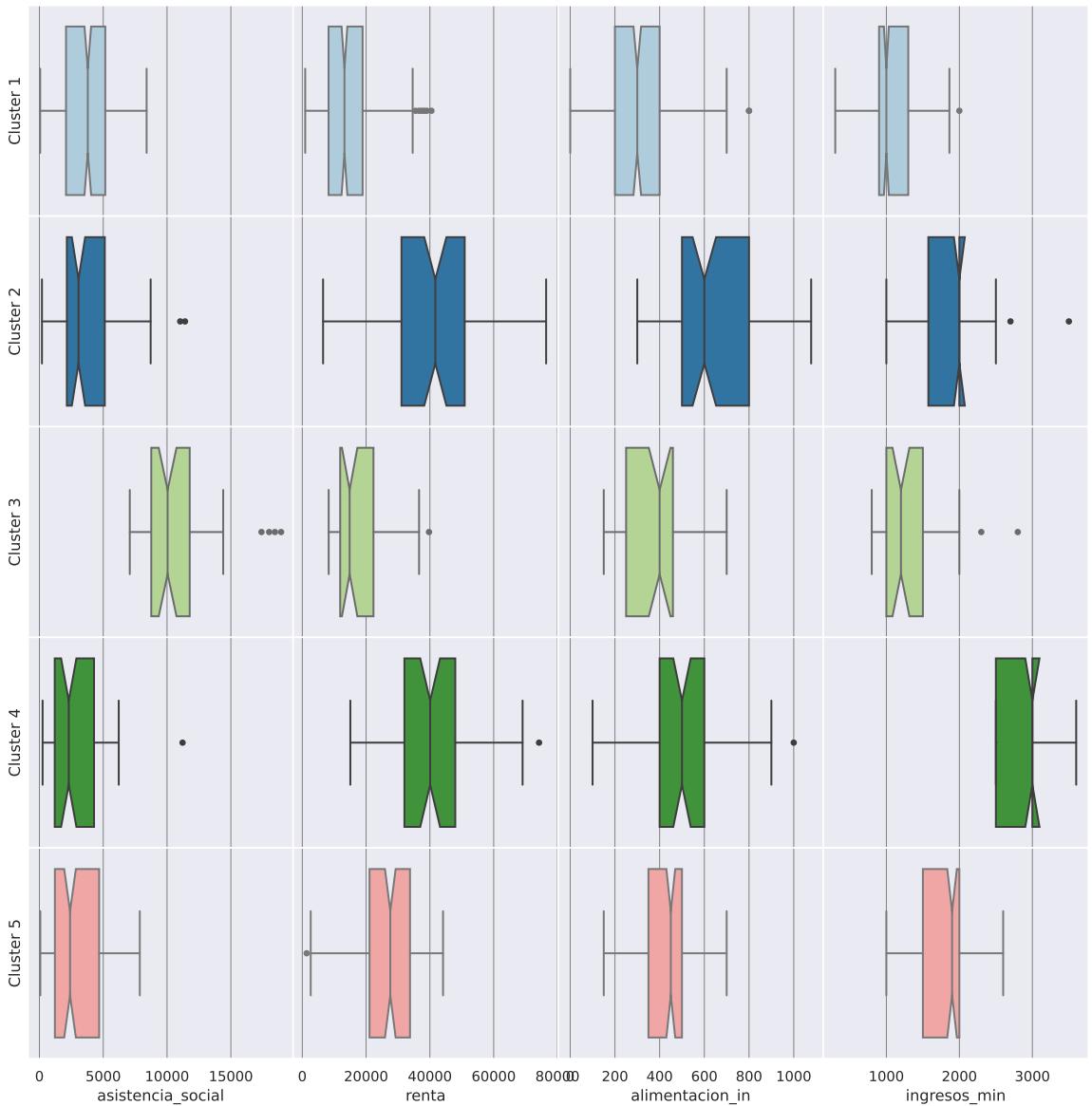


Figura 16: Boxplot de Agglomerative Clustering en el caso 2

En el cluster 1 tenemos que **asistencia\_social** toma valores entre 2.200 y 5.100, **renta** toma valores entre 10.000 y 19.000, **alimentacion\_in** toma valores entre 200 y 400 e **ingresos\_min** toma valores entre 9.000 y 13.000. Este cluster tiene la renta más baja, necesita pocos ingresos mínimos para llegar a fin de mes, tiene pocos gastos en la alimentación y recibe pocos ingresos de asistencia social. Por lo que este cluster contiene a la gente que tiene poco dinero y gasta muy poco, pero aun así necesitan poco dinero por lo que reciben poca asistencia social.

En el cluster 2 tenemos que **asistencia\_social** toma valores entre 2.200 y 5.100, al igual que en el cluster anterior. Se diferencia de él porque tiene mayor renta, entre 30.000 y 50.000, gasta mucho más en alimentación, entre 500 y 800 y necesita más ingresos mínimos, entre 1.600 y 2.000.

Por lo que este cluster puede que contenga a personas que tienen personas dependientes a las que tienen que alimentar o hacerse cargo de ellas, eso explicaría el elevado gasto en alimentación y los ingresos mínimos que necesitan. Aun así, tienen mucha renta por lo que reciben poca asistencia social.

En el cluster 3 tenemos que **asistencia\_social** toma valores entre 9.000 y 12.000, la más elevada de todos los clusters. Es por esto que tiene poca renta, entre 11.000 y 21.000, no gasta demasiado en alimentación, entre 220 y 420, y necesita entre unos 1.000 y 1.500 ingresos mínimos. Este cluster contiene a las personas que tienen poca renta, pero necesitan dinero para la alimentación y para llegar a final de mes, es por esto que reciben mucha más asistencia social.

En el cluster 4 tenemos que **asistencia\_social** toma valores entre 1.000 y 4.500. Tienen una renta elevada, entre 30.000 y 45.000, tienen más gastos en alimentación, entre 400 y 600, y, lo que más destaca, es que necesitan unos ingresos mínimos muy elevados, entre 2.500 y 3.000. Es por esto que, al necesitar tantos ingresos mínimos y los gastos en alimentación, no les alcanza con su renta y es por ello que necesitan un poco de asistencia social.

En el cluster 5 tenemos que **asistencia\_social** toma valores entre 1.000 y 4.700, similar al cluster anterior. Sin embargo, este tiene menos renta, entre 20.000 y 33.000, gasta un poco menos en alimentación, entre 370 y 500, y necesita pocos ingresos mínimos, entre 1.500 y 2.000. Este cluster contiene a la gente que no tiene mucha renta, pero aun así con los gastos en alimentación y los ingresos mínimos para llegar a final de mes, necesitan un poco de asistencia social.

En la siguiente imagen se representan los distintos clusters y cómo de cerca o lejos están unos de otros y el tamaño de cada uno.

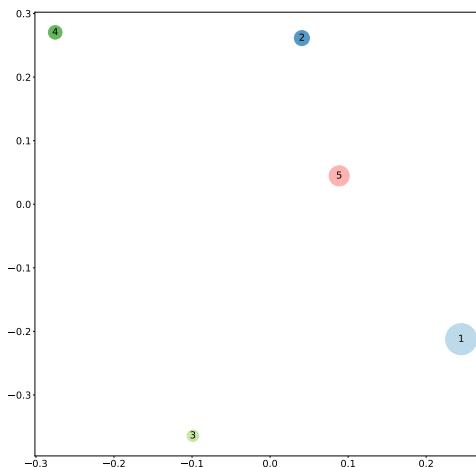


Figura 17: Multidimensional scaling de Agglomerative Clustering en el caso 2

Y en la siguiente imagen, se puede ver la distancia intercluster.

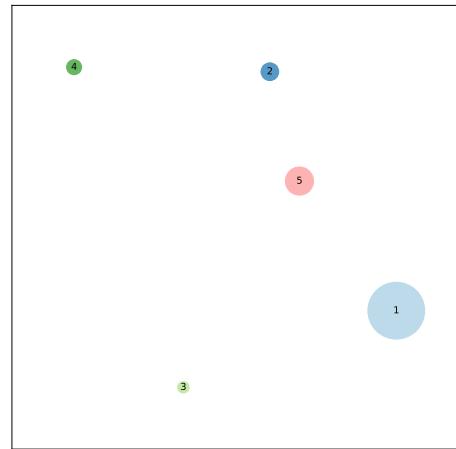


Figura 18: Distancia intercluster de Agglomerative Clustering en el caso 2

A continuación, se muestra la distribución por variable y cluster.

