UNIVERSIDAD DE GRANADA

PRÁCTICA 2: ANÁLISIS RELACIONAL MEDIANTE SEGMENTACIÓN

ANA BUENDÍA RUIZ-AZUAGA

Práctica 2: Análisis relacional mediante segmentación

Correo electrónico
anabuenrua@correo.ugr.es
Grupo de prácticas: A (Jorge Casillas)

E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, a 12 de diciembre de 2021

ÍNDICE GENERAL

1

INTRODUCCIÓN

1.1 CONJUNTO DE DATOS

En esta práctica nos encontramos ante un problema de aprendizaje no supervisado en el que usaremos algoritmos de agrupamiento o clustering para realizar un análisis relacional.

El conjunto de datos sobre el que trabajaremos es el resultado de la encuesta sobre condiciones de vida en 2020, realizada por el Instituto Nacional de Estadística. A partir de estos datos, se realizarán tres casos de estudio distintos a analizar.

El conjunto de datos consta de 15043 respuestas a la encuesta, cada una con 209 variables, tales como la renta, número de miembros en la familia, ayudas sociales, etc. En cada caso de estudio elegiremos un subconjunto de estas variables tratando de obtener la mayor cantidad de información relevante posible.

1.2 ALGORITMOS Y MEDIDAS

En cada uno de los casos de estudio aplicaremos cinco algoritmos de clustering, de los cuales se interpretará uno de ellos, y se hará un estudio de parámetros de otros dos. Los cinco algoritmos que se van a estudiar son:

- **Kmeans:** Agrupa en tantos clusters como se le indique. Se escogen los centros aleatoriamente, se asigna a cada ejemplo el cluster correspondiente al centro más cercano. Se recalculan los centros y se repite el proceso hasta que ningún ejemplo se reasigna. Genera clusters convexos y recibe como argumento el número de clusters. En las ejecuciones se usa n clusters=5.
- **Birch:** Es un algoritmo de clustering incremental, esto es, agrupa los objetos según los va recibiendo. Usa un árbol en el que almacena las características necesarias para determinar los clusters. Cada nodo de este árbol tiene un número de subclusters acotado por el factor de ramificación y un umbral para determinar si el nuevo ejemplo que llega está lo bastante cerca del cluster como para añadirlo a este. También recibe el número de clusters. En las ejecuciones se han fijado los parámetros a 15 como factor de ramificación y 0.1 como threshold.

- **Spectral cluster:** Este algoritmo requiere de otro auxiliar, como kmeans. Consiste en diagonalizar la matriz de afinidad, calcular sus autovectores y sobre estos aplica kmeans. Recibe el número de clusters. Para la ejecución se ha usado n clusters=5
- **DBSCAN:** Es un algoritmo basado en densidad, por lo que no está forzado a obtener clusters convexos. Usa dos parámetros: ϵ que determina cuándo un objeto es densamente alcanzable a partir de otro y min samples, un número mínimo de objetos por los que podemos alcanzar a otro para agruparlos en un mismo cluster. Genera un cluster especial, etiquetado como -1 que contiene los puntos que no pueden ser agrupados. En la ejecución se ha usado ϵ=0,126 y min samples=20
- **Meanshift:** Es un algoritmo basado en densidad de las muestras, parte de un radio fijo que recibe como parámetro, determina un número de clusters y desplaza sus centros hacia las regiones más densas. Genera clusters convexos. Durante las ejecuciones se ha estimado automáticamente el mejor radio.

Para comparar los resultados obtenidos por cada algoritmo se van a usar las siguientes medidas:

- **Tiempo**: No es una medida especialmente relevante, pues no refleja la calidad del clustering realizado por cada algoritmo, pero puede servir para decantarnos por un algoritmo u otro en un momento dado.
- **Calinski-Harabasz**: Ratio entre la dispersión intercluster e intracluster para los clusters. Cuanto más alto sea su valor, mas densos y similares son los grupos y más distintos son de los demás. Es más difícil de interpretar pues no está normalizado.
- **Silhouette**: Mide la similaridad entre los objetos de un mismo cluster comparados con los de los clusters restantes. Toma valores en $[-1, 1]$. Un valor cercano a 1 indica clusters concentrados y separados entre sí.
- **Número de clusters**: Esta medida no tiene sentido en algoritmos para los que se fija de antemano el número de clusters a realizar, como es kmeans, pero puede ser útil para otros algoritmos, pudiendo ver así si generan pocos o demasiados clusters.

1.3 PREPROCESADO

Para la aplicación de los distintos algoritmos se ha usado un preprocesado básico en todos los casos de estudio, que consiste en la imputación de valores perdidos en todas las variables. Notemos que esta imputación no debería tener un efecto importante en el clustering, y por tanto no se tendrá en cuenta de ahora en adelante.

Después se han eliminado outliers usando Zscore. Esto mejora el agrupamiento realizado por muchos algoritmos que no son robustos ante outliers, como kmeans.

Finalmente se ha normalizado usando normalización minmax. La normalización es muy importante para que no se le de más peso a unas variables que a otras basándose en su rango al calcular la distancia.

‘

2

CASO 1: PERSONAS CON RENTA PROCEDENTE DE UN ALQUILER

2.1 DESCRIPCIÓN DEL CASO DE ESTUDIO

En este caso de estudio se ha elegido el subconjunto con renta neta procedente del alquiler de una propiedad o terreno en el año anterior al de encuesta mayor que o y cualquier régimen de tenencia de la vivienda distinto a cesión gratuita, esto es, en propiedad con o sin hipoteca y en alquiler o realquier en precio normal o inferior al de mercado.

Mediante este caso de estudio se pretende estudiar los gastos, renta e impuestos de las personas que tienen en alquiler algún terreno, para comprobar si la creencia popular sobre si las personas con alquileres tienen mayor poder adquisitivo y más comodidades.

Las variables que se van a seleccionar para realizar el análisis son:

- **Renta**: Renta disponible total del hogar en el año anterior al de encuesta.
- **Gastos**: 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.).
- **Alimentacion in**: Durante el mes pasado, ¿cuál fue aproximadamente el importe que el hogar gastó en alimentos y bebidas no alcohólicas para ser consumidas en casa?
- **IRPF**:Devoluciones/ingresos complementarios por ajustes en impuestos en el año anterior al de encuesta (declaración del IRPF).

Este caso de estudio consta de 2292 respuestas a la encuesta, cada una con las 4 variables indicadas.

2.2 EJECUCIÓN DE ALGORITMOS

Ejecutamos ahora la primera celda del notebook de jupyter (que se puede configurar para sacar las distintas gráficas) y obtenemos así las diferentes medidas obtenidas para cada algoritmo, que se muestran en (1).

Cuadro 1: Caso 1 - Resultados de ejecución de algoritmos.

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
kmeans	0.057	625.999	0.33432	5
birch	0.106	336.711	0.48426	5
spectral	0.646	534.083	0.29880	5
dbscan	0.126	333.456	0.67367	2
meanshift	10.410	73.801	0.35908	32

En primer lugar, nos llama la atención que spectral cluster es el segundo algoritmo que más ha tardado en su ejecucución, pese a que no ha obtenido (a simple vista y basándonos solo en los resultados mostrados en la tabla) mejores resultados que otros algoritmos. El algoritmo más lento ha sido meanshift, lo que puede deberse a tener que calcular el bandwith, que es una operación costosa.

También es destacable como DBSCAN ha generado solamente 2 clusters, donde uno de ellos contiene el 97,21 % de los objetos, y el resto se encuentran en el cluster -1. Esto causa que su silhouette sea mucho mayor que el del resto de algoritmos.

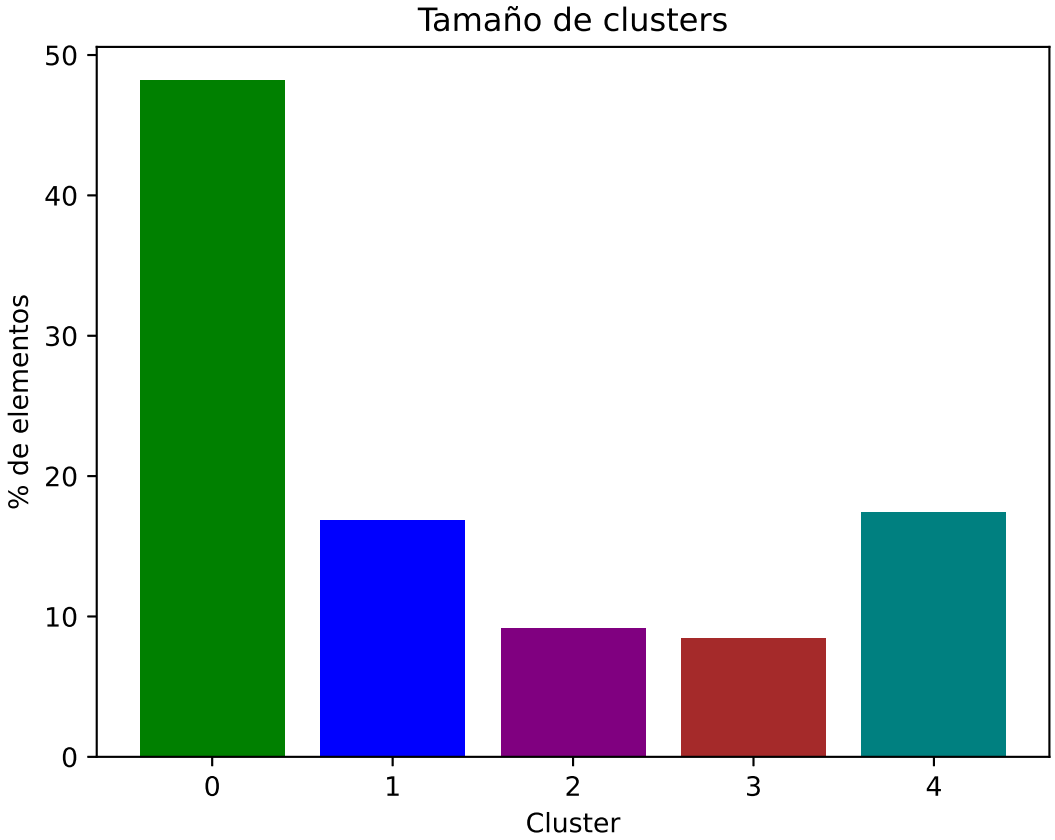
Observamos además que meanshift ha generado 32 clusters y ha obtenido el valor de Calinski-Harabasz más bajo de todos. Observando los tamaños de los clusters que ha generado meanshift vemos que uno contiene el 90,66 % de los datos, muchos de ellos contienen un solo elemento (0,04 % de los datos totales) y ninguno de los restantes supera el 2 % de los datos totales. Podemos concluir por tanto que meanshift no ha hecho un buen agrupamiento.

Los algoritmos kmeans, birch y spectral cluster han obtenido resultados similares, siendo birch el que ha obtenido mejor silhouette, aunque esto puede no ser muy representativo, ya que es una medida muy sensible a la separación de los clusters. Por tanto, nos fijamos de nuevo en Calinski-Harabasz, donde kmeans y spectral cluster han conseguido mejores resultados.

2.3 ANÁLISIS

Para el análisis se va a usar spectral cluster, ya que ha obtenido unos resultados aceptables en las medidas. Este algoritmo genera 5 clusters. uno de tamaño mayor y el resto de tamaños similares, como se puede ver en (1)

Figura 1: Caso 1- Tamaño de los clusters generados por spectral cluster



A continuación observamos la scatter matrix (2), donde se puede apreciar como se separan los clusters y qué variables son necesarias para separar cada uno.

Figura 2: Caso 1- Scatter matrix de spectral cluster

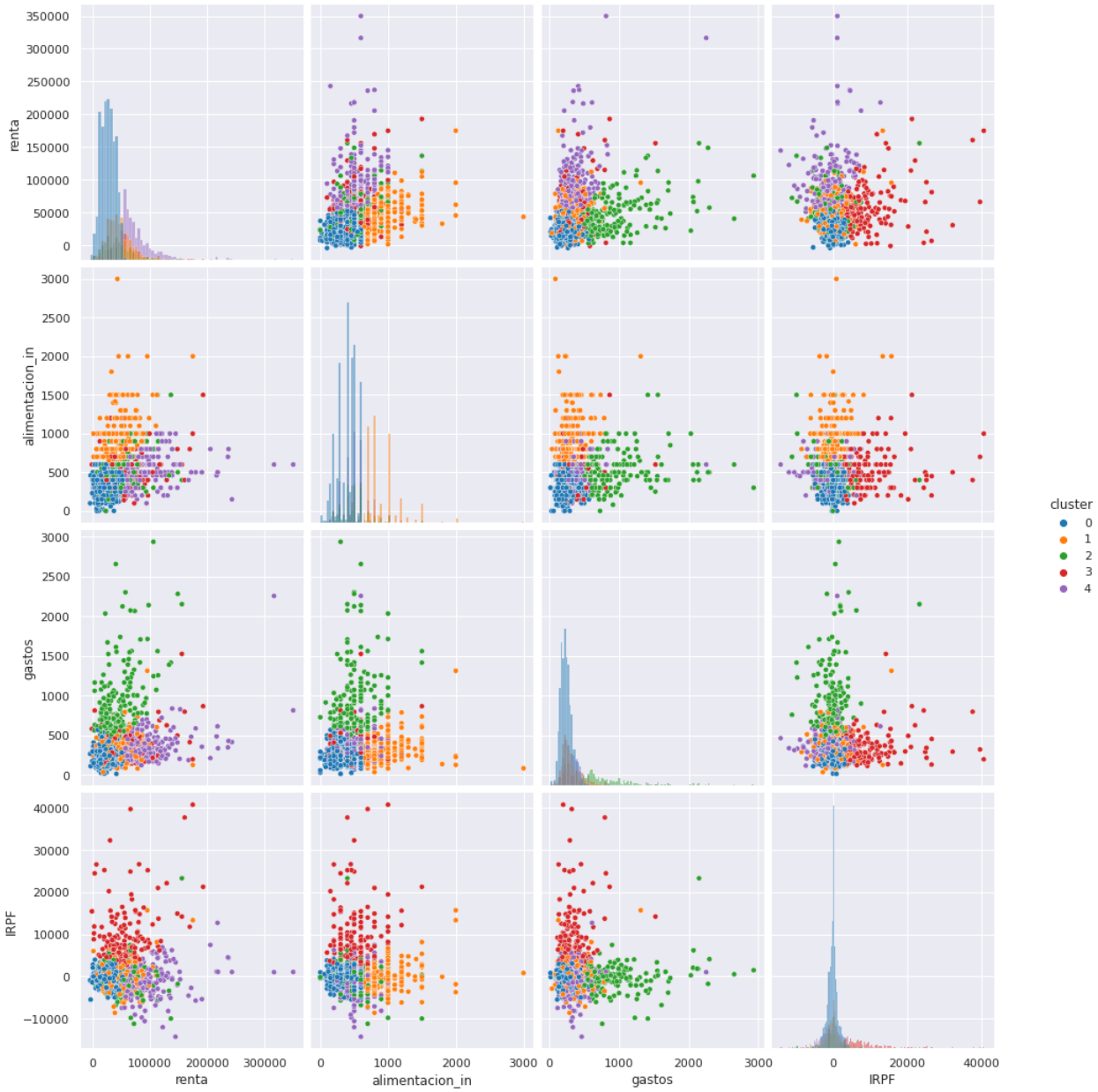
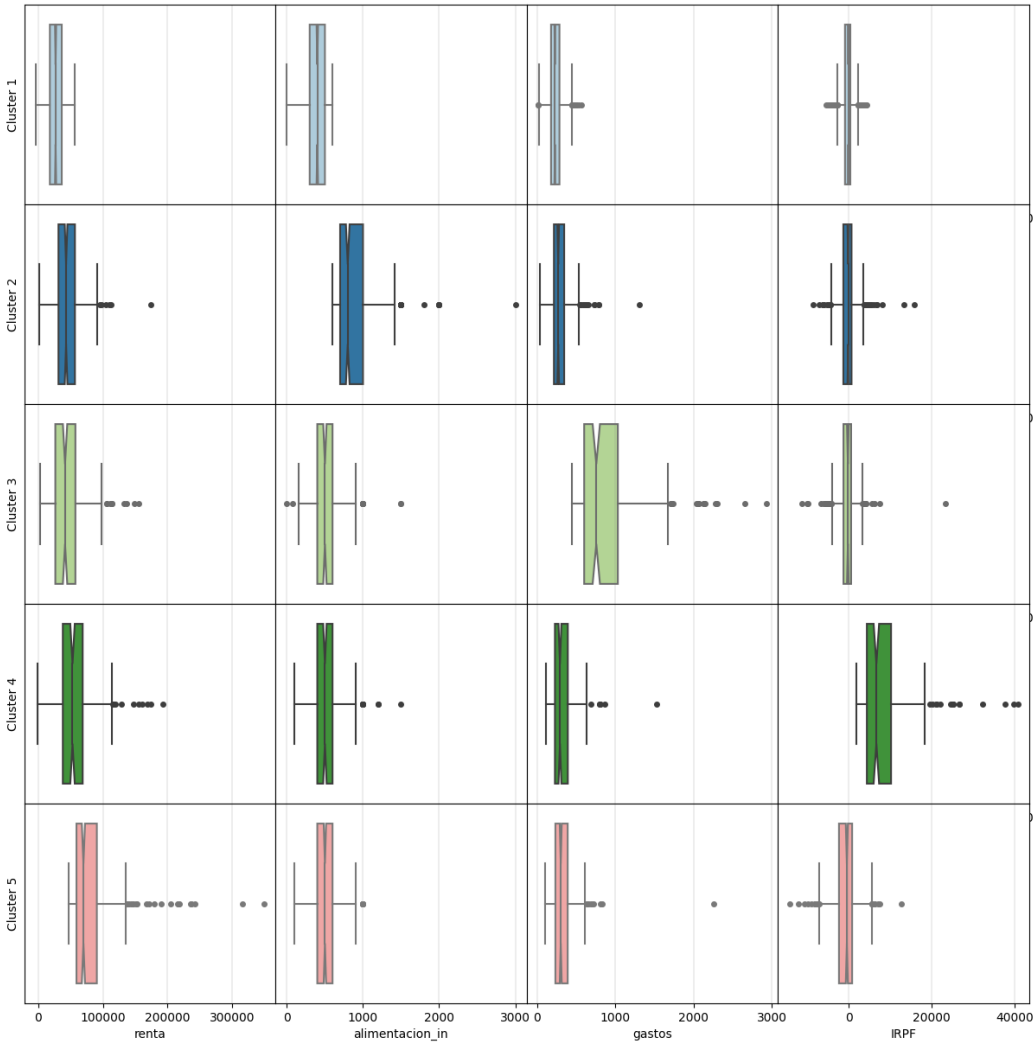


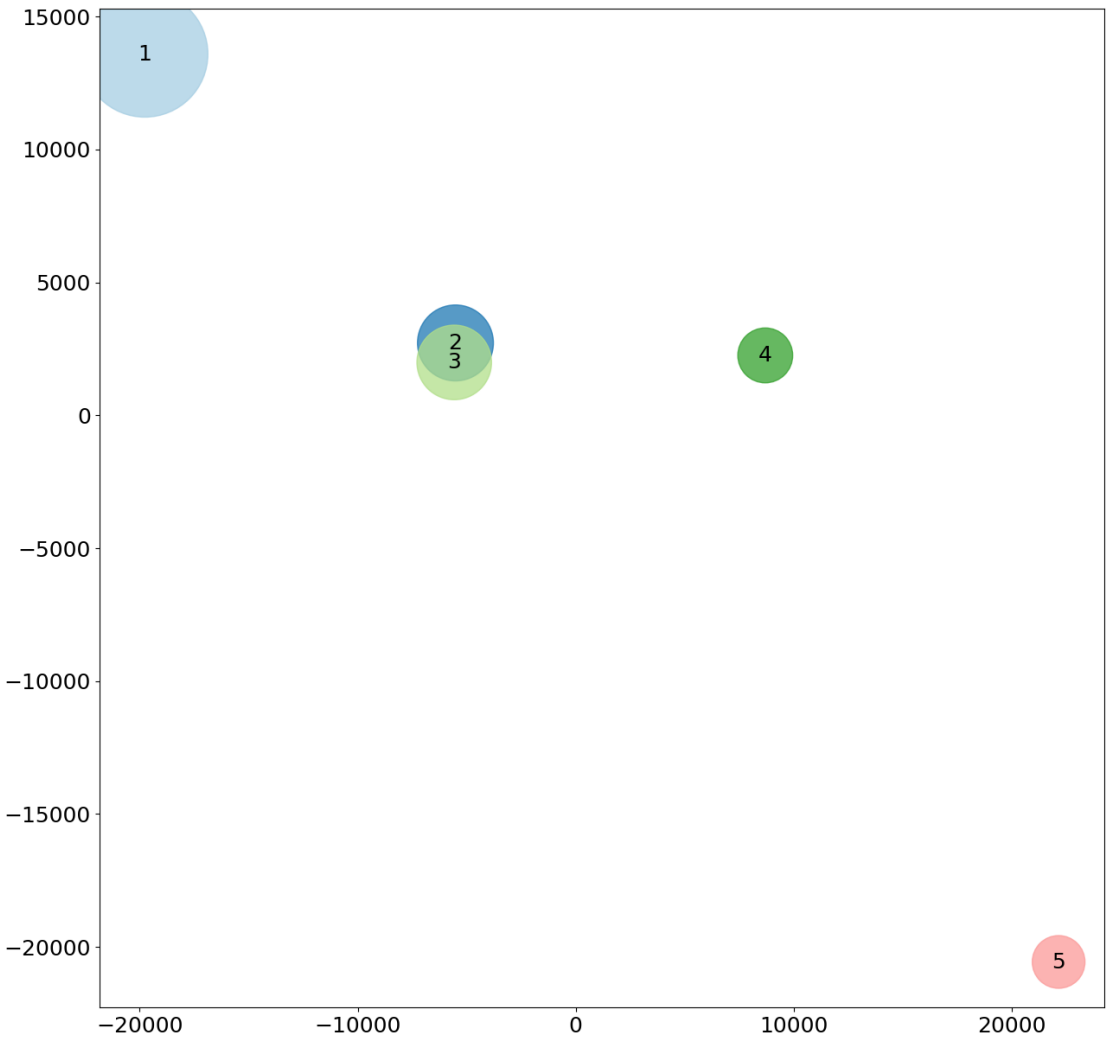
Figura 3: Caso 1- Boxplot de spectral cluster



En (3) podemos observar como el cluster 3 toma valores en general mayores en gastos que los otros clusters. Esto también se observa en el cluster 4 con el IRPF y en el 2 con alimentacion in. Mientras los clusters 1 y 5 son más similares.

Notemos que al representar el boxplot la numeración de los clusters está de 1 a 5, mientras en la scatter matrix es de 0 a 4. Por tanto, hemos obtenido conclusiones análogas a las vistas en la scatter matrix.

Figura 4: Caso 1- MDS de spectral cluster



En (4) se aprecia como los clusters están considerablemente separados salvo dos que se superponen, aunque esto puede deberse a la proyección realizada. De nuevo, los clusters van de 1 a 5, pudiendo observarse desde otra perspectiva los tamaños de los clusters, cuyas proporciones coinciden con las expuestas anteriormente.

En la tabla (2) se muestran qué variables son necesarias para identificar cada cluster:

Cuadro 2: Caso 1 - Variables necesarias para separar el clustering en spectral cluster

Cluster	renta	alimentacion in	gastos	IRPF
0	Baja	Baja		
1		Alto		
2			Alto	
3				Alto
4	Alta	Baja		

Como se aprecia en (2) con una sola variable en cada cluster nos sirve para diferenciarlos entre ellos, mientras para los clusters 0 y 4 necesitamos dos para diferenciarlos entre ellos.

2.4 ESTUDIO DE PARÁMETROS DE DBSCAN

Vamos a analizar el comportamiento de DBSCAN según sus parámetros principales. Para ello ejecutaremos la celda de parámetros de DBSCAN, que nos proporciona las medidas obtenidas para cada par de parámetros fijados.

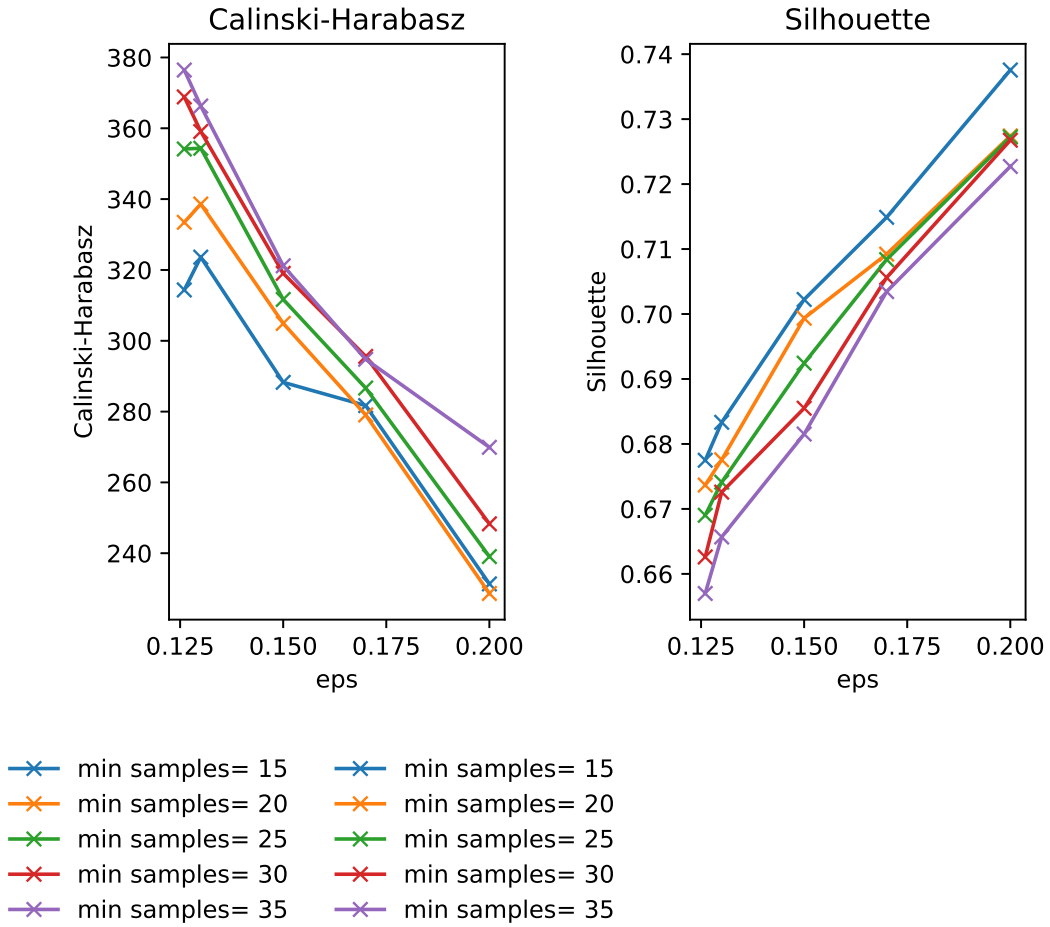
Como podemos apreciar en (3) DBSCAN siempre realiza dos clusters. El tamaño de estos es siempre similar, más del 90 % los ejemplos están agrupados en el cluster 0 y los restantes están en el -1.