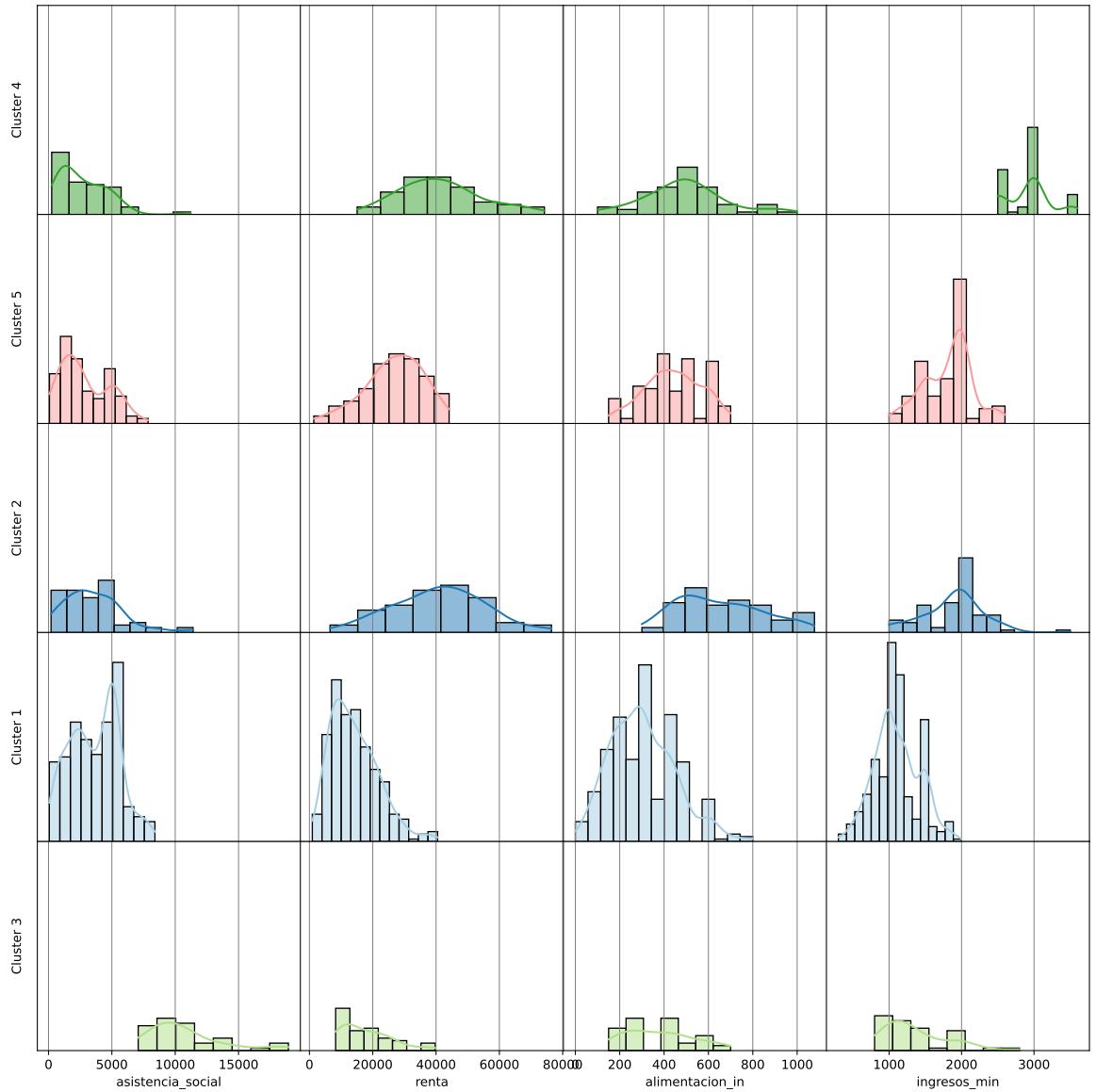


Figura 19: Distribución por variable y cluster de Agglomerative Clustering en el caso 2

Ya sabíamos los rangos de valores que podían tomar las variables, pues ya lo comentamos en la gráfica del Boxplot. Sin embargo, con esta gráfica obtenemos más información sobre qué elementos toman valores mayormente.

En el cluster 4 observamos que las personas suelen recibir una asistencia social cercana a 0 y los ingresos mínimos que necesitan son 3.000.

En el cluster 5 destaca que los ingresos mínimos que necesitan son 2.000, aunque el resto de variables toman valores con frecuencia similar. A diferencia del anterior, en este cluster reciben algo más de asistencia social.

En el cluster 2 destaca que los ingresos mínimos que necesitan son 2.000, al igual que en el cluster

5. A diferencia de este cluster, tienen más renta y gastan más en alimentación.

En el cluster 1 destaca que la asistencia social, mayormente, es de 6.000 y que la mayoría de los ingresos mínimos que necesitan son de 1.000.

En el resto de clusters y variables, cabe destacar que casi todos siguen una normal. No hay valores que destaque por ser más frecuentes que los ya comentados.

A continuación, se muestra una gráfica con las coordenadas paralelas para visualizar y analizar los clusters.

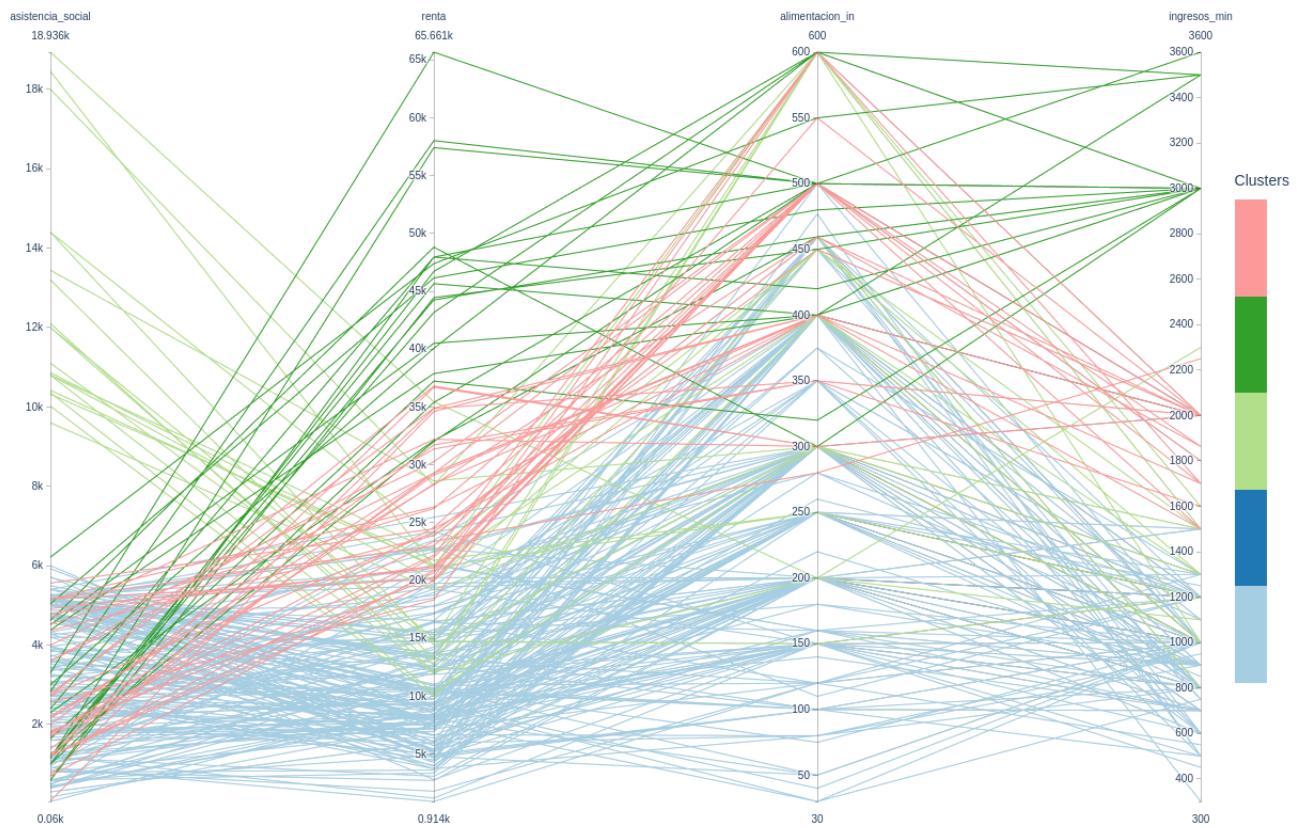


Figura 20: Parallel coordinates de Agglomerative Clustering en el caso 2

A diferencia de los otros algoritmos que hemos usado, lo que destaca de este algoritmo es que es un algoritmo jerárquico aglomerativo. Por lo que tiene mucha importancia cómo genera los clusters, es decir, cuáles son las variables por las que empieza agrupando. Para ello, se han representado varios dendrogramas.

A continuación podemos ver el dendrograma general de este algoritmo en este caso de estudio.