Este algoritmo no ha hecho ninguna buena agrupación en ningún caso de los probados, pese a que su silhouette es bastante alto, lo que puede ser consecuencia de que los clusters estén muy separados.

Cuadro 3: Caso 1 - Cambio de parámetros DBSCAN

Algoritmo	eps	min samples	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
dbscan	0.126	15	0.129	314.352	0.67749	2
dbscan	0.126	20	0.125	333.456	0.67367	2
dbscan	0.126	25	0.121	354.162	0.66902	2
dbscan	0.126	30	0.125	368.859	0.66260	2
dbscan	0.126	35	0.128	376.451	0.65698	2
dbscan	0.130	15	0.127	323.611	0.68333	2
dbscan	0.130	20	0.121	338.608	0.67757	2
dbscan	0.130	25	0.130	354.371	0.67411	2
dbscan	0.130	30	0.123	359.105	0.67258	2
dbscan	0.130	35	0.124	366.292	0.66568	2
dbscan	0.150	15	0.135	288.264	0.70224	2
dbscan	0.150	20	0.134	304.901	0.69936	2
dbscan	0.150	25	0.132	311.673	0.69243	2
dbscan	0.150	30	0.132	319.038	0.68550	2
dbscan	0.150	35	0.132	321.172	0.68152	2
dbscan	0.170	15	0.143	281.696	0.71489	2
dbscan	0.170	20	0.137	279.062	0.70922	2
dbscan	0.170	25	0.139	286.660	0.70839	2
dbscan	0.170	30	0.138	295.586	0.70562	2
dbscan	0.170	35	0.145	294.783	0.70345	2
dbscan	0.200	15	0.145	231.341	0.73756	2
dbscan	0.200	20	0.143	228.641	0.72741	2
dbscan	0.200	25	0.144	239.060	0.72724	2
dbscan	0.200	30	0.141	248.309	0.72676	2
dbscan	0.200	35	0.141	269.916	0.72272	2

Figura 5: Caso 1- Parámetros de DBSCAN



No podemos interpretar demasiado Calinsk-Harabasz, ya que al no estar normalizado no sabemos si los valores dados son muy altos. Sí podemos decir que los mejores agrupamientos (que han proporcionado un valor de esta medida más alto) se han dado cuando min samples es 35, como se puede ver en (5).

También cabe destacar como el silhouette va creciendo conforme aumenta epsilon y calinski sube ligeramente al principio para valores bajos de min samples pero luego decrece, mientras para valores altos solo decrece. Asimismo resulta llamativo como los valores de min samples que producen valores de Calinski-harabasz más altos producen silhouettes más bajos.

Finalmente, cabe destacar como el tiempo de ejecución se incrementaba a la par que crecían los parámetros.

2.5 ESTUDIO DE PARÁMETROS DE BIRCH

Estudiamos ahora el algoritmo Birch. Para este de nuevo se han probado distintas combinaciones de parámetros.

Cuadro 4: Caso 1 - Cambio de parámetros Birch

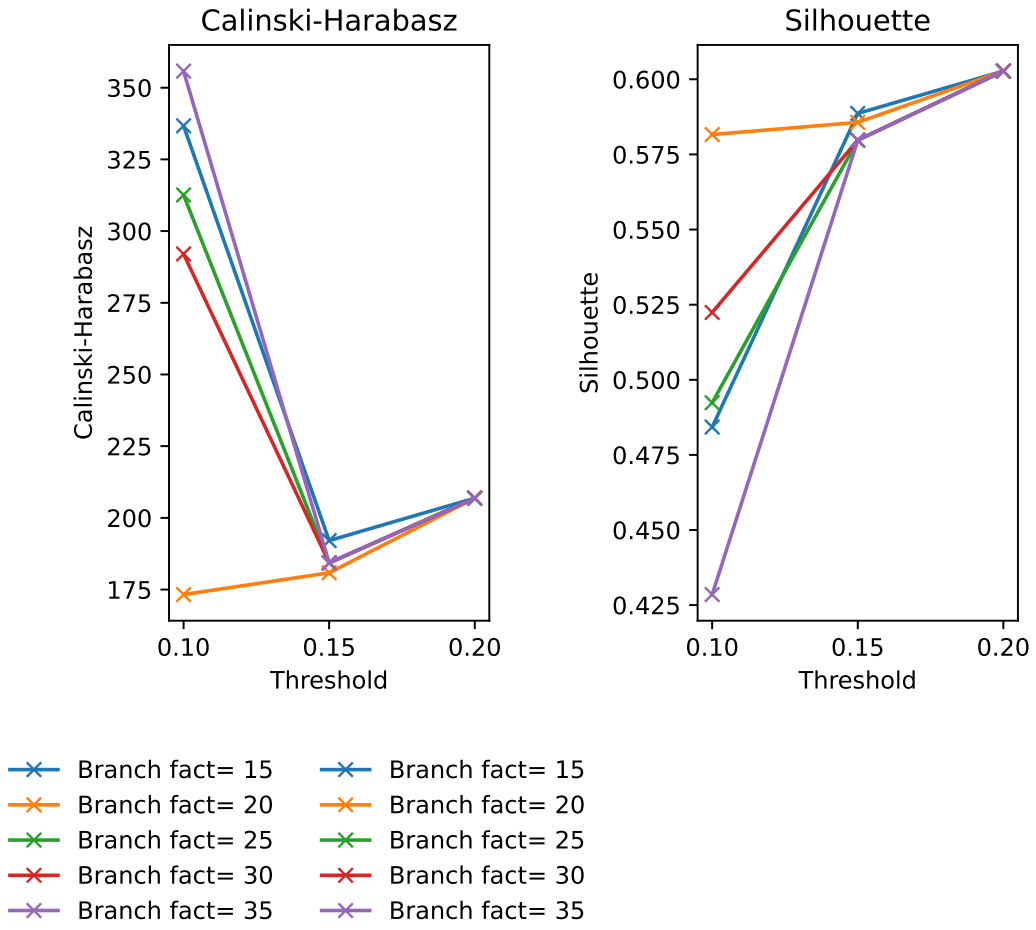
Algoritmo	Branch fact	Threshold	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
birch	15	0.100	0.087	336.711	0.48426	5
birch	15	0.150	0.065	192.073	0.58865	5
birch	15	0.200	0.060	206.873	0.60273	5
birch	15	0.250	0.062			1
birch	15	0.300	0.049			1
birch	20	0.100	0.147	173.266	0.58162	5
birch	20	0.150	0.077	180.842	0.58567	5
birch	20	0.200	0.046	206.873	0.60273	5
birch	20	0.250	0.053			1
birch	20	0.300	0.052			1
birch	25	0.100	0.153	312.603	0.49236	5
birch	25	0.150	0.064	184.304	0.57974	5
birch	25	0.200	0.045	206.873	0.60273	5
birch	25	0.250	0.051			1
birch	25	0.300	0.043			1
birch	30	0.100	0.136	292.036	0.52240	5
birch	30	0.150	0.059	184.304	0.57974	5
birch	30	0.200	0.066	206.873	0.60273	5
birch	30	0.250	0.057			1
birch	30	0.300	0.074			1
birch	35	0.100	0.129	355.783	0.42852	5
birch	35	0.150	0.058	184.304	0.57974	5
birch	35	0.200	0.061	206.873	0.60273	5
birch	35	0.250	0.061			1
birch	35	0.300	0.054			1

Podemos ver en (4) como el algoritmo en general agrupa en 5 clusters y obtiene resultados similares a kmeans. En general, ha hecho buenas agrupaciones a juzgar por las medidas.

Sin embargo, si miramos el tamaño de los clusters en muchas ocasiones estos se distriubuyen con un solo cluster agrupando el 90 % o más de los datos y los 4 restantes el resto.

Cabe destacar que es bastante sensible a los parámetros, pues variando solo uno de ellos en ocasiones pasa de hacer 5 clusters a solo 1. Además, en estos casos no se proporcionan valores para las medidas Calinski-Harabasz y Silhouette, ya que para realizar estas es necesario tener más de un cluster.

Figura 6: Caso 1- Parámetros de Birch



En (6) se aprecia como para los valores de branch factor que producen mayor calinski-harabasz el silhouette es menor, y viceversa. Además, el silhouette siempre crece, mientras que Calinski-harabasz primero decrece y luego crece.

Teniendo en cuenta que Birch es una algoritmo que no está pensado para realizar clustering obteniendo todos los datos de una misma vez, ha generado un número de clusters razonable, pese a ser estas agrupaciones dudosas, pues los tamaños de estas son demasiado dispares.

2.6 INTERPRETACIÓN DE LA SEGMENTACIÓN

A la vista de los resultados obtenidos, vemos que no se aprecian diferencias significativas en la renta entre las agrupaciones y no se encuentra demasiado dispersa. Esta variable por tanto no es tan relevante como otras.

No hay ninguna variable que destaque sobre otra, pero usando solamente las variables gastos, IRPF y alimentacion in podríamos diferenciar con claridad 3 clusters, y los otros dos agruparlos por descarte. Las otras variables nos sirvieron para refinar la diferenciación entre los clusters para lograr diferenciar los 5.

Respecto a la hipótesis inicial, que las personas con propiedades alquiladas tienen mayor renta y por tanto más comodidades, en principio no parece cierta, pues la renta no ha sido una variable significativa respecto a la que dividir y otros factores son más influyentes.

3

CASO 2: COMPARACIÓN DE RENTAS

3.1 DESCRIPCIÓN DEL CASO DE ESTUDIO

Este caso de estudio trata de contrastar las diferencias que hay entre grupos con renta baja (menor de 15000€ al año) y renta alta (mayor de 25000€ al año).

Para ello, las variables seleccionadas son:

- **Renta**: Renta disponible total del hogar en el año anterior al de encuesta.
- **Alimentos**: Durante el mes pasado, ¿cuál fue aproximadamente el importe que el hogar gastó en alimentos y bebidas no alcohólicas para ser consumidas en casa?
- **Asistencia social**: Ingresos por asistencia social en el año anterior al de encuesta.
- **Gastos**: 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.)

Para realizar la comparativa, vamos a estudiar dos subcasos, el primero con renta baja y el segundo con renta alta.

3.2 CASO 2 A: RENTA BAJA

Este caso de estudio consta de 3197 elementos, cada uno con las cuatro variables especificadas antes.

3.2.1 *Ejecución de algoritmos*

Ejecutamos la celda del notebook de jupyter correspondiente al caso 2A (que se puede configurar para sacar las distintas gráficas) y obtenemos así las diferentes medidas obtenidas para cada algoritmo, que se muestran en (5).

Cuadro 5: Caso 2A - Resultados de ejecución de algoritmos.

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
kmeans	0.385	1157.588	0.31325	5
birch	0.124	510.565	0.26449	5
spectral	1.739	1068.474	0.30286	5
dbscan	0.171	304.282	0.60850	2
meanshift	18.233	184.067	0.40807	11

Respecto a los tiempos, de nuevo el algoritmo más lento es meanshift, seguido de spectral cluster. El resto de algoritmos obtienen tiempo similares.

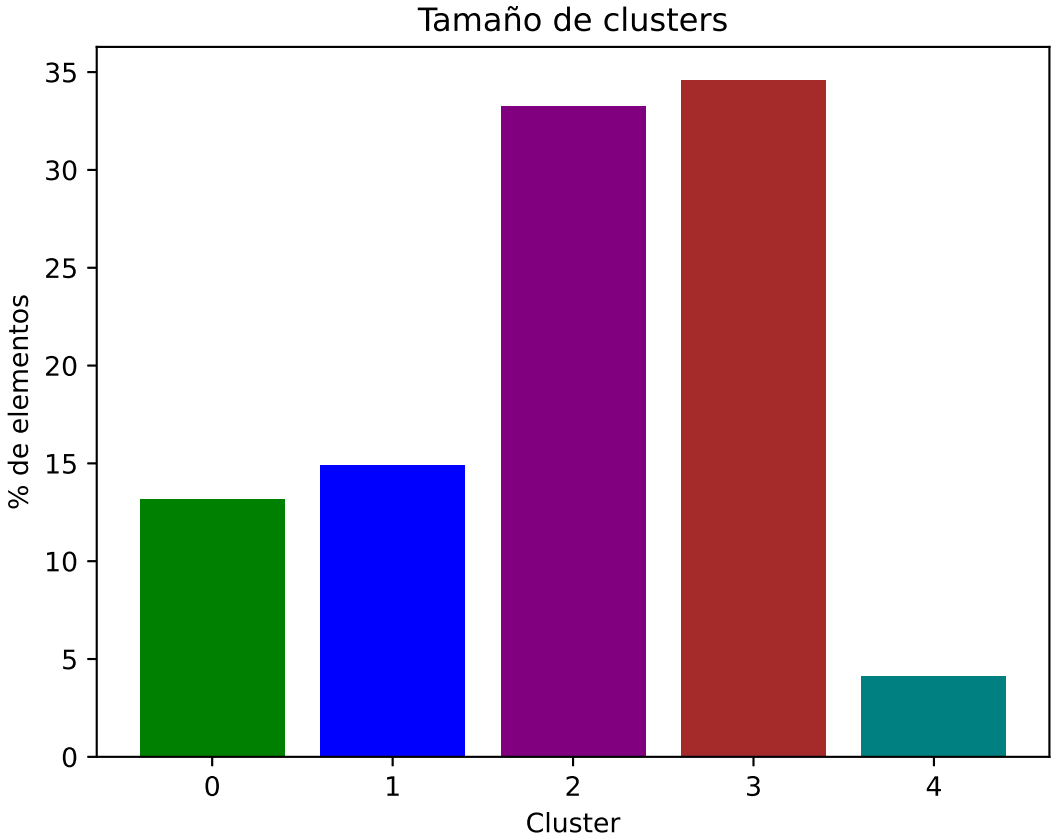
Además, al igual que en el caso 1 DBSCAN ha generado solamente dos clusters, uno con el 97,12 % de los datos y los restantes en el cluster -1. Meanshift vuelve a generar más cluster que el resto, agrupando uno de ellos al 91,46 % de elementos, otro al 4,86 % y el resto cada uno contiene un número de elementos menor al 1,2 % del total. Por ello, parece que a pesar de las medidas que han obtenido, estos algoritmos no han realizado un buen agrupamiento.

Finalmente vemos como Birch tiene un valor de Calinski-Harabasz y silhouette más bajo que kmeans y spectral cluster. Estos algoritmos han conseguido medidas similares, obteniendo en principio kmeans mejores resultados.

3.2.2 *Análisis*

Para el análisis se va a usar kmeans, ya que ha obtenido las mejores medidas. Este algoritmo genera 5 clusters. Se ha generado un cluster de tamaño menor, y los otros tienen tamaños similares dos a dos, como se puede ver en (7).

Figura 7: Caso 2A- Tamaño de los clusters generados por kmeans



A continuación observamos la scatter matrix (8), donde se puede apreciar como se separan los clusters y qué variables son necesarias para diferenciar cada uno. Hay varios clusters que se diferencian bien con una sola variable, como es el caso de cluster 0, que tiene valores altos de gastos, el 4 tiene valores más altos en asistencia social y el 2 gasta más en alimentación. Los clusters 1 y 3 se distinguen bien usando renta y gastos o renta y alimentos, ya que ambos tienen valores bajos de renta y uno tiene valores mayores que el otro en la variable restante.

Figura 8: Caso 2A- Scatter matrix de kmeans

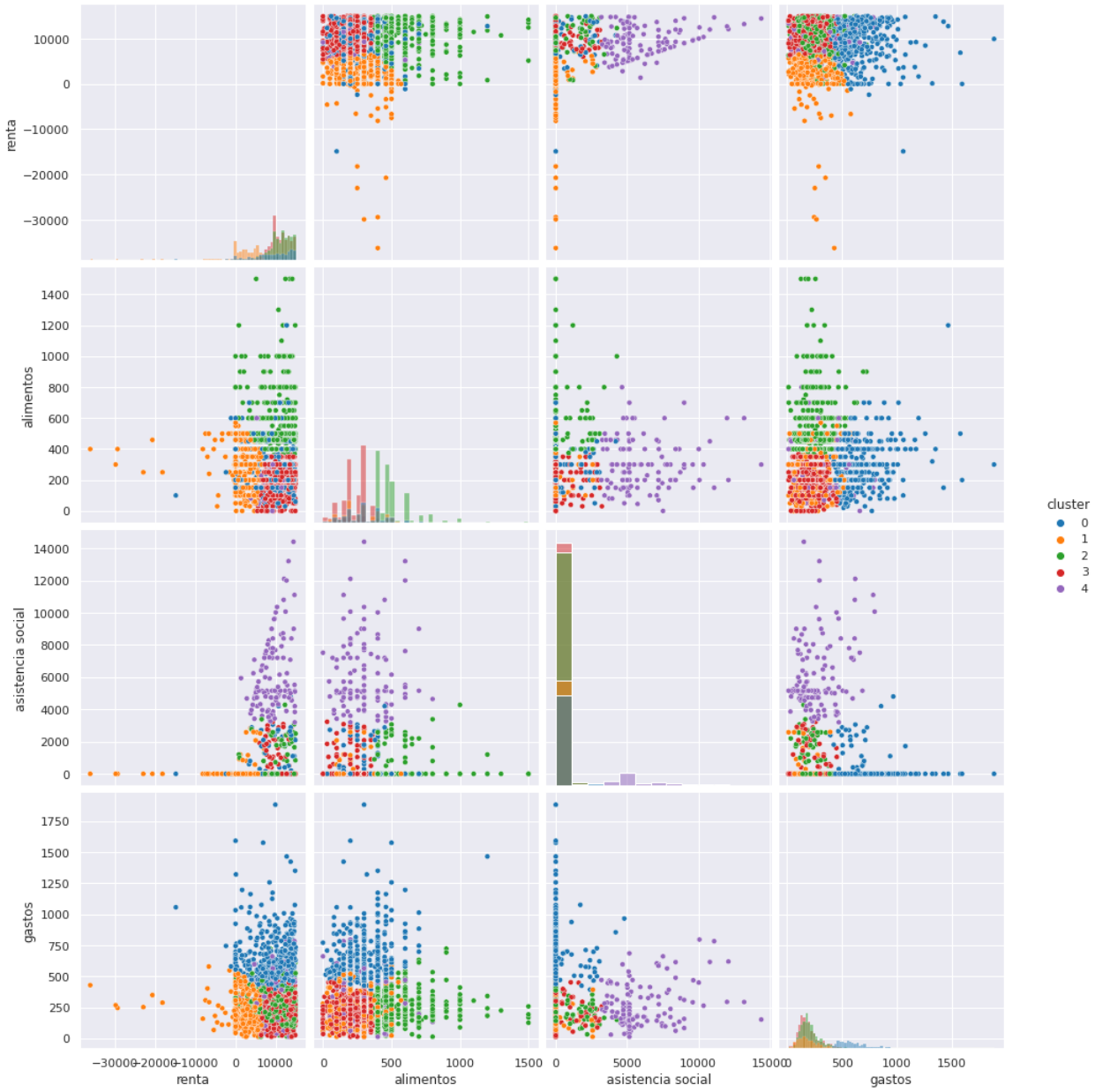
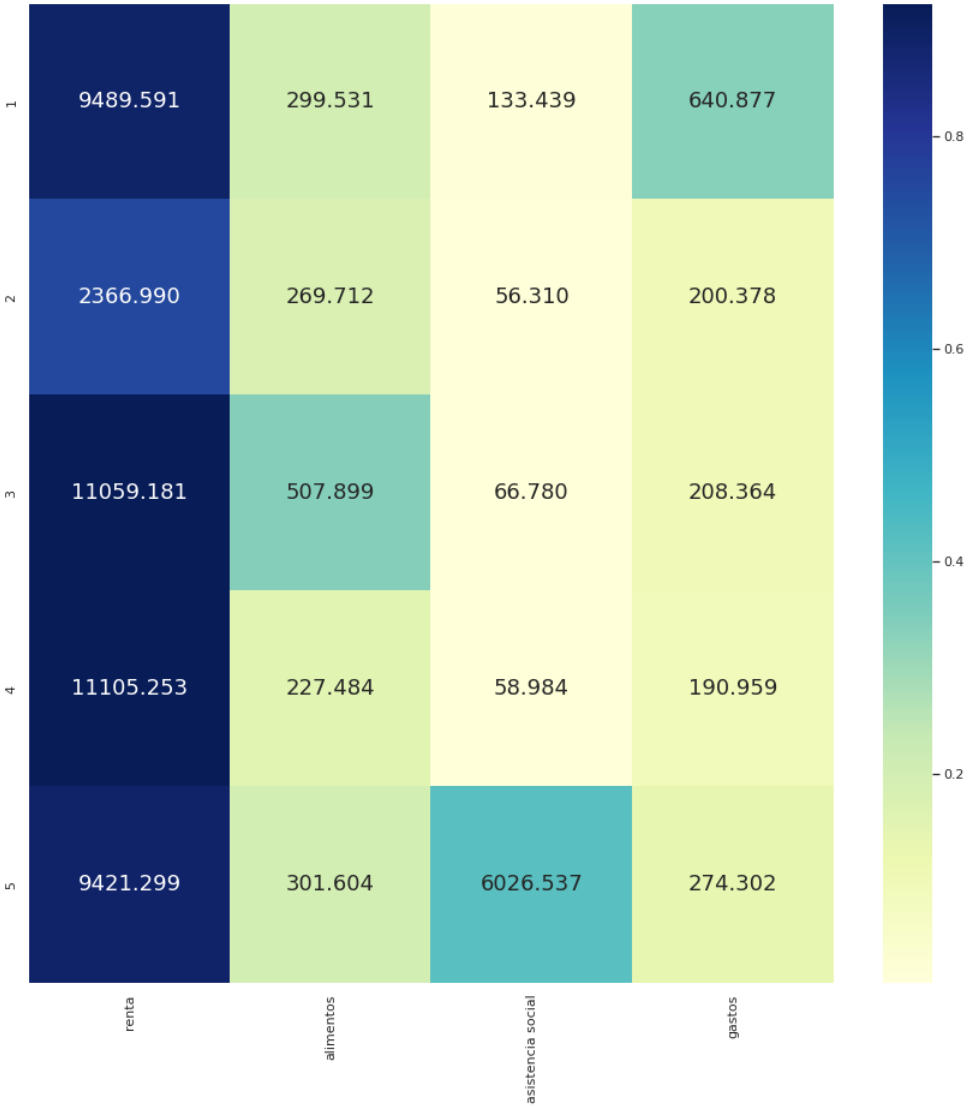
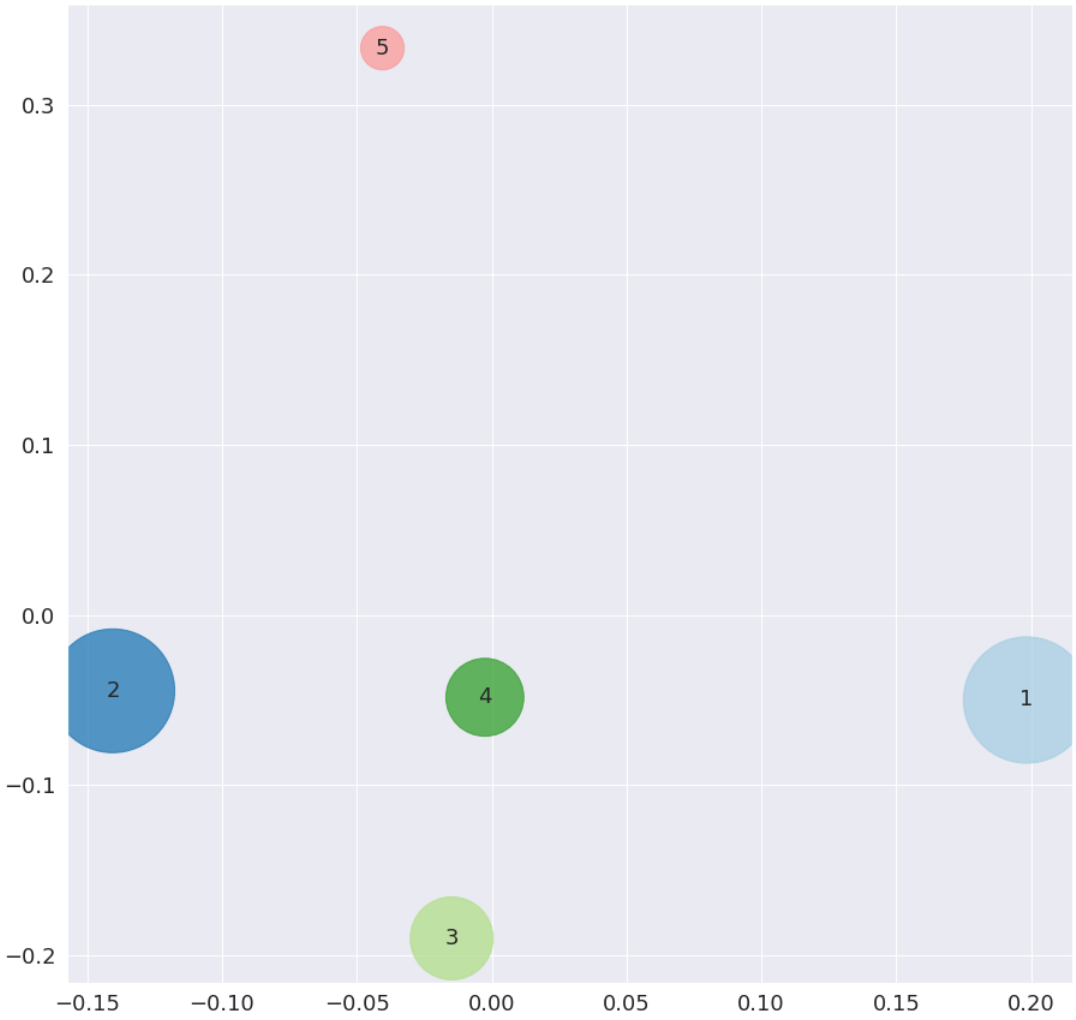