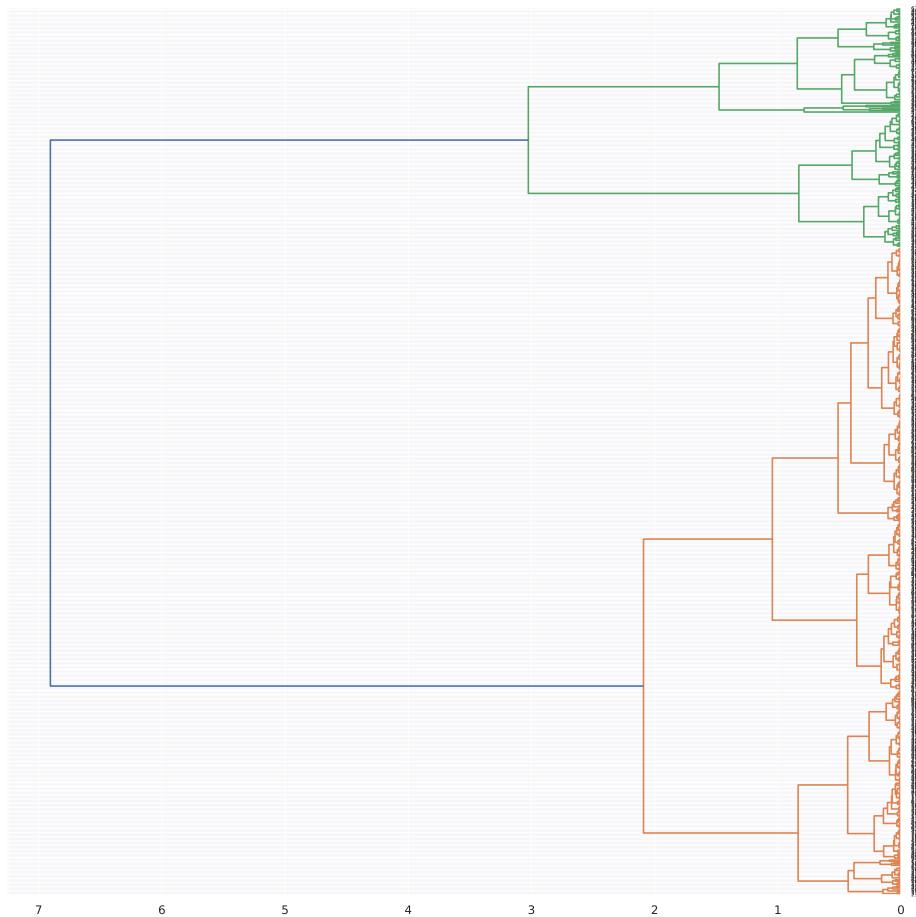


Figura 21: Dendrograma de Agglomerative Clustering en el caso 2

Como no se distinguen mucho, en la siguiente figura se muestra el mismo diagrama pero truncado a 15 niveles.

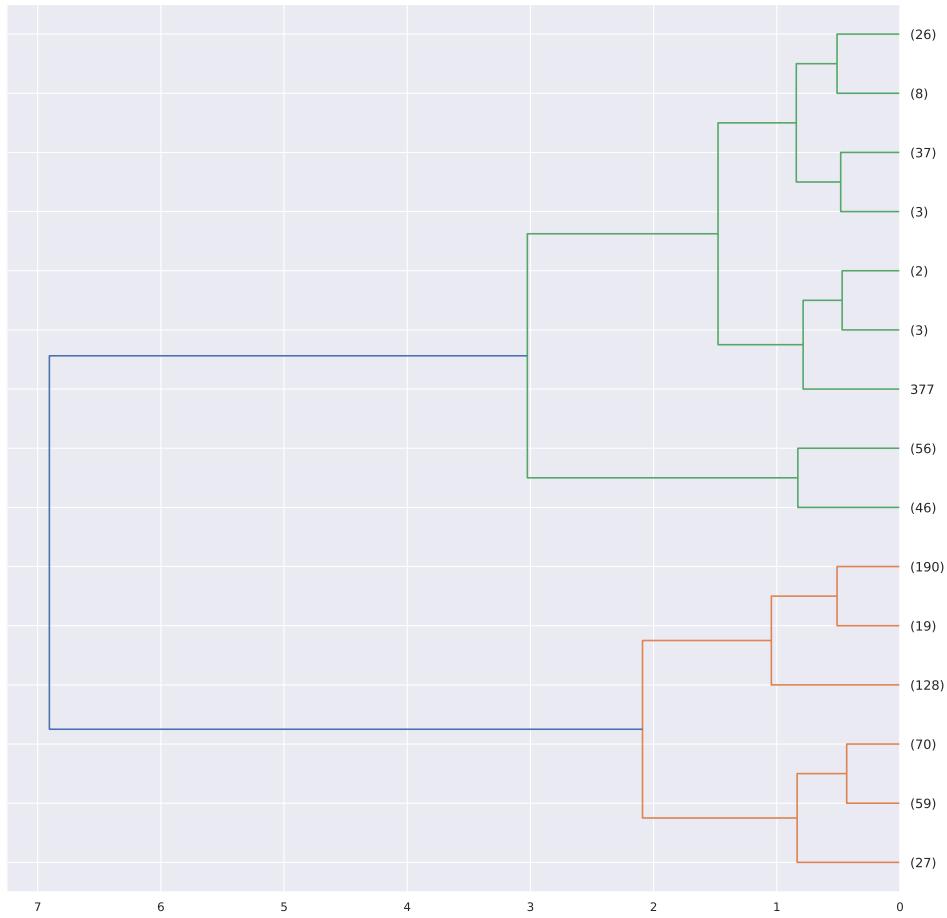


Figura 22: Dendrograma (hasta el nivel 15) de Agglomerative Clustering en el caso 2

Cada número entre paréntesis en el eje vertical indica el número de objetos que pertenecen al cluster correspondiente. El eje horizontal indica la distancia entre los enlaces. Observamos que primero se agruparon las hojas que están a distancia 0.5, esto es (26) y (8), (37) y (3), (2) y (3), (190) y (19), (70) y (59). Conforme avanzamos hacia la izquierda, podemos ver nuevas agrupaciones según la distancia a la que están.

Para ver la importancia de las variables en cada agrupación, podemos observar la siguiente gráfica.

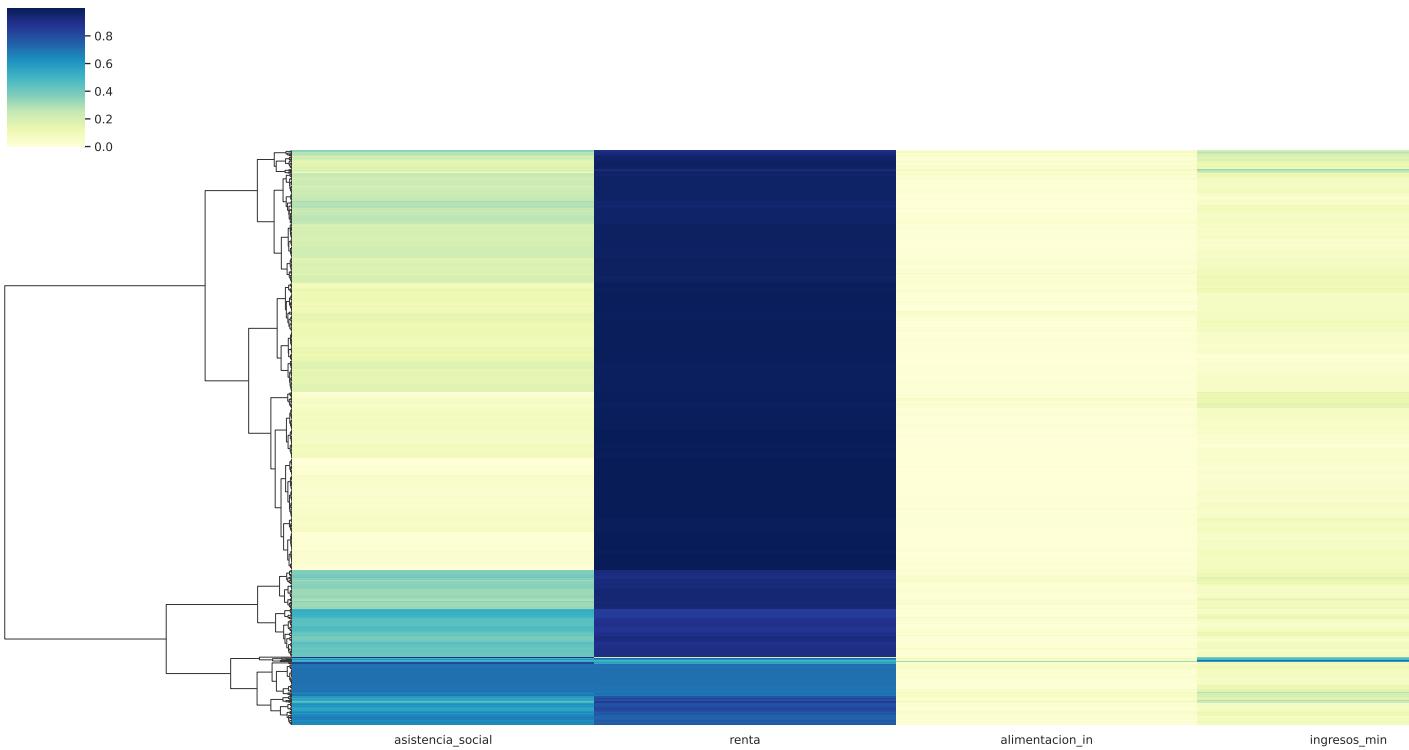


Figura 23: Dendrogramma Heatmap de Agglomerative Clustering en el caso 2

Observamos que a partir de la variable `renta` podemos distinguir dos grupos, en las que se distinguen la rama superior e inferior. En las siguientes ramas, se puede observar que divide en función de la variable `asistencia_social`. De hecho, en la rama inferior se divide en función de las variables `asistencia_social` y `renta`, de nuevo. Observamos que la variable `alimentacion_in` apenas aporta información. La variable `ingresos_min` tampoco aporta demasiada información, aunque algo más que `alimentacion_in`.

En conclusión, la variable más significativa es `renta`.

**Análisis de parámetros** En este apartado se analizará el parámetro decisivo de Agglomerative Clustering, el número de clusters. Por defecto, se ha ejecutado con 5 clusters. En este apartado variaremos este número de tal forma que estará comprendido entre 2 y 20, pues no nos interesa que sea demasiado elevado pues no obtendríamos apenas información. De esta forma, en la siguiente tabla se muestran los resultados para cada valor de este parámetro.

Tabla 7: Resultados caso de estudio 2

Número de clusters	Tiempo (s)	Calinski-Harabasz	Silhouette
2	0.008	377.560	0.32303
3	0.012	255.870	0.28631
4	0.013	226.372	0.25233
5	0.014	201.209	0.21281
6	0.021	190.010	0.15443
7	0.026	183.451	0.16387
8	0.026	176.994	0.15762
9	0.014	166.034	0.16051
10	0.014	158.148	0.16374
11	0.026	152.469	0.15597
12	0.026	148.655	0.15425
13	0.035	146.360	0.15929
14	0.035	142.909	0.16574
15	0.034	140.026	0.16837
16	0.024	137.550	0.16904
17	0.034	135.363	0.17061
18	0.025	132.093	0.16436
19	0.034	129.435	0.16272
20	0.034	127.038	0.16762

Destaca que al aumentar el número de clusters, aumenta el tiempo de ejecución. Esto se debe a que si aumenta el número de clusters, aumenta el número de operaciones a realizar. Con respecto a los índices Silhouette y Calinski-Harabasz, en las siguientes figuras se muestra su gráfica para cada número de clusters.

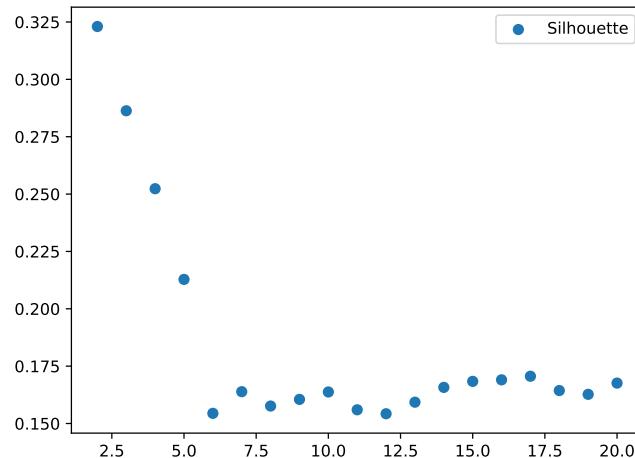


Figura 24: Silhouette de Agglomerative Clustering en el caso 2

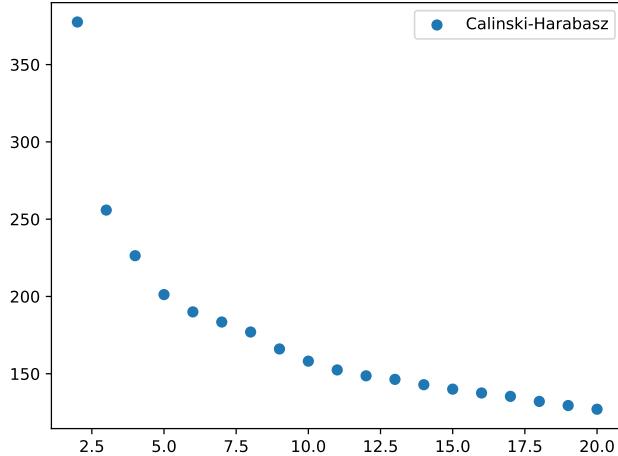


Figura 25: Calinski-Harabasz de Agglomerative Clustering en el caso 2

Observamos que, para un número pequeño de clusters, obtenemos mejores índices Silhouette y Calinski-Harabasz. Esto se debe a que, al haber pocos clusters posibles, los objetos se quedan más cerca de su centro. Además, observamos que el decremento en cada medida es distinta. Esto puede deberse a que, al aumentar el número de clusters, haya menos dispersión intra-clusters por lo que el índice Calinski-Harabasz disminuye. En cuanto al índice Silhouette, al aumentar el número de clusters, no habrá tanta diferencia entre los clusters por lo que disminuye con menos rapidez.

En conclusión, para este caso de estudio con este algoritmo, deberíamos establecer el número de clusters a 2, que es cuando obtenemos un mejores valores de ambas medidas.

### 2.2.3 Interpretación de la segmentación

A partir de los resultados obtenidos, concluimos que la variable `renta` determina los distintos clusters. Pues las personas con menos renta reciben más asistencia social que las personas con mayor renta. También es determinante la variable `ingresos_min` porque, por mucha renta que tengas, si necesitas muchos ingresos para llegar a final de mes, acabarás con poco dinero por lo que también necesitarás recibir más asistencia social.

### 2.3 Caso de estudio 3

El tercer caso de estudio se basará en el alquiler imputado. Las variables que nos ayudarán a llevar a cabo el agrupamiento son las siguientes.

- **HY030N.** Alquiler imputado. Toma valores entre 1 y 99999999.99
- **HY020.** Renta disponible total del hogar en el año anterior al de la encuesta. Toma valores entre -99999999.99 y 99999999.99
- **HC010.** Durante el mes pasado, importe que el hogar gastó en alimentos y bebidas no alcohólicas para ser consumidas en casa. Toma valores entre 0 y 99999999.99
- **HH061.** Importe mensual que tendría que pagar por el alquiler de una vivienda como esta a precio de mercado. Toma valores entre 1 y 99999999.99.
- **HC040.** Durante un mes normal, importe de gastos en transporte privado (gasolina, aparcamiento regulado, peajes, etc, gasto en mantenimiento/revisión, reparación o seguro del vehículo). Toma valores entre 0 y 999999.

Los resultados obtenidos al ejecutar los cinco algoritmos antes mencionados con este caso de estudio y estas variables se recogen en la siguiente tabla.

Tabla 8: Resultados caso de estudio 3

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
k-Means	1.822	3109.938	0.19927	5
MeanShift	87.652	288.929	0.36830	3
Birch	0.333	1552.710	0.22402	5
Agglomerative Clustering	3.588	2401.951	0.18364	5
DBSCAN	0.650	155.352	0.29576	4

Con respecto al **tiempo de ejecución**, destaca el de MeanShift por ser notablemente superior al del resto, esto se debe a que su complejidad tiende a  $O(T * n * \log(n))$ , con  $n$  el número de muestras y  $T$  el número de puntos. En cuanto a los otros algoritmos, apenas superan los 4 segundos.

El índice **Calinski-Harabasz** de los algoritmos k-Means y Agglomerative Clustering es bastante alto, a diferencia del de los algoritmos MeanShift y DBSCAN. Esto puede deberse a que estos dos últimos algoritmos no agrupan bien los objetos en los clusters, pues analizando los clusters que generan, siempre hay un cluster con la mayoría de los objetos y otros clusters con tamaño muy pequeño con los objetos que no puede agrupar, por lo que no se realiza un buen agrupamiento.

El índice **Silhouette** es bastante similar en k-Means, Birch y Agglomerative Clustering los algoritmos. Los algoritmos MeanShift y DBSCAN proporcionan unos índices Silhouette más elevados, pues por lo ya comentado antes, tenemos que los objetos del cluster con tamaño más grande estén muy separados del otro cluster.

Con respecto al **número de clusters**, los algoritmos k-Means, Birch y AgglomerativeClustering generan 5 clusters pues así se lo indicamos en los parámetros. Los algoritmos MeanShift y DBSCAN generan 3 y 4 clusters, respectivamente, pero realmente un cluster tiene la mayoría de los elementos y el resto apenas tienen elementos.