Figura 9: Caso 2A- Heatmap de kmeans



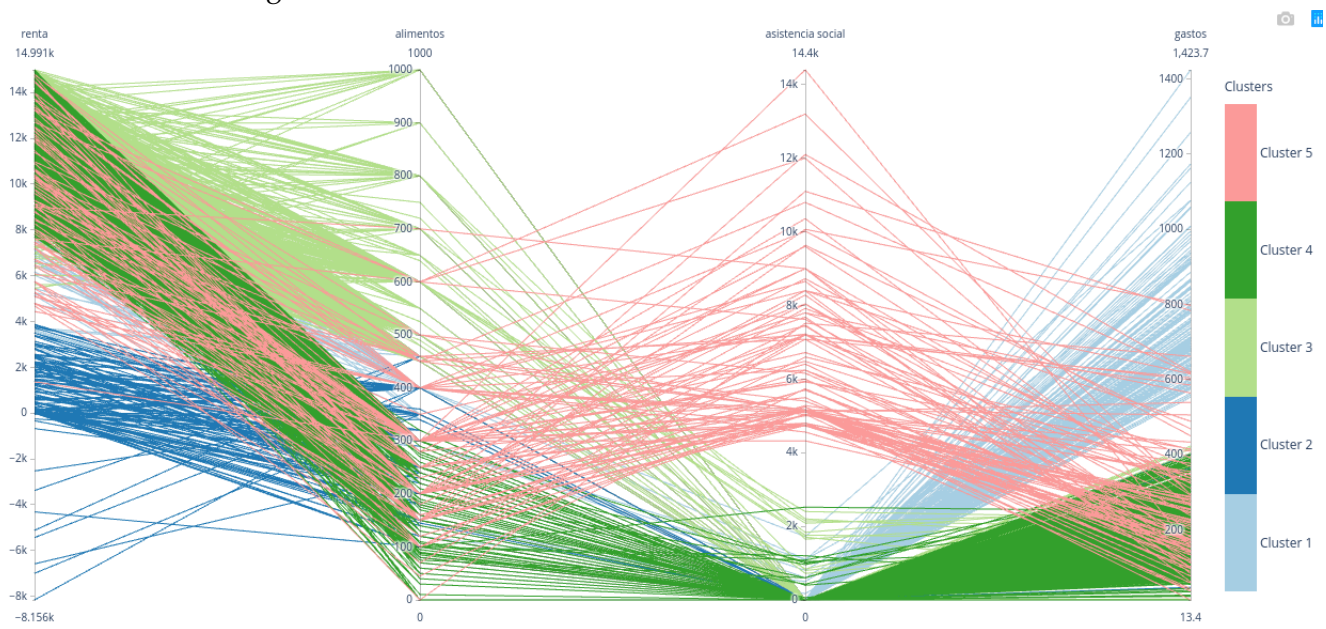
En (9) se ve con claridad los tres clusters que se diferencian por tomar valores especialmente altos en una de sus variables. Teniendo en cuenta que la numeración en este gráfico va de 1 a 5 y en la scatter matrix va de 0 a 4, vemos la analogía. En el heatmap se aprecia como el cluster 5 toma valores considerablemente más altos en asistencia social que el resto de clusters, el 3 toma valores más altos en alimentación y el 1 en gastos, lo que concuerda con lo visto en la scatter matrix.

Figura 10: Caso 2A- MDS de kmeans



En (10) se observa como los clusters están separados entre sí y tienen tamaños similares salvo por el cluster 5, que es de menor tamaño, como ya vimos al analizar los tamaños antes.

Figura 11: Caso 2A - Parallel coordinates de kmeans



En (11) se aprecia como el cluster rosa, que alcanza mayores valores de asistencia social, es más disperso que los demás, lo que concuerda con la scatter matrix (8). De nuevo, es claro que hay dos clusters que se diferencian por sus valores altos en alimentación y gastos.

En la tabla (6) se muestran qué variables son necesarias para identificar cada cluster:

Cuadro 6: Caso 2A - Variables necesarias para separar el clustering en spectral cluster

Cluster	renta	alimentos	asistencia social	gastos
0				Altos
1	Baja	Bajos		Bajos
2		Altos		
3	Alta	Bajos		Bajos
4			Altos	

Como se aprecia en (6) con una sola variable en cada cluster nos sirve para diferenciarlos entre los clusters 0, 2 y 4, mientras para los clusters 1 y 3 necesitamos dos para diferenciarlos entre ellos.

3.2.3 *Estudio de parámetros de DBSCAN*

Vamos a analizar el comportamiento de DBSCAN según sus parámetros principales. Para ello ejecutaremos la celda de parámetros de DBSCAN, que nos proporciona las medidas obtenidas para cada par de parámetros.

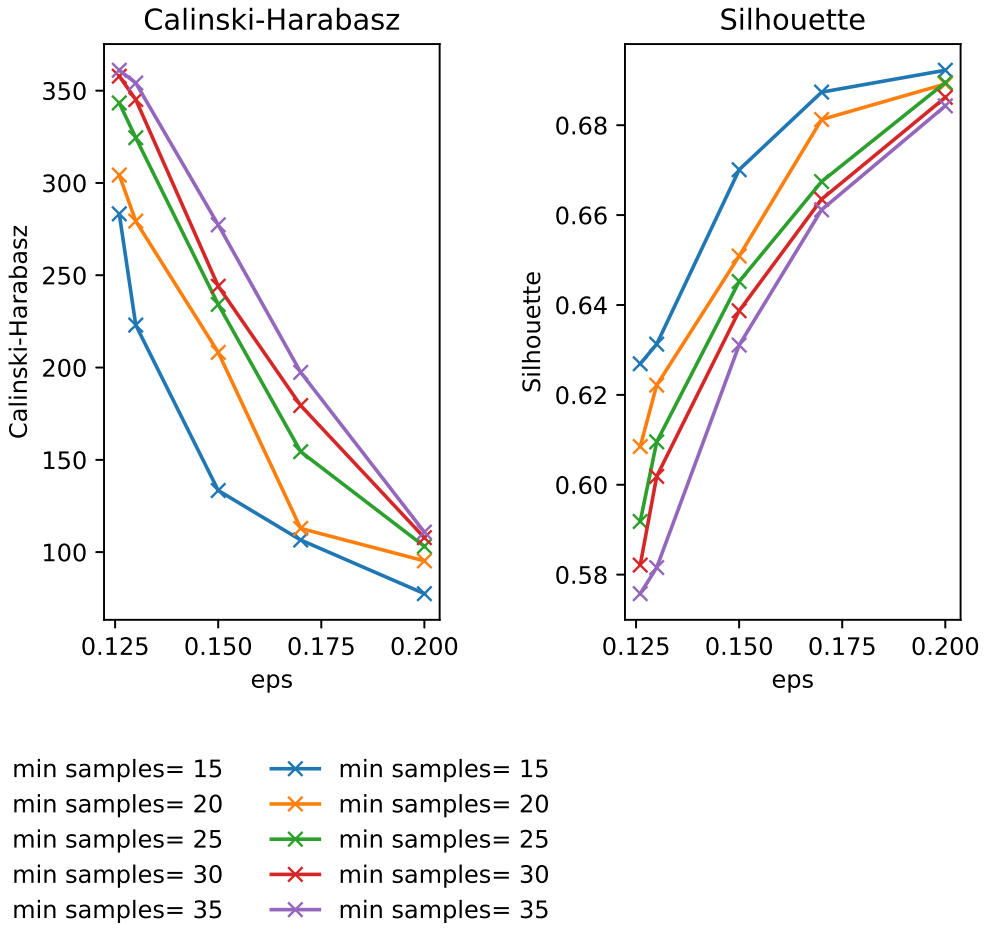
Cuadro 7: Caso 2A - Cambio de parámetros DBSCAN

Algoritmo	eps	min samples	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
dbscan	0.126	15	0.210	283.280	0.62692	2
dbscan	0.126	20	0.177	304.282	0.60850	2
dbscan	0.126	25	0.185	343.349	0.59182	2
dbscan	0.126	30	0.181	357.820	0.58212	2
dbscan	0.126	35	0.188	361.064	0.57576	2
dbscan	0.130	15	0.190	223.006	0.63128	2
dbscan	0.130	20	0.194	279.313	0.62217	2
dbscan	0.130	25	0.184	324.483	0.60954	2
dbscan	0.130	30	0.193	345.157	0.60186	2
dbscan	0.130	35	0.193	354.160	0.58156	2
dbscan	0.150	15	0.247	133.391	0.67010	2
dbscan	0.150	20	0.212	208.220	0.65089	2
dbscan	0.150	25	0.213	234.186	0.64523	2
dbscan	0.150	30	0.219	244.053	0.63871	2
dbscan	0.150	35	0.232	277.293	0.63108	2
dbscan	0.170	15	0.245	106.562	0.68737	2
dbscan	0.170	20	0.243	112.836	0.68127	2
dbscan	0.170	25	0.241	154.421	0.66746	2
dbscan	0.170	30	0.231	179.415	0.66353	2
dbscan	0.170	35	0.479	197.375	0.66118	2
dbscan	0.200	15	0.283	77.501	0.69224	2
dbscan	0.200	20	0.256	95.258	0.68918	2
dbscan	0.200	25	0.254	103.127	0.68949	2
dbscan	0.200	30	0.252	107.818	0.68620	2
dbscan	0.200	35	0.239	110.824	0.68431	2

Como podemos apreciar en (7) DBSCAN siempre realiza dos clusters. El tamaño de estos es siempre similar, más del 90 % los ejemplos están agrupados en el cluster 0 y los restantes están en el -1.

Nuevamente este algoritmo no ha hecho ninguna buena agrupación en ningún caso de los probados, pese a que su silhouette es bastante alto, lo que puede ser consecuencia de que los clusters estén muy separados.

Figura 12: Caso 2A- Parámetros de DBSCAN



Respecto a los valores obtenidos de Calinski-Harabasz se puede ver en (12) como, de nuevo, a mayor valor de silhouette menor de Calinski-harabasz, donde este es decreciente y el silhouette creciente. Si quisiéramos maximizar silhouette, tomaríamos min samples como 15 y el mayor valor de epsilon, mientras para maximizar Calinski-Harabasz es mejor el valor 35 de min samples con el menor valor de epsilon.

3.2.4 *Estudio de parámetros de Kmeans*

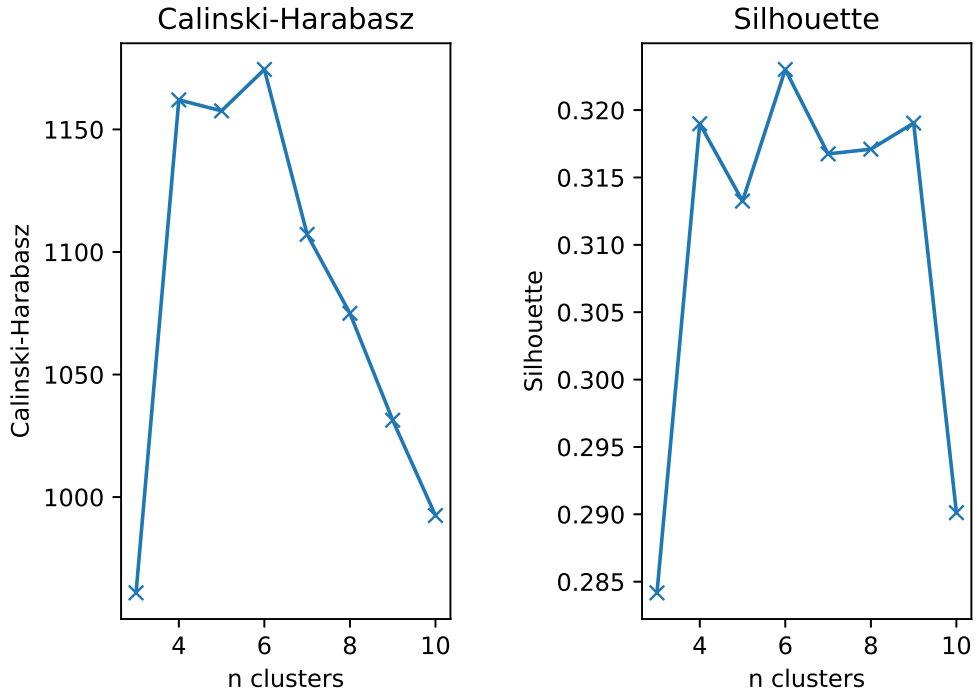
Estudiamos ahora el algoritmo Kmeans. Como parámetro se ha variado el número de clusters.

Cuadro 8: Caso 2A - Cambio de parámetros Kmeans

Algoritmo	n clusters	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
kmeans	3	0.224	960.927	0.28417	3
kmeans	4	0.122	1162.049	0.31899	4
kmeans	5	0.283	1157.588	0.31325	5
kmeans	6	0.292	1174.483	0.32302	6
kmeans	7	0.328	1107.187	0.31675	7
kmeans	8	0.183	1074.974	0.31710	8
kmeans	9	0.487	1031.378	0.31904	9
kmeans	10	1.043	992.485	0.29012	10

Podemos ver en (8) que no hay demasiada diferencia entre las medidas obtenidas para los distintos valores del número de clusters.

Figura 13: Caso 2A- Parámetros de Kmeans



En (13) destaca que el valor para el que se alcanza el máximo en Silhouette es el mismo para el que se alcanza el máximo en Calinski-harabasz, que es 6 clusters. Por tanto, este parece el mejor agrupamiento a realizar.

Aún así, kmeans ha dado resultados buenos en todos los casos y ha demostrado ser bastante robusto en sus agrupaciones.

3.3 CASO 2 B: RENTA ALTA

Este caso de estudio consta de 8372 elementos, cada uno con las cuatro variables especificadas antes.

3.3.1 *Ejecución de algoritmos*

Ejecutamos la celda del notebook de jupyter correspondiente al caso 2B y obtenemos las diferentes medidas obtenidas para cada algoritmo, que se muestran en (9).

Cuadro 9: Caso 2B - Resultados de ejecución de algoritmos.

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
kmeans	0.531	2962.319	0.30911	5
birch	0.312	645.084	0.47807	5
spectral	20.642	2145.100	0.32184	5
dbscan	1.532	473.629	0.75774	2
meanshift	68.273	211.786	0.40726	31

Respecto a los tiempos, de nuevo el algoritmo más lento es meanshift, seguido de spectral cluster. El resto de algoritmos obtienen tiempo similares. Cabe destacar que al incrementarse el número de instancias los tiempo han subido considerablemente, en especial el de meanshift, ya que calcular bandwith es costoso.