En resumen, para este caso los algoritmos k-Means y Agglomerative Clustering, que son los que estudiaremos más adelante, son los que mejor resultados obtienen, teniendo en cuenta el tiempo de ejecución y los índices Silhouette y Calinski-Harabasz.

### 2.3.1 k-Means

Este algoritmo genera 5 clusters, cuya organización se puede observar en la siguiente figura.

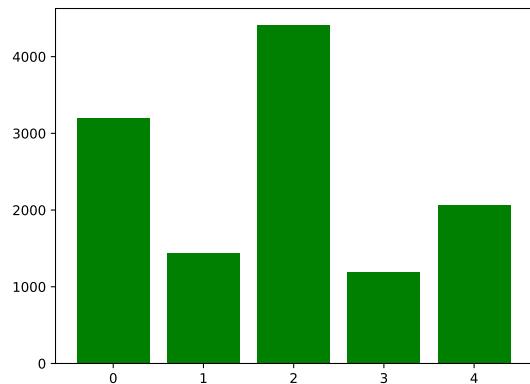


Figura 26: Tamaño de los clusters de k-Means en el caso 3

Observamos que el cluster número 2 tiene el mayor número de objetos (el 36%) y el cluster número 3, el que menos (un 10%). Sin embargo, no hay tantas diferencias entre los tamaños a diferencia de los anteriores casos de estudio.

En la siguiente figura podemos observar el Heatmap de este algoritmo. Esta gráfica puede ayudarnos a determinar las características de cada cluster. Esto es, cuanto más alto sea el valor, mayor determinación habrá.



Figura 27: Heatmap de k-Means en el caso 3

El valor que tiene cada cluster en cada variable indica el centroide que tiene. Esto es, el cluster 1 tiene el centroide en 6728.329 con respecto a la variable `alquiler_imputado`. Además, el color indica que cuanto más azul sea, más determinante será esa variable a la hora de agrupar los objetos. Por ejemplo, los el alquiler imputado de los objetos del cluster 4 es muy determinante.

En la siguiente figura se muestra la matriz de dispersión de las variables dos a dos y en la diagonal, el histograma de cada variable.

## 2 Casos de estudio

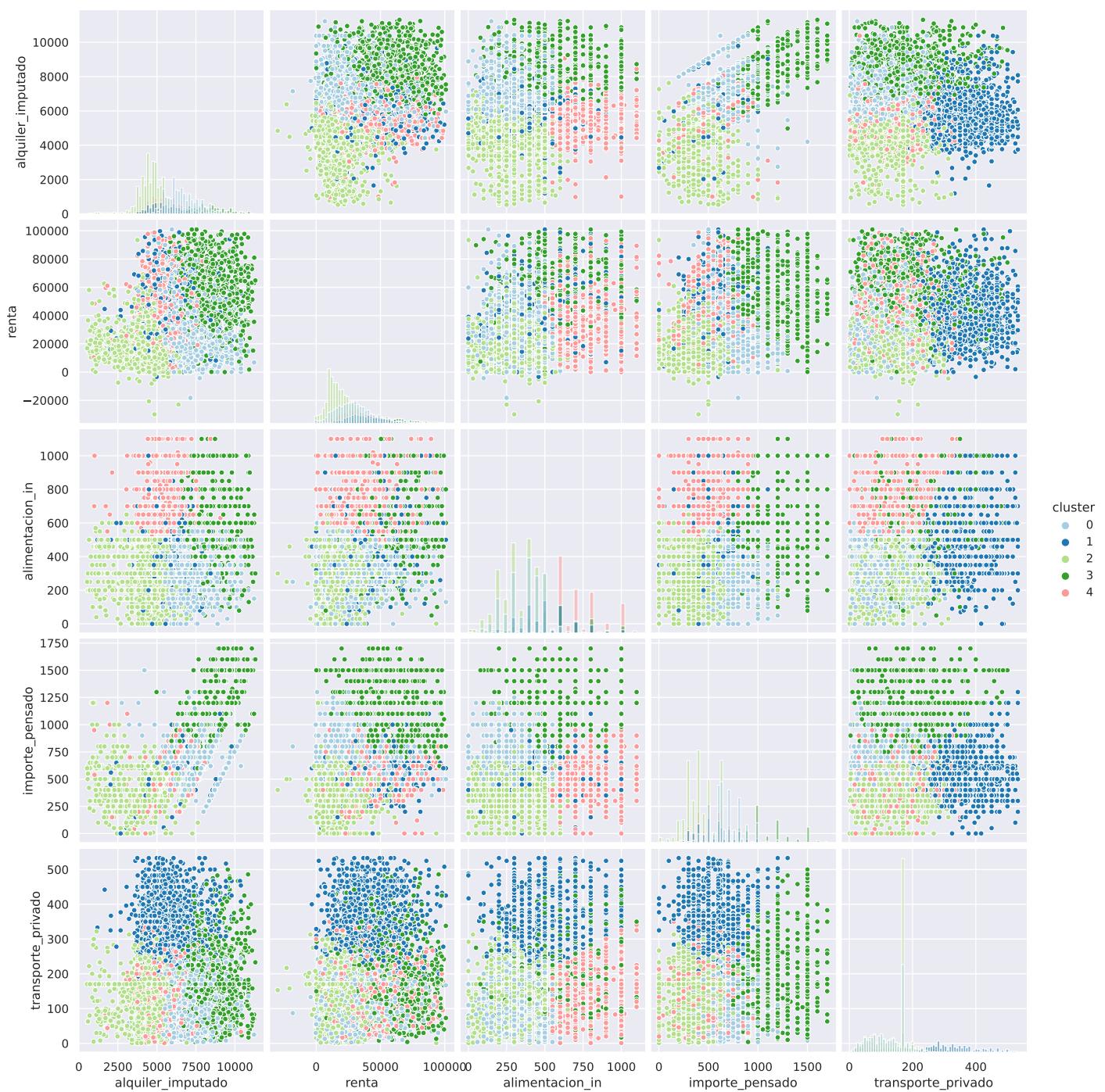


Figura 28: Scatter matrix de k-Means en el caso 3

Observamos que las variables **renta**, **alimentacion\_in**, **importe\_pensado** y **transporte\_privado** ayudan a separar los objetos en cuatro de los clusters.

La siguiente figura muestra los diagramas de cajas de cada variable. Este diagrama será útil también para distinguir las características de cada grupo, pues nos da información del rango de valores que toman las variables.