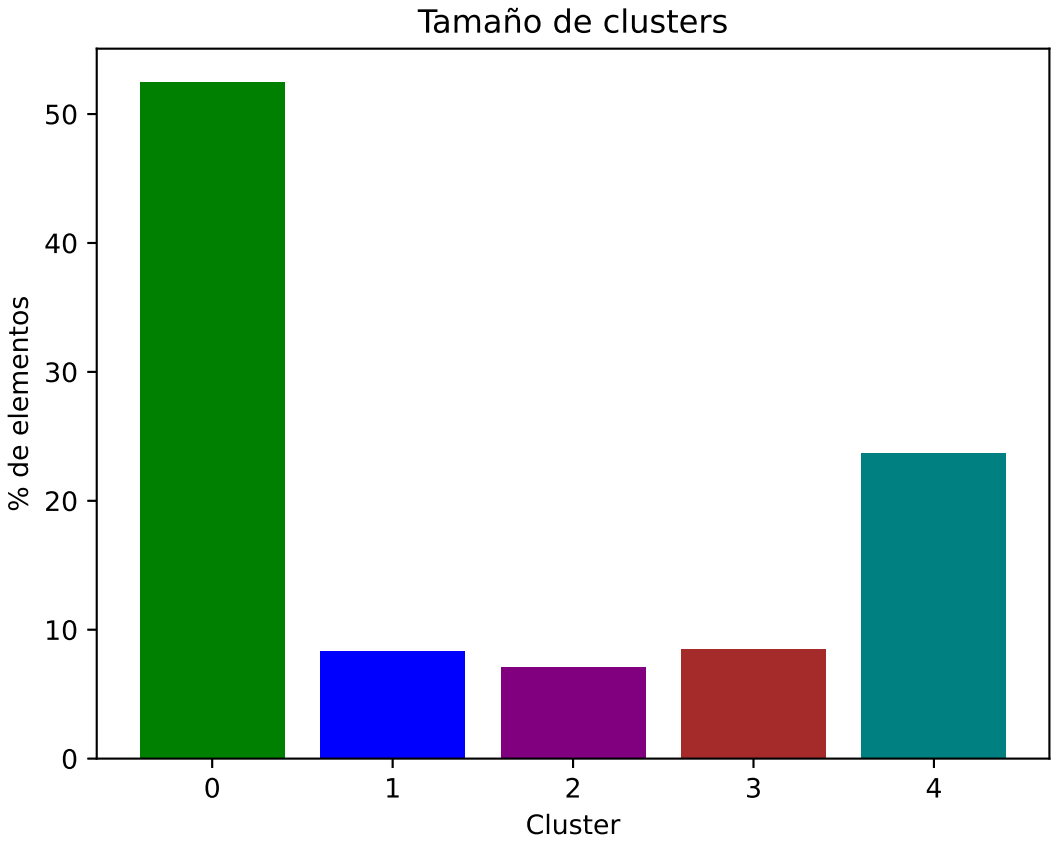
Además, al igual que en los casos anteriores DBSCAN ha generado solamente dos clusters, uno con el 99,44 % de los datos y los restantes en el cluster -1. Meanshift vuelve a generar más cluster que el resto, agrupando uno de ellos el 93,37 % de elementos, y el resto son agrupaciones de menos el 2 % de elementos. Por ello, parece que a pesar de las medidas que han obtenido, estos algoritmos no han realizado un buen agrupamiento.

Finalmente vemos como Birch tiene un valor de Calinski-Harabasz más bajo que kmeans y spectral cluster, aunque obtiene un silhouette mayor.

3.3.2 *Análisis*

Para el análisis se va a usar kmeans, como se hizo en el caso anterior para poder realizar la comparación. Este algoritmo genera 5 clusters. Se ha generado un cluster grande con algo más de la mitad de los datos, uno mediano y el resto pequeños, como se puede ver en (14).

Figura 14: Caso 2B- Tamaño de los clusters generados por kmeans



A continuación observamos la scatter matrix (15), donde se puede apreciar como se separan os clusters y qué variables son necesarias para separar cada uno. De manera análoga al caso 2A vemos como se pueden separar varios clusters solamente por valores altos de ciertas variables, que son el cluster 1 con grandes gastos, el 2 con alimentos y el 3 por valores muy altos de renta. Los clusters 0 y 4 requieren de varias variables para distinguirse, que son alimentos y gastos.

Figura 15: Caso 2B- Scatter matrix de kmeans

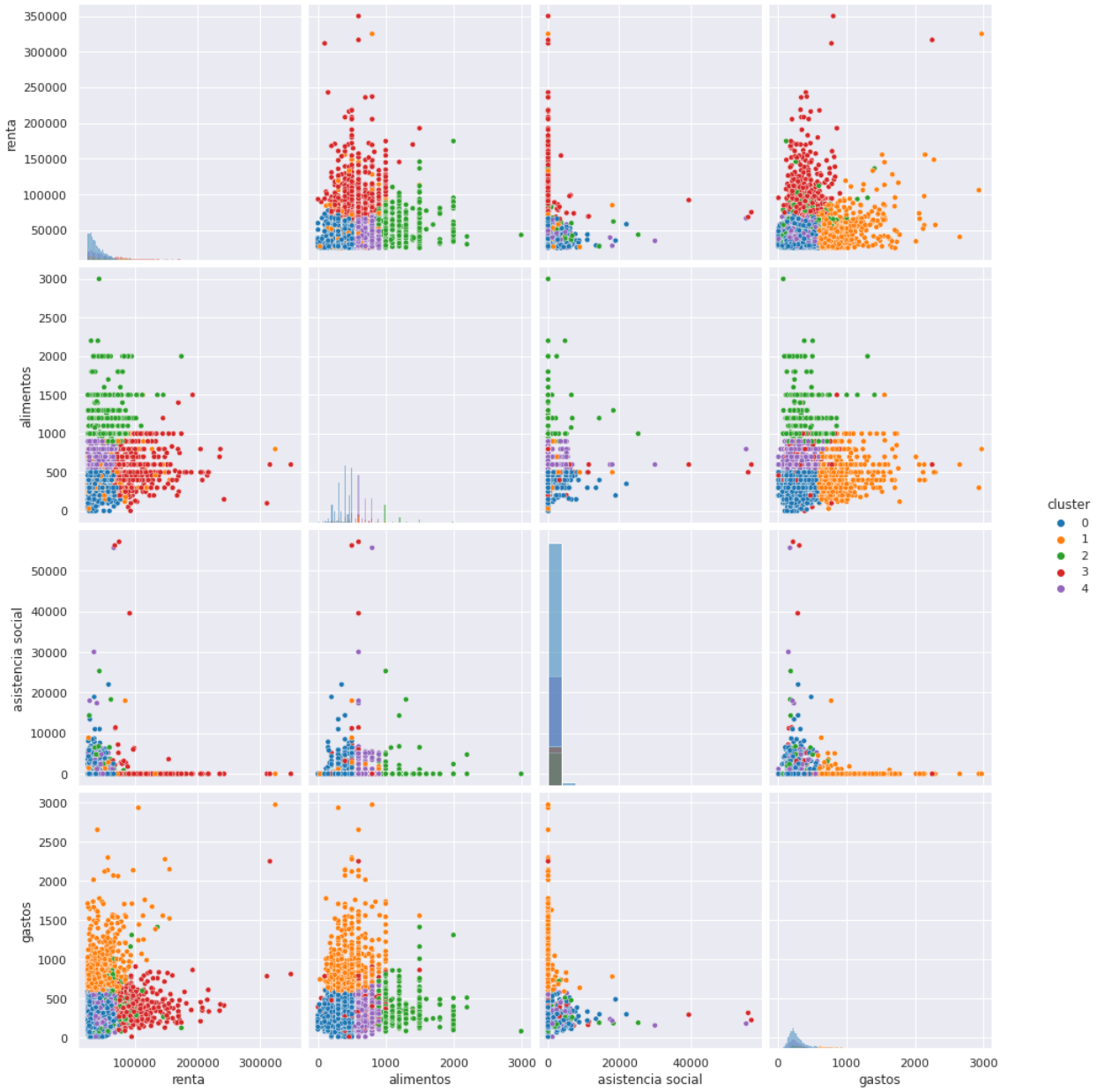
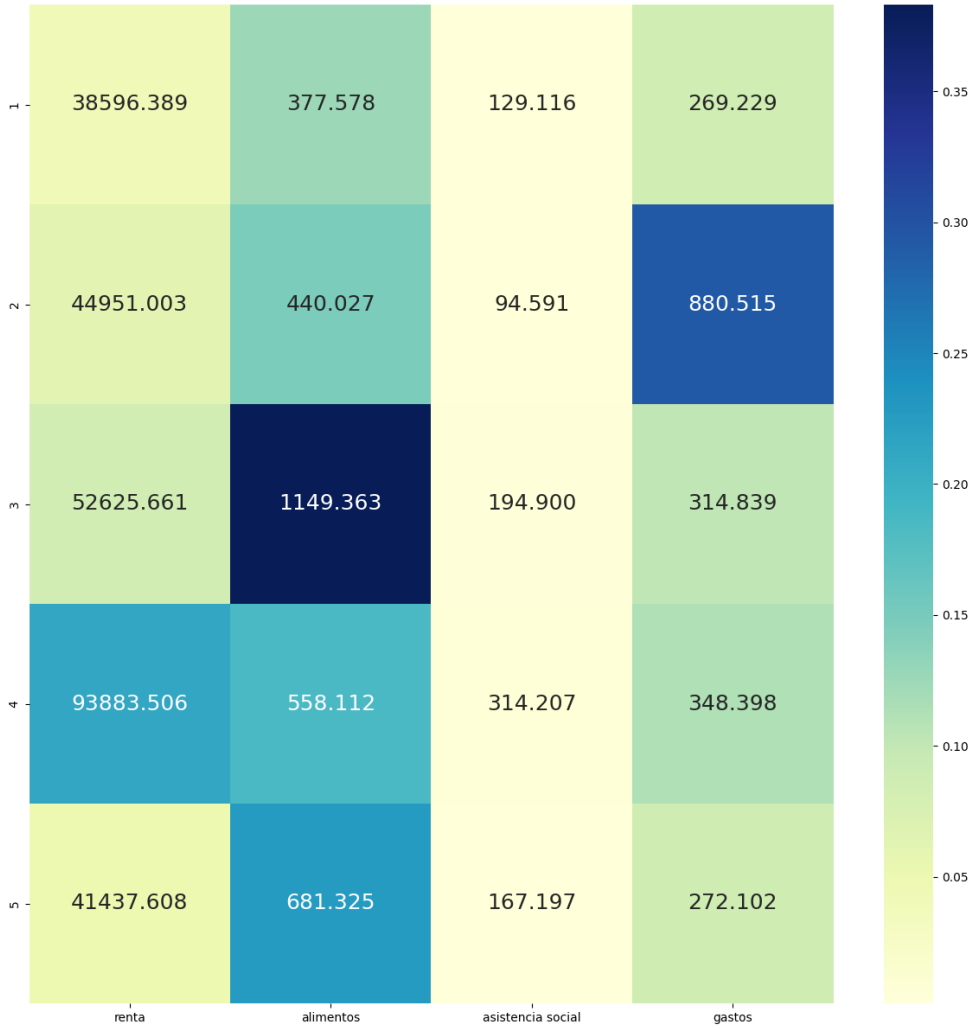
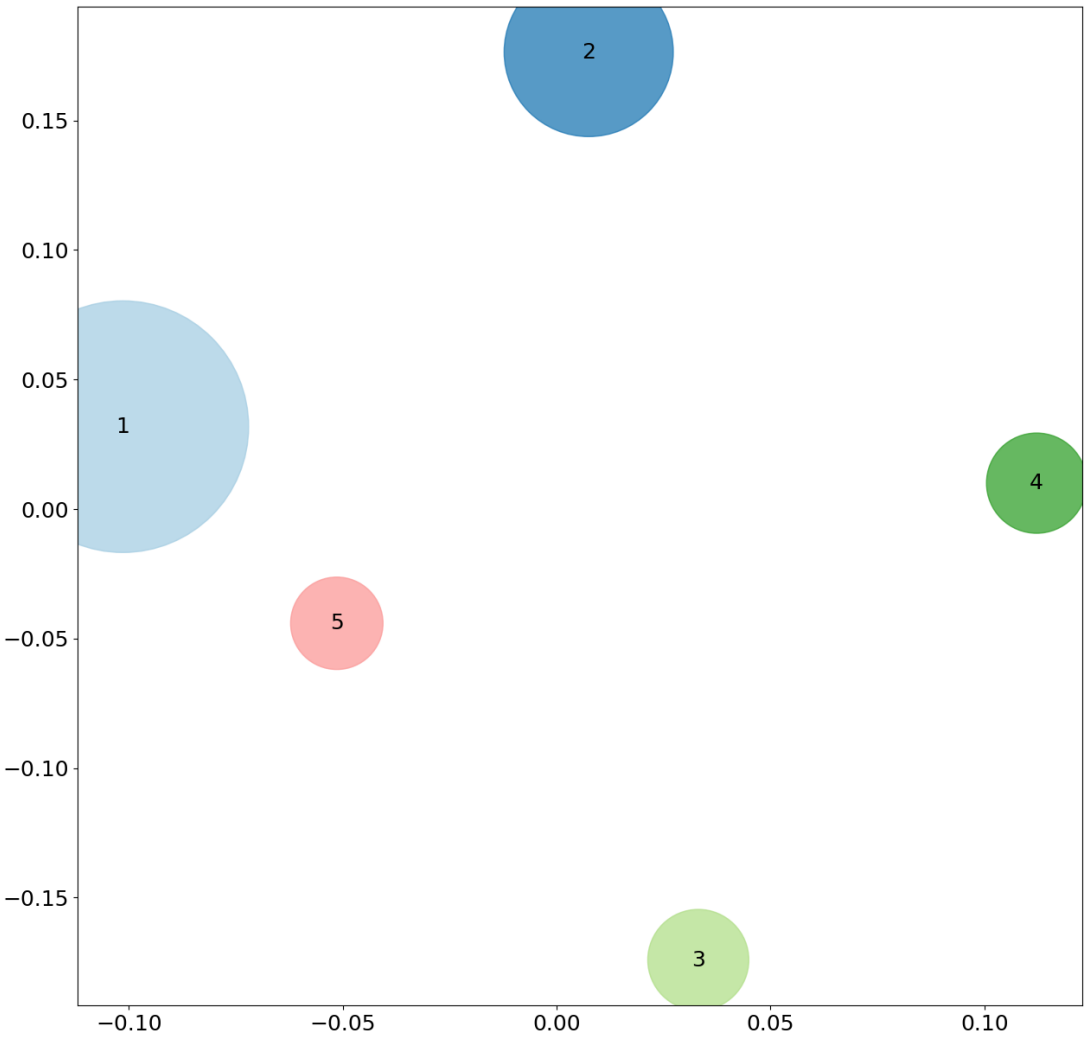