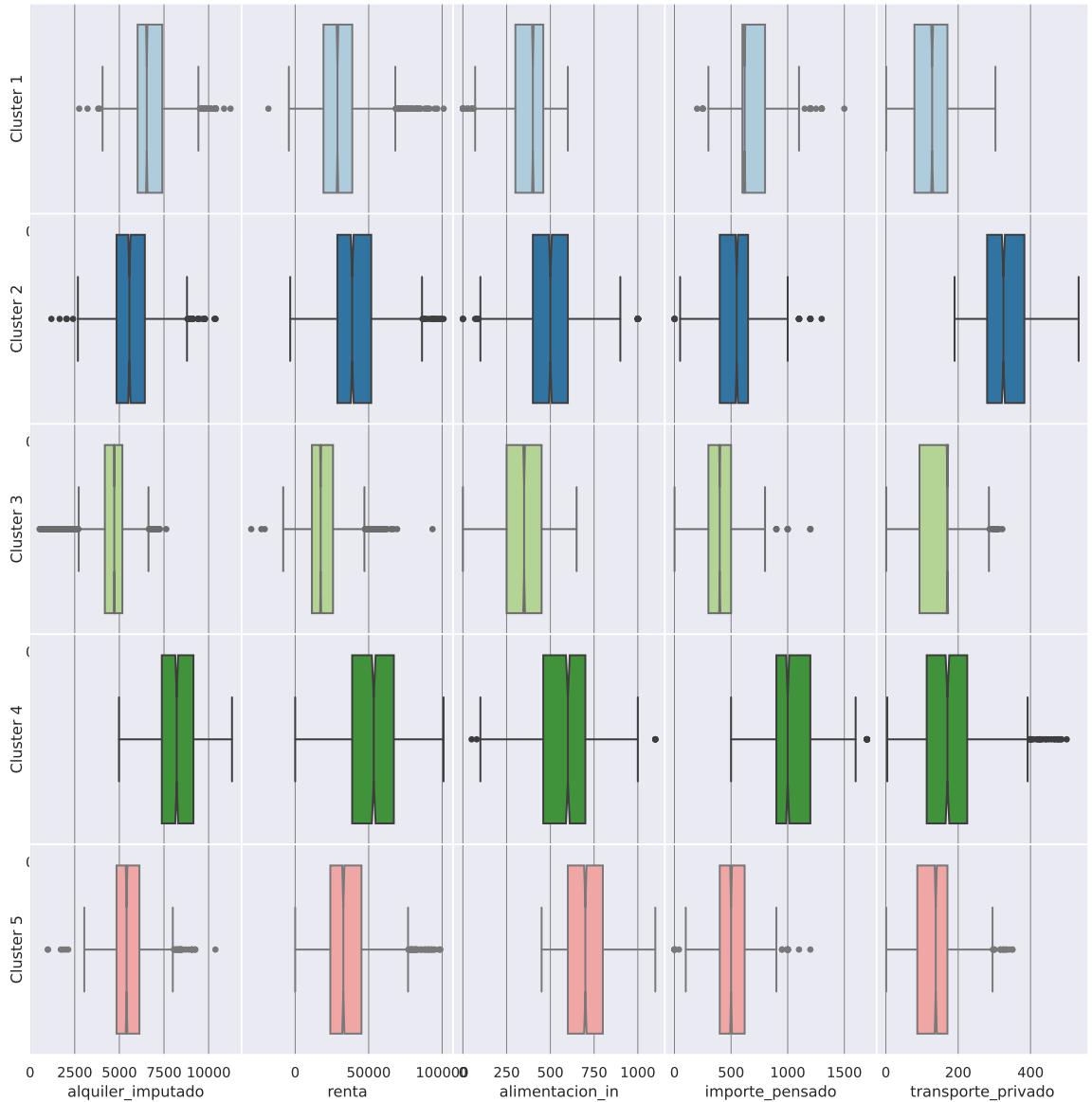


Figura 29: Boxplot de k-Means en el caso 3

En el cluster 1 tenemos que `alquiler_imputado` toma valores entre 6.000 y 7.500, `renta` toma valores entre 20.000 y 40.000, `alimentacion_in` toma valores entre 280 y 400, `importe_pensado` toma valores entre 600 y 750 y `transporte_privado` toma valores entre 100 y 180. En este cluster, en general, las personas tienen un alquiler imputado algo elevado, pero su renta, sus gastos en alimentación y en transporte privado son bajos.

En el cluster 2 tenemos que `alquiler_imputado` toma valores entre 4.800 y 6.500. Tiene algunos gastos más en alimentación, entre 400 y 600. Pero destaca el elevado gasto en transporte privado, entre 300 y 400, que es lo que determina la agrupación en este cluster.

En el cluster 3 tenemos que `alquiler_imputado` toma valores entre 4.000 y 5.100, el menos

elevado de todos los clusters. Es por esto que tiene poca renta, entre 10.000 y 20.000, no gasta demasiado en alimentación, entre 250 y 450, y tampoco gasta demasiado en transporte privado, entre 100 y 180. Este cluster contiene a las personas que tienen poca renta y tienen muy pocos gastos, es por eso que el alquiler imputado es bastante más bajo que en el resto de clusters.

En el cluster 4 tenemos que **alquiler\_imputado** toma valores entre 7.500 y 9.000, el más elevado de todos. Es por esto que la renta es la más elevada también, entre 40.000 y 65.000. De la misma forma, tienen muchos gastos en alimentación, entre 450 y 700, aunque no gastan demasiado en el transporte privado, entre 120 y 220. Es por esto que, al tener una renta tan elevada y no gastar demasiado en alimentación ni en el transporte privado, pueden permitirse tener un alquiler imputado algo más elevado que el resto.

En el cluster 5 tenemos que **alquiler\_imputado** toma valores entre 5.000 y 6.000. Lo que destaca de este cluster es que tiene los gastos en alimentación más elevados que el resto de clusters, aunque su renta no es ni alta ni baja, al igual que los gastos en transporte privado.

En la siguiente imagen se representan los distintos clusters y cómo de cerca o lejos están unos de otros y el tamaño de cada uno.

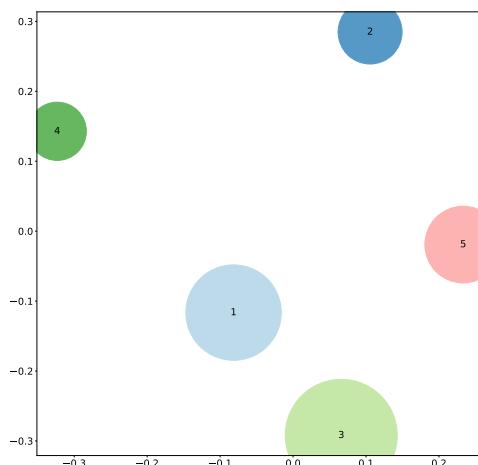


Figura 30: Multidimensional scaling de k-Means en el caso 3

Y en la siguiente imagen, se puede ver la distancia intercluster.

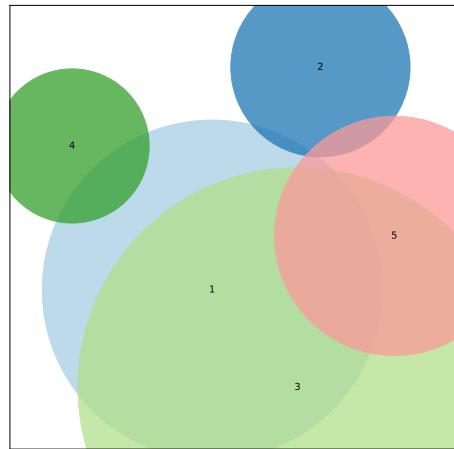


Figura 31: Distancia intercluster de k-Means en el caso 3

A continuación, se muestra la distribución por variable y cluster.

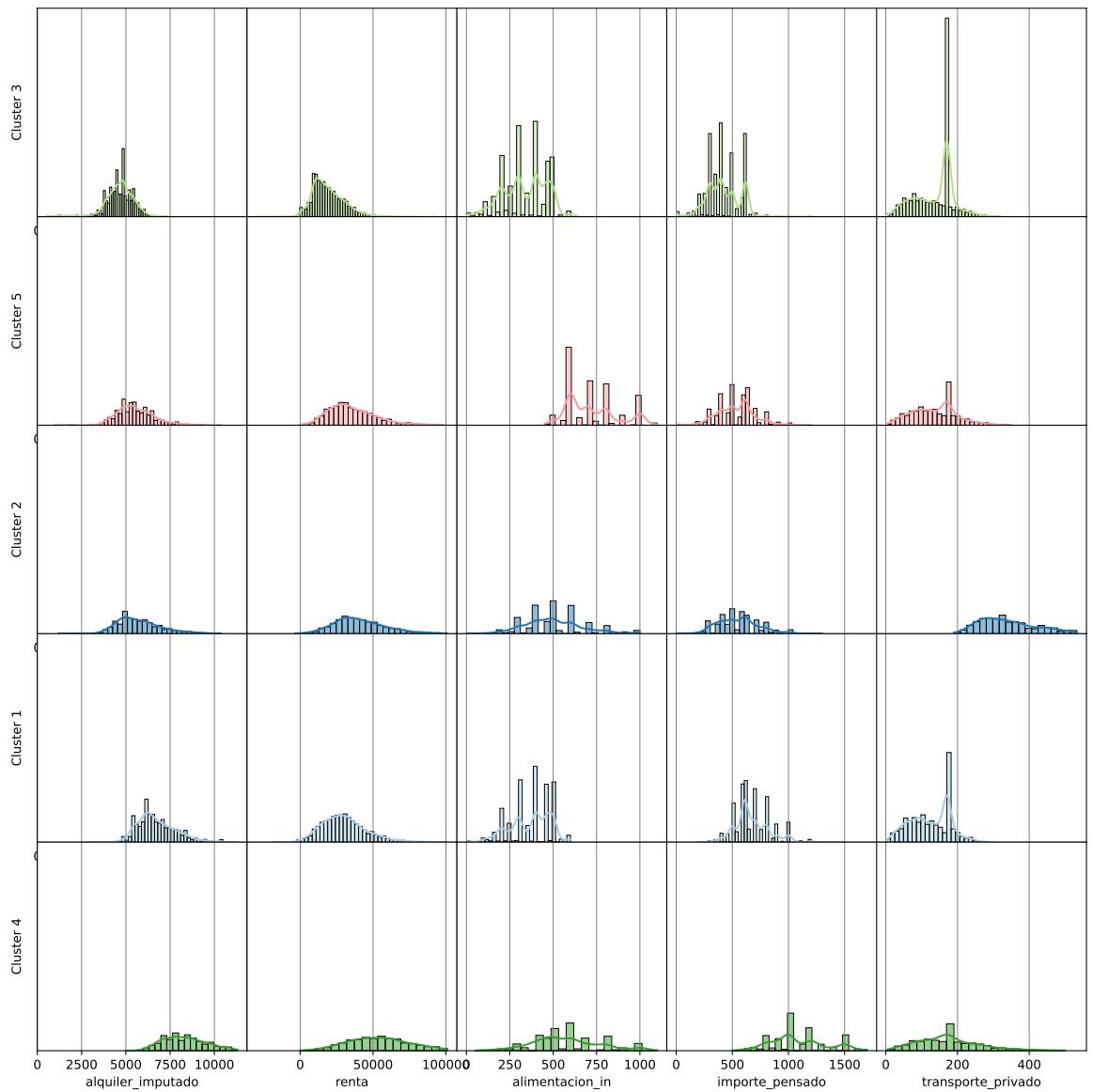


Figura 32: Distribución por variable y cluster de k-Means en el caso 3

Ya sabíamos los rangos de valores que podían tomar las variables, pues ya lo comentamos en la gráfica del Boxplot. Sin embargo, con esta gráfica obtenemos más información sobre qué elementos toman valores mayomente.