Figura 16: Caso 2B- Heatmap de kmeans



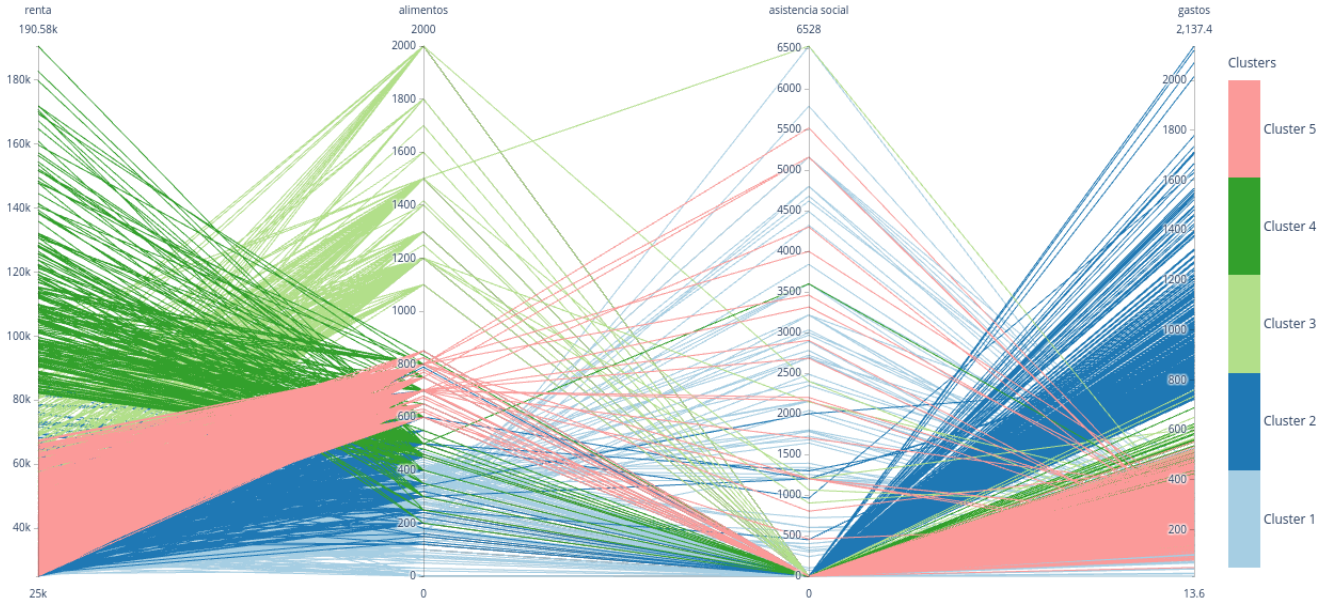
En (16) se ve con claridad los tres clusters que se diferencian por tomar valores especialmente altos en una de sus variables. Teniendo en cuenta que la numeración en este gráfico va de 1 a 5 y en la scatter matrix va de 0 a 4, vemos la analogía. En el heatmap se distingue como el cluster 2 toma valores considerablemente más altos en gastos que el resto de clusters, el 3 toma valores más altos en alimentación y el 4 en renta, lo que concuerda con lo visto en la scatter matrix.

Figura 17: Caso 2B- MDS de kmeans



En (17) se aprecia como los clusters están separados entre sí y hay un cluster más grande, otro medio y tres más pequeños.

Figura 18: Caso 2B - Parallel coordinates de kmeans



En (18) se observa como el cluster verde claro alcanza mayores valores de alimentos, como veíamos en las gráficas anteriores. De nuevo, es claro que hay dos clusters que se diferencian por sus valores altos en alimentación y gastos.

También resalta mucho en (16) y (18) como la asistencia social, en general, es 0 salvo para algunos objetos.

En la tabla (10) se muestran qué variables son necesarias para identificar cada cluster:

Cuadro 10: Caso 2B - Variables necesarias para separar el clustering en spectral cluster

Cluster	renta	alimentos	asistencia social	gastos
0		Bajos		Bajos
1				Altos
2		Altos		
3	Alta			
4		Medios		Bajos

Como se aprecia en (10) con una sola variable en cada cluster nos sirve para diferenciarlos entre los clusters 1, 2 y 3, mientras para los clusters 0 y 4 necesitamos dos para diferenciarlos entre ellos.

3.3.3 *Estudio de parámetros de DBSCAN*

Vamos a analizar el comportamiento de DBSCAN según sus parámetros principales. Para ello ejecutaremos la celda de parámetros de DBSCAN, que nos proporciona las medidas obtenidas para los parámetros dados.

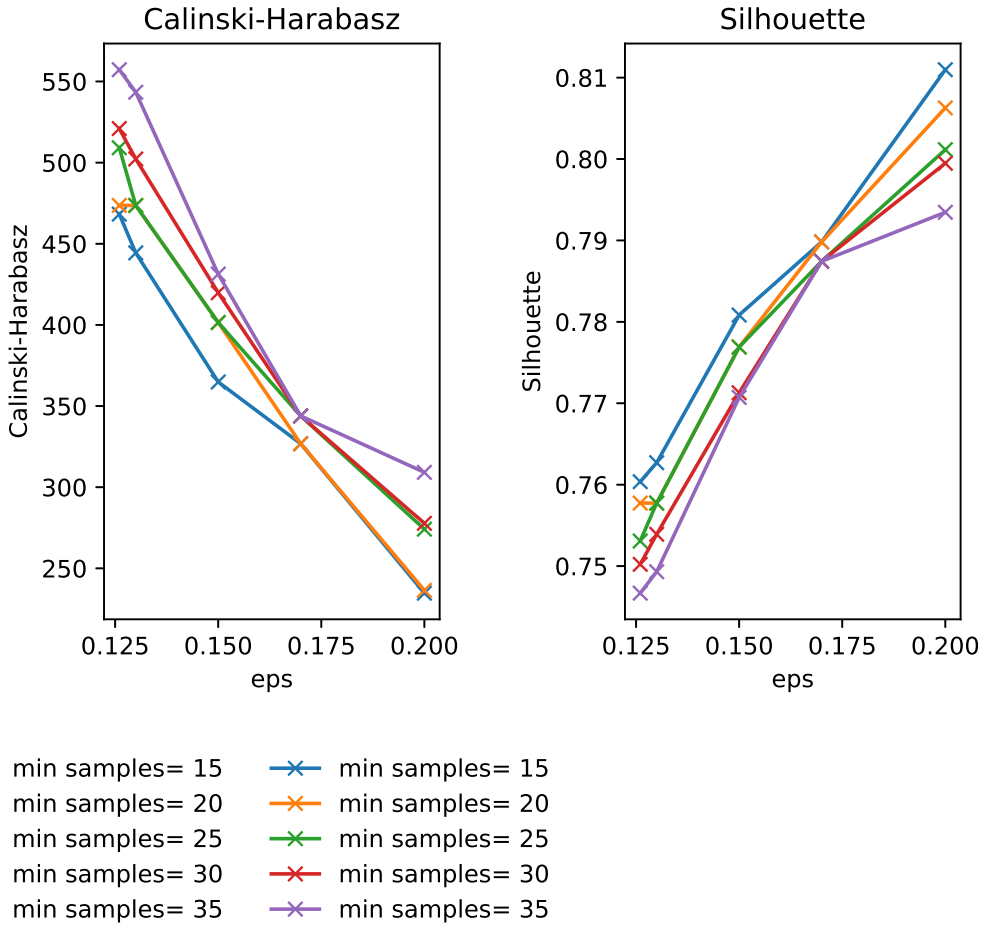
Cuadro 11: Caso 2B - Cambio de parámetros DBSCAN

Algoritmo	eps	min samples	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
dbscan	0.126	15	2.367	468.250	0.76037	2
dbscan	0.126	20	1.638	473.629	0.75774	2
dbscan	0.126	25	1.662	509.168	0.75308	2
dbscan	0.126	30	1.580	520.993	0.75022	2
dbscan	0.126	35	1.564	557.247	0.74668	2
dbscan	0.130	15	1.676	444.386	0.76270	2
dbscan	0.130	20	1.624	473.629	0.75774	2
dbscan	0.130	25	1.622	473.629	0.75774	2
dbscan	0.130	30	1.557	502.252	0.75391	2
dbscan	0.130	35	1.516	543.389	0.74931	2
dbscan	0.150	15	1.775	364.846	0.78084	2
dbscan	0.150	20	1.693	401.486	0.77691	2
dbscan	0.150	25	1.640	401.486	0.77691	2
dbscan	0.150	30	1.701	419.823	0.77129	2
dbscan	0.150	35	1.676	431.335	0.77072	2
dbscan	0.170	15	1.725	326.815	0.78982	2
dbscan	0.170	20	1.751	326.815	0.78982	2
dbscan	0.170	25	1.852	343.916	0.78744	2
dbscan	0.170	30	1.673	343.916	0.78744	2
dbscan	0.170	35	1.713	343.916	0.78744	2
dbscan	0.200	15	1.743	234.734	0.81097	2
dbscan	0.200	20	1.728	236.469	0.80628	2
dbscan	0.200	25	1.601	274.152	0.80115	2
dbscan	0.200	30	1.732	277.749	0.79948	2
dbscan	0.200	35	1.686	309.161	0.79346	2

Como podemos apreciar en (11) DBSCAN siempre realiza dos clusters. El tamaño de estos es siempre similar, más del 90 % los ejemplos están agrupados en el cluster 0 y los restantes están en el -1.

Este algoritmo no ha hecho ninguna buena agrupación en ningún caso de los probados, pese a que, de nuevo su silhouette es bastante alto.

Figura 19: Caso 2B- Parámetros de DBSCAN



En (19) se aprecia que no hay demasiada diferencia entre las medidas al variar los parámetros, aunque Calinski-harabasz es decendiente y silhouette creciente. No hay un punto donde ambos alcancen el máximo, y se debería realizar un estudio más a fondo o probar para los valores.

Igualmente, al generar solamente dos clusters, como ya se ha mencionado, la agrupación no parece ser buena.

3.3.4 *Estudio de parámetros de Kmeans*

Estudiamos ahora el algoritmo Kmeans. Se ha variado el número de clusters.

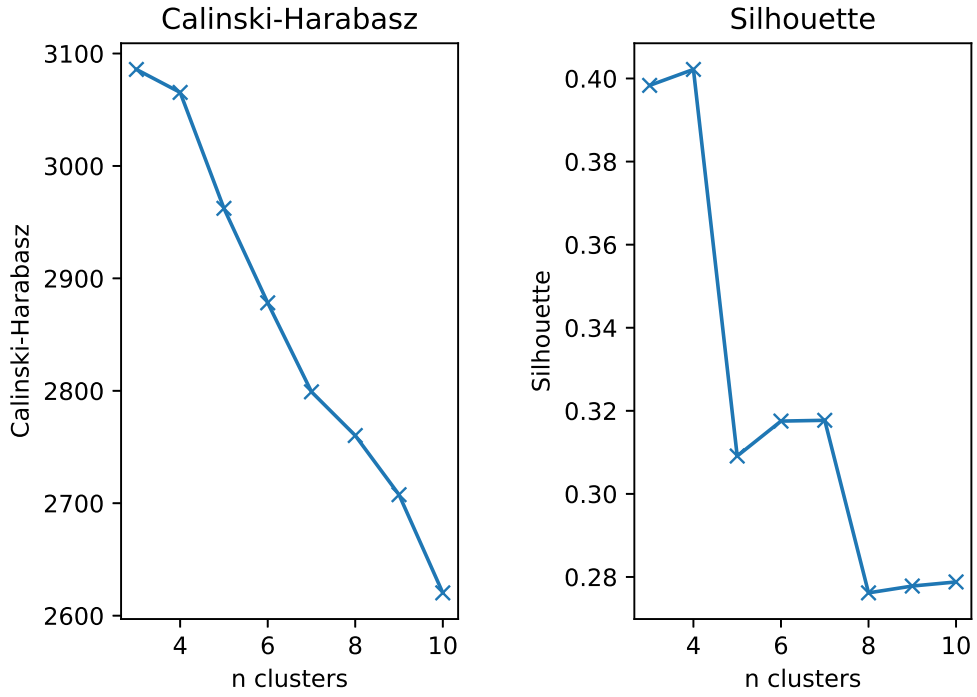
Cuadro 12: Caso 2B - Cambio de parámetros Kmeans

Algoritmo	n clusters	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
kmeans	3	0.392	3085.841	0.39834	3
kmeans	4	0.243	3065.400	0.40218	4
kmeans	5	0.407	2962.319	0.30911	5
kmeans	6	0.441	2878.260	0.31752	6
kmeans	7	0.496	2799.096	0.31771	7
kmeans	8	0.508	2760.228	0.27617	8
kmeans	9	0.787	2707.579	0.27781	9
kmeans	10	0.891	2620.283	0.27880	10

Podemos ver en (12) como las medidas comienzan subiendo hasta que el número de clusters es 4, donde alcanzan su máximo Silhouette y el segundo mayor valor de Calinski-harabasz, para después volver comenzar a decrementarse.