En el cluster 3 observamos que las personas suelen tener un alquiler imputado, mayormente, cercano a 5.000. Además, la mayoría tienen una renta baja y los gastos en transporte privado son cercanos a 200.

En el cluster 5 destaca que los gastos en transporte privado son cercanos a 200 y los gastos en la alimentación son cercanos a 550.

En el cluster 1 destaca que los gastos en transporte privado, al igual que en los clusters 3 y 5,

son cercanos a 200.

En el resto de clusters y variables, cabe destacar que casi todos siguen una normal. No hay valores que destaque por ser más frecuentes que los ya comentados.

A continuación, se muestra una gráfica con las coordenadas paralelas para visualizar y analizar los clusters.

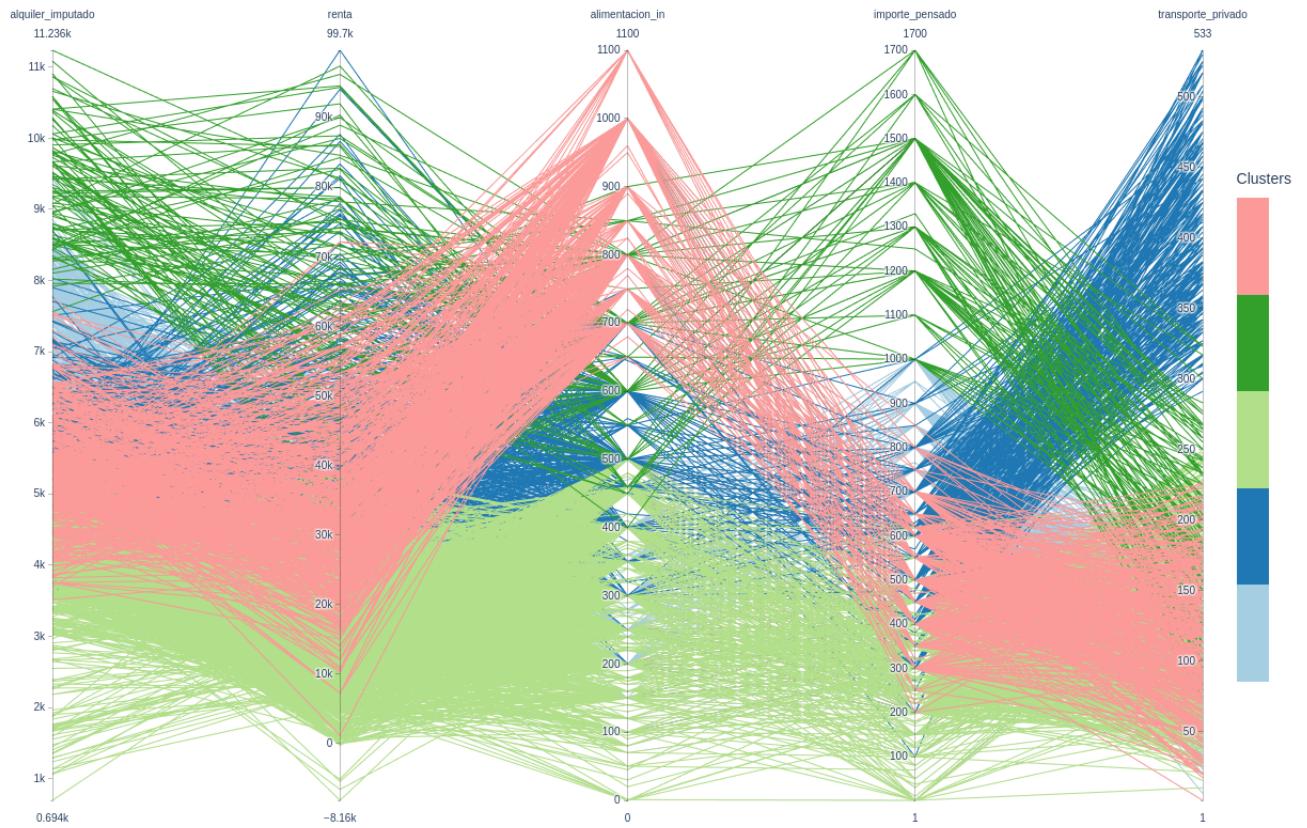


Figura 33: Parallel coordinates de k-Means en el caso 3

**Análisis de parámetros** En este apartado se analizará el parámetro decisivo de k-Means, el número de clusters. Por defecto, se ha ejecutado con 5 clusters. En este apartado variaremos este número de tal forma que estará comprendido entre 2 y 20, pues no nos interesa que sea demasiado elevado pues no obtendríamos apenas información. De esta forma, en la siguiente tabla se muestran los resultados para cada valor de este parámetro.

Tabla 9: Resultados caso de estudio 2

Número de clusters	Tiempo (s)	Calinski-Harabasz	Silhouette
2	0.272	4002.528	0.26033
3	0.378	3433.760	0.26187
4	0.505	3343.067	0.24801
5	0.802	3109.938	0.19927
6	0.880	2886.133	0.18297
7	1.289	2720.233	0.18296
8	1.499	2549.318	0.17313
9	1.399	2409.520	0.17151
10	1.582	2296.521	0.16733
11	2.209	2202.718	0.16715
12	4.466	2120.696	0.17349
13	2.949	2042.751	0.16206
14	2.968	1984.332	0.16468
15	3.006	1924.884	0.16579
16	3.568	1871.764	0.16649
17	3.650	1817.845	0.16452
18	3.868	1770.462	0.16503
19	3.956	1719.282	0.16301
20	4.525	1674.861	0.16238

Destaca que al aumentar el número de clusters, aumenta el tiempo de ejecución considerablemente. Esto se debe a que si aumenta el número de clusters, aumenta el número de operaciones a realizar. Con respecto a los índices Silhouette y Calinski-Harabasz, en las siguientes figuras se muestra su gráfica para cada número de clusters.

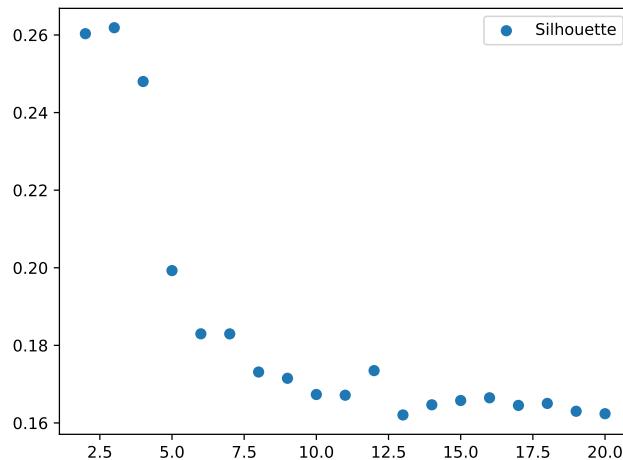


Figura 34: Silhouette de k-Means en el caso 3

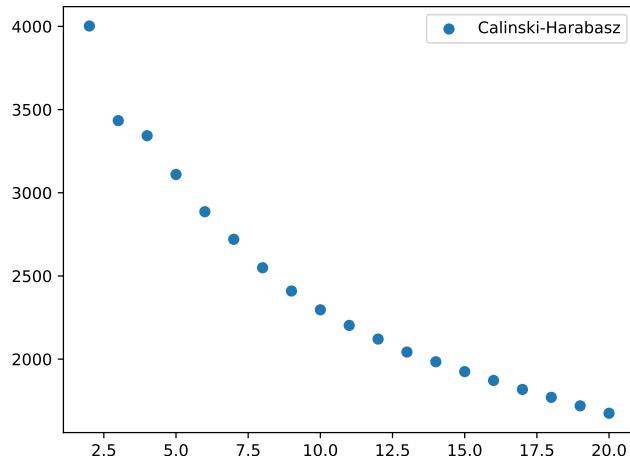


Figura 35: Calinski-Harabasz de k-Means en el caso 3

Observamos que, para un número pequeño de clusters, obtenemos mejores índices Silhouette y Calinski-Harabasz. Esto se debe a que, al haber pocos clusters posibles, los objetos se quedan más cerca de su centro. Además, observamos que el decremento en cada medida es distinta. Esto puede deberse a que, al aumentar el número de clusters, haya menos dispersión intra-clusters por lo que el índice Calinski-Harabasz disminuye. En cuanto al índice Silhouette, al aumentar el número de clusters, no habrá tanta diferencia entre los clusters por lo que disminuye con menos rapidez.

En conclusión, para este caso de estudio con este algoritmo, deberíamos establecer el número de clusters a 2, que es cuando obtenemos un mejores valores de ambas medidas.

### 2.3.2 Agglomerative Clustering

**Análisis de parámetros** En este apartado se analizará el parámetro decisivo de Agglomerative Clustering, el número de clusters. Por defecto, se ha ejecutado con 5 clusters. En este apartado variaremos este número de tal forma que estará comprendido entre 2 y 20, pues no nos interesa que sea demasiado elevado pues no obtendríamos apenas información. De esta forma, en la siguiente tabla se muestran los resultados para cada valor de este parámetro.

Tabla 10: Resultados caso de estudio 2

Número de clusters	Tiempo (s)	Calinski-Harabasz	Silhouette
2	3.959	2858.128	0.18681
3	3.609	2723.376	0.20605
4	3.769	2673.030	0.20678
5	4.314	2401.951	0.18364
6	4.103	2201.732	0.11540
7	4.275	2011.869	0.11298
8	4.091	1890.707	0.11554
9	3.951	1780.175	0.11144
10	3.722	1701.860	0.09937
11	3.627	1620.396	0.09675
12	3.669	1558.754	0.09735
13	3.751	1510.083	0.08209
14	4.238	1458.663	0.07911
15	4.004	1411.990	0.07770
16	3.764	1371.738	0.07759
17	4.176	1333.624	0.07780
18	4.081	1302.043	0.08098
19	5.407	1270.980	0.08080
20	4.416	1244.351	0.08155

Destaca que al aumentar el número de clusters, aumenta el tiempo de ejecución. Esto se debe a que si aumenta el número de clusters, aumenta el número de operaciones a realizar. Con respecto a los índices Silhouette y Calinski-Harabasz, en las siguientes figuras se muestra su gráfica para cada número de clusters.

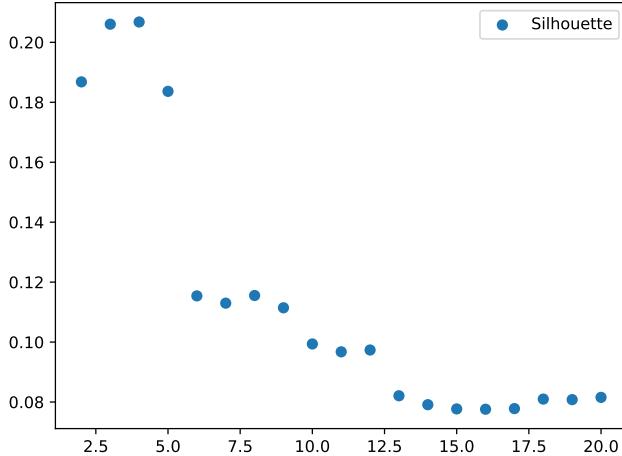


Figura 36: Silhouette de Agglomerative Clustering en el caso 3

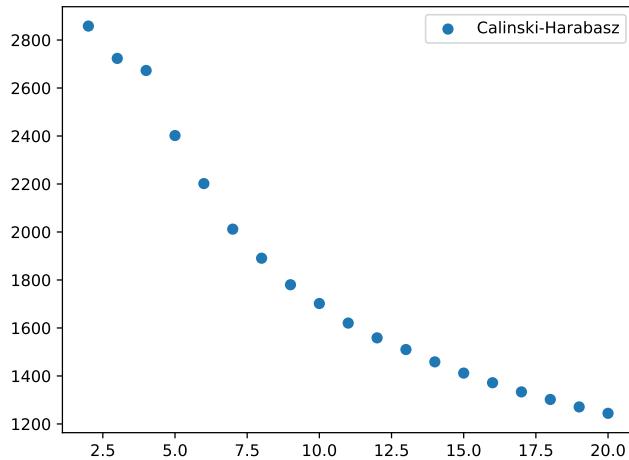


Figura 37: Calinski-Harabasz de Agglomerative Clustering en el caso 3

Observamos que, para un número pequeño de clusters, obtenemos mejores índices Silhouette y Calinski-Harabasz. Esto se debe a que, al haber pocos clusters posibles, los objetos se quedan más cerca de su centro. Además, observamos que el decremento en cada medida es distinta. Esto puede deberse a que, al aumentar el número de clusters, haya menos dispersión intra-clusters por lo que el índice Calinski-Harabasz disminuye. En cuanto al índice Silhouette, al aumentar el número de clusters, no habrá tanta diferencia entre los clusters por lo que disminuye con menos rapidez.

En conclusión, para este caso de estudio con este algoritmo, deberíamos establecer el número de clusters a 2 o 3, que es cuando obtenemos un mejores valores de ambas medidas.

### 2.3.3 Interpretación de la segmentación

A partir de los resultados obtenidos, concluimos que la variable `renta` determina los distintos clusters. Pues las personas con menos renta tienen un elquiler imputado menor que las personas con mayor renta. También es determinante la variable `alimentacion_in` porque son gastos que

## *2 Casos de estudio*

las personas tienen y, puede indicar si tienen personas dependientes a las que necesitan darles de comer o invertir sus gastos en ellas, por lo que si tienen muchos gastos, aunque rengan una renta alta, les quedará poco dinero para el alquiler, luego este alquiler imputado será también menor.

### 3 Contenido adicional

En esta sección se mostrarán los *scripts* para generar las gráficas adicionales.

#### 3.1 Boxplot

```
1 def boxplot(X_alg, usadas, centers, alg, k, n_var):
2     print("----- Boxplots...")
3     sns.set()
4     colors = sns.color_palette(palette='Paired', n_colors=k, desat=
5         None)
6
7     fig, axes = plt.subplots(k, n_var, sharey=True, figsize=(15,15))
8     fig.subplots_adjust(wspace=0, hspace=0)
9
10    rango = []
11    for j in range(n_var):
12        rango.append([X_alg[usadas[j]].min(), X_alg[usadas[j]].max()])
13
14    for i in range(k):
15        dat_filt = X_alg.loc[X_alg['cluster']==i]
16        for j in range(n_var):
17            ax = sns.boxplot(x=dat_filt[usadas[j]], notch=True,
18                color=colors[i], flierprops={'marker':'o',
19                    'markersize':4}, ax=axes[i,j])
20
21            if (i==k-1):
22                axes[i,j].set_xlabel(usadas[j])
23            else:
24                axes[i,j].set_xlabel("")
25
26            if (j==0):
27                axes[i,j].set_ylabel("Cluster "+str(i+1))
28            else:
29                axes[i,j].set_ylabel("")
30
31            axes[i,j].set_yticks([])
32            axes[i,j].grid(axis='x', linestyle='-', linewidth
33                ='0.2', color='gray')
34            axes[i,j].grid(axis='y', visible=False)
```

```

32         ax.set_xlim(rango[j][0]-0.05*(rango[j][1]-rango[j][0]),
33                         rango[j][1]+0.05*(rango[j][1]-rango[j][0]))
34
35     fig.set_size_inches(15,15)
36     fig.savefig("img/" + caso + "/" + alg + "/boxplots.pdf")
37     plt.clf()

```

---

### 3.2 Dendrograma (sin y con mapa de calor)

```

1 def dendrograma(X_alg, alg, usadas):
2     print("----- Dendrogramas . . .")
3
4     # Sin mapa del calor
5     sns.set_theme(color_codes=True)
6     X_normal = preprocessing.normalize(X_alg, norm='l2')
7     linkage_array = hierarchy.ward(X_normal)
8     p = 15
9     hierarchy.dendrogram(linkage_array, orientation='left', p=p,
10                           truncate_mode='lastp') # con la opción truncate_mode podemos
11                           truncar el nivel
12     plt.savefig("img/" + caso + "/" + alg + "/dendrograma_" + str(p)
13                 + ".pdf")
14     plt.clf()
15
16     # Con mapa del calor
17     X_normal = pd.DataFrame(X_normal, index=X_alg.index, columns=
18                             usadas)
19     sns.clustermap(X_normal, method='ward', col_cluster=False,
20                     figsize=(20,10), cmap="YlGnBu", yticklabels=False)
21
22     plt.savefig("img/" + caso + "/" + alg + "/dendrogramaHeatmap.
23                 pdf")
24     plt.clf()

```

---

## 4 Bibliografía

- Material proporcionado por los profesores sobre la asignatura.  
<https://pradogrado2122.ugr.es/>
- Clustering  
<http://scikit-learn.org/stable/modules/clustering.html>
- k-Means y algoritmos de clustering en python  
<http://www.learndatasci.com/k-means-clustering-algorithms-python-intro/>
- Comparación de algoritmos de clustering  
[http://hdbSCAN.readthedocs.io/en/latest/comparing\\_clustering\\_algorithms.html](http://hdbSCAN.readthedocs.io/en/latest/comparing_clustering_algorithms.html)
- Clustering jerárquico y dendogramas  
<https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/>