Asimismo el tiempo de ejecución se incrementa a la par que el número de clusters crece, pues hay que hacer más cálculos.

Figura 20: Caso 2B- Parámetros de Kmeans



Estudiamos ahora (20), en la que se puede ver como el silhouette no varía monótonamente. Calinski-Harabasz sí es monótonamente decreciente.

Al contrario que en el subcaso anterior, no hay un valor del número de clusters que maximice ambas medidas, aunque el punto en el que se alcanza el mayor silhouette, con 4 clusters, tiene también el segundo valor más alto de Calinski-harabasz.

A la vista de estas medidas, y a la falta de realizar un estudio mucho más en profundidad, parece que la mejor agrupación sería con 4 clusters.

3.4 INTERPRETACIÓN DE LA SEGMENTACIÓN

Una variable bastante relevante al comparar estos casos ha sido la asistencia social, ya que en el caso de rentas altas la mayor parte de objetos no recibían (valía 0), mientras para rentas bajas ha sido muy relevante a la hora de realizar agrupamientos, incluso siendo suficiente como para diferenciar un cluster si su valor era alto en las rentas bajas.

En ambos casos los gastos y alimentación han servido para separar dos clusters del resto usando solamente una de ellas. Si comparamos ambas, usando por ejemplos (8) y (15) vemos que en el caso de alimentación para rentas altas se considera un valor alto de esta variable a partir de 1000, mientras para rentas bajas es 400. Igualmente, para gastos en rentas altas el valor de corte es en torno a 500, pero para rentas bajas se considera más o menos 350.

Además, vemos como en general los valores de gastos y alimentación son considerablemente más altos para rentas altas que para rentas bajas.

Para separar los otros clusters usamos dos variables, donde una de ellas es la renta, por lo tanto es de menos utilidad al compararlos, ya que la renta está sesgada en cada caso de estudio.

Así, vemos como las personas con rentas más altas prácticamente no tienen ayudas sociales y gastan más en alimentos y gastos de alquiler y similares, mientras las personas con rentas bajas cuentan con más ayudas sociales y gastan menos en alimentación y gastos del hogar.

4

CASO 3: PERSONAS CON HIPOTECA, ALQUILER O CESIÓN GRATUITA

4.1 DESCRIPCIÓN DEL CASO DE ESTUDIO

En este caso de estudio se han elegido las respuestas para las que el régimen de tenencia es distinto de en propiedad sin hipoteca, es decir, pueden estar en propiedad con hipoteca, en alquiler a precio de mercado o inferior o en cesión gratuita.

Como en los casos anteriores los gastos, alimentación y ayudas sociales han sido muy relevantes, se intentará estudiar más a fondo la relación entre estas variables y si son más útiles que otras. Se pretende estudiar las relaciones entre gastos en la hipoteca o alquiler, alimentación, número de miembros y ayudas sociales.

Las variables que se van a seleccionar para realizar el análisis son:

- **Renta**: Renta disponible total del hogar en el año anterior al de encuesta.
- **Alimentos**: Durante el mes pasado, ¿cuál fue aproximadamente el importe que el hogar gastó en alimentos y bebidas no alcohólicas para ser consumidas en casa?
- **Asistencia social**: Ingresos por asistencia social en el año anterior al de encuesta.
- **Gastos**: 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.)
- **Número de miembros**: Número de miembros del hogar.

Este caso de estudio consta de 7295 respuestas a la encuesta, cada una con las 5 variables indicadas.

4.2 EJECUCIÓN DE ALGORITMOS

Las diferentes medidas obtenidas para cada algoritmo se muestran en (13).

Cuadro 13: Caso 3 - Resultados de ejecución de algoritmos.

Algoritmo	Tiempo (s)	Calinski-Harabasz	Silhouette	Número de clusters
kmeans	0.145	3108.765	0.28264	5
birch	0.277	668.429	0.35988	5
spectral	10.964	2192.446	0.28476	5
dbscan	0.801	310.408	0.57966	2
meanshift	58.323	140.587	0.28696	15

Al igual que en los otros casos, meanshift sigue tardando mucho más tiempo que el resto de algoritmos, seguido por spectral cluster. Llama la atención que DBSCAN en este caso ha resultado mucho más lento que en los anteriores, aunque sigue sin llegar a un segundo. Kmeans y birch han tardado tiempos similares.

De nuevo, DBSCAN ha hecho una agrupación mala en dos clusters, el cluster 0 con el 89,51 % de los objetos y los restantes en el cluster -1.

Meanshift ha agrupado los objetos en 15 clusters, y ha obtenido un valor de Calinski-Harabasz bastante bajo en comparación con los demás algoritmos, lo que hace sospechar que su agrupación no va a ser demasiado buena. Mirando el tamaño de los clusters generados ha realizado uno con el 94,67 % de los elementos, y el resto los ha agrupado en clusters con tamaños inferiores al 2,6 %. Por tanto, a pesar de que su silhouette no sea especialmente bajo en comparación con los otros algoritmos, parece que el clustering ha sido bastante malo.

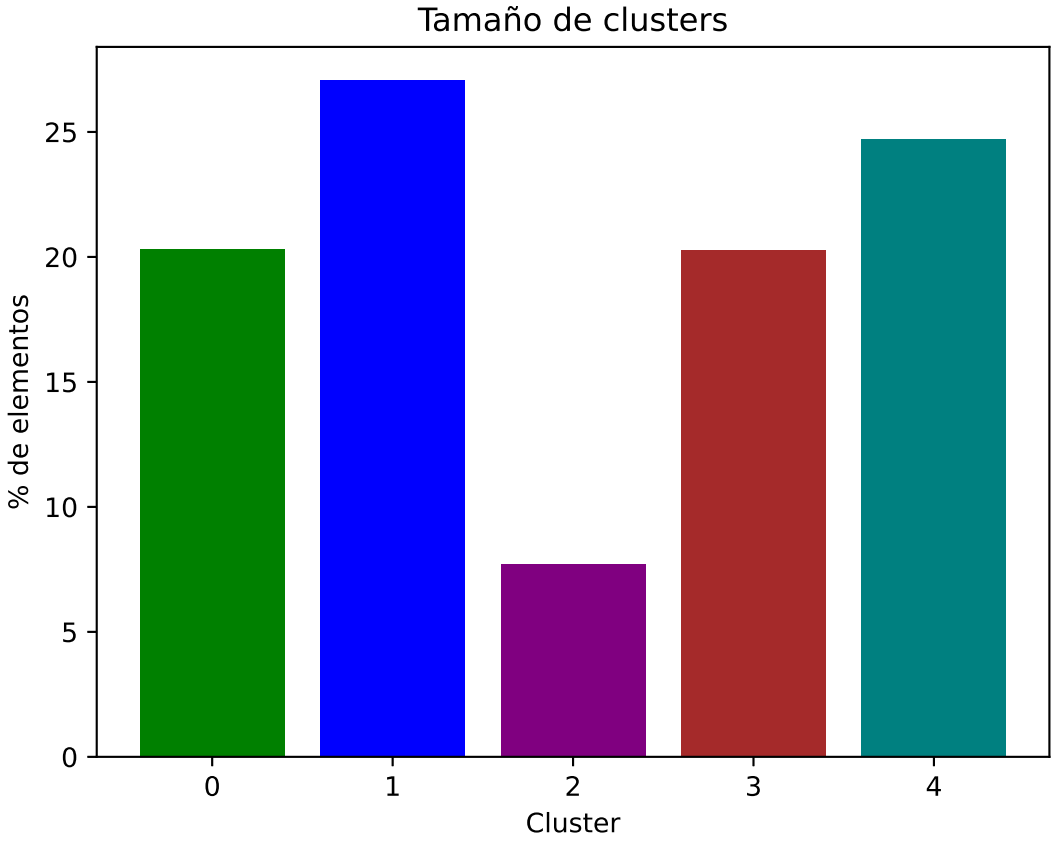
Estos algoritmos no han proporcionado buenos resultados en ninguno de los casos de estudio en comparación con los otros tres. Incluso birch, que es un algoritmo de clustering incremental y por tanto no se está usando en su campo, está consiguiendo mejores resultados.

De entre los tres algoritmos restantes, pese a que birch tiene el mayor silhouette, su calinski-harabasz es muy bajo. Y entre kmeans y spectral clustering, dado que tienen valores de silhouette similares, a priori kmeans habría realizado mejor agrupamiento, pues su calinski-harabasz es mayor.

4.3 ANÁLISIS

Para el análisis nos basaremos en los resultados de kmeans, pues ha obtenido, en principio, los mejores resultados en la ejecución. Este algoritmo genera 5 clusters. cuatro con tamaño similar y uno más pequeño, como se puede ver en (21)

Figura 21: Caso 3- Tamaño de los clusters generados por kmeans



Ahora estudiamos la scatter matrix (22):

Figura 22: Caso 3- Scatter matrix de kmeans

Analizamos (22), obteniendo así una distinción relativamente clara de los clusters. El cluster 2, de color verde, solo requiere una variable para identificarse, la de alimentos, mientras que los demás pueden distinguirse entre ellos usando solamente alimentos y el número de miembros del hogar.

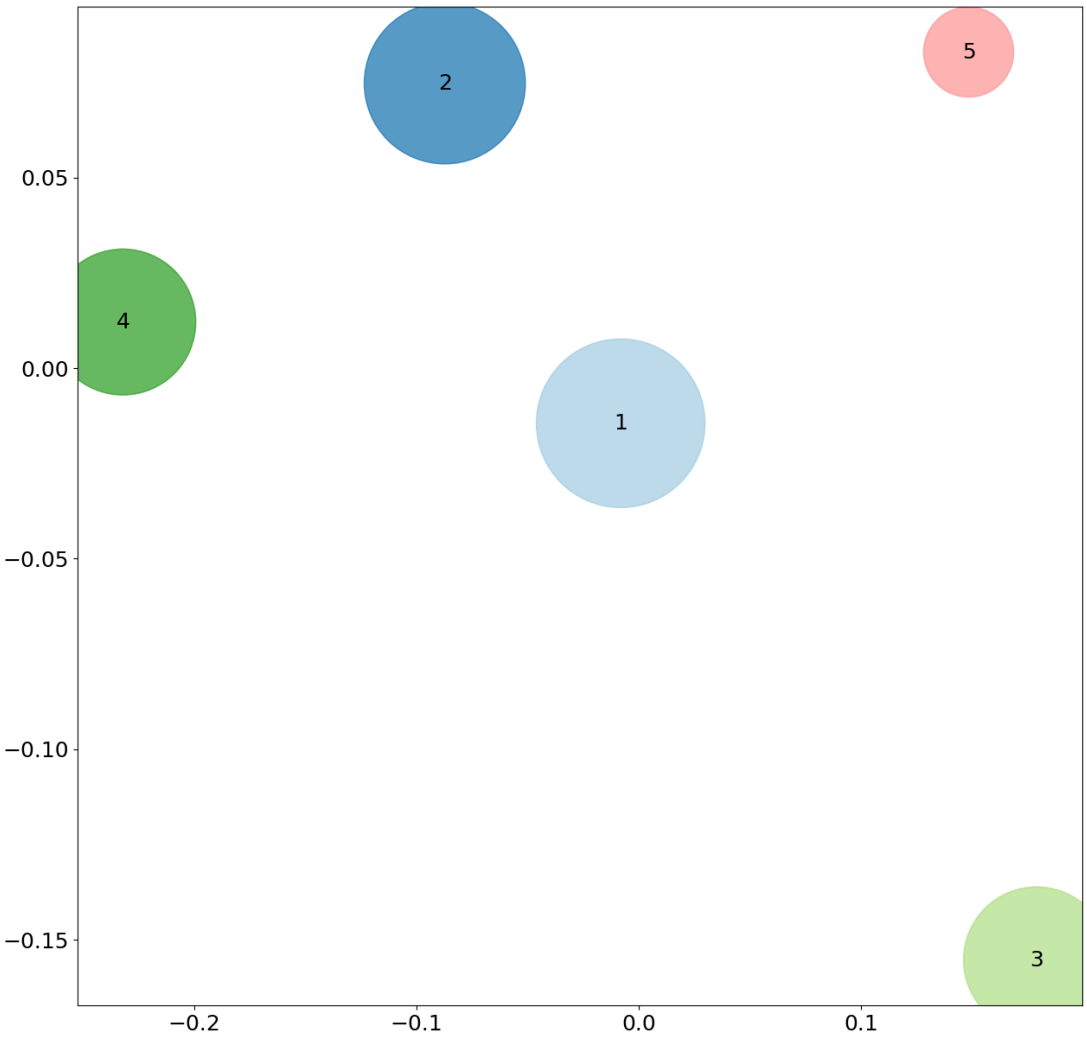
Llama la atención sobre esta gráfica como el número de miembros, al ser una variable discreta, los puntos se encuentran alineados en los valores enteros.

Figura 23: Caso 3- Heatmap de kmeans

En (23) podemos observar como el cluster 3 toma valores en general mayores en alimentos que los otros clusters.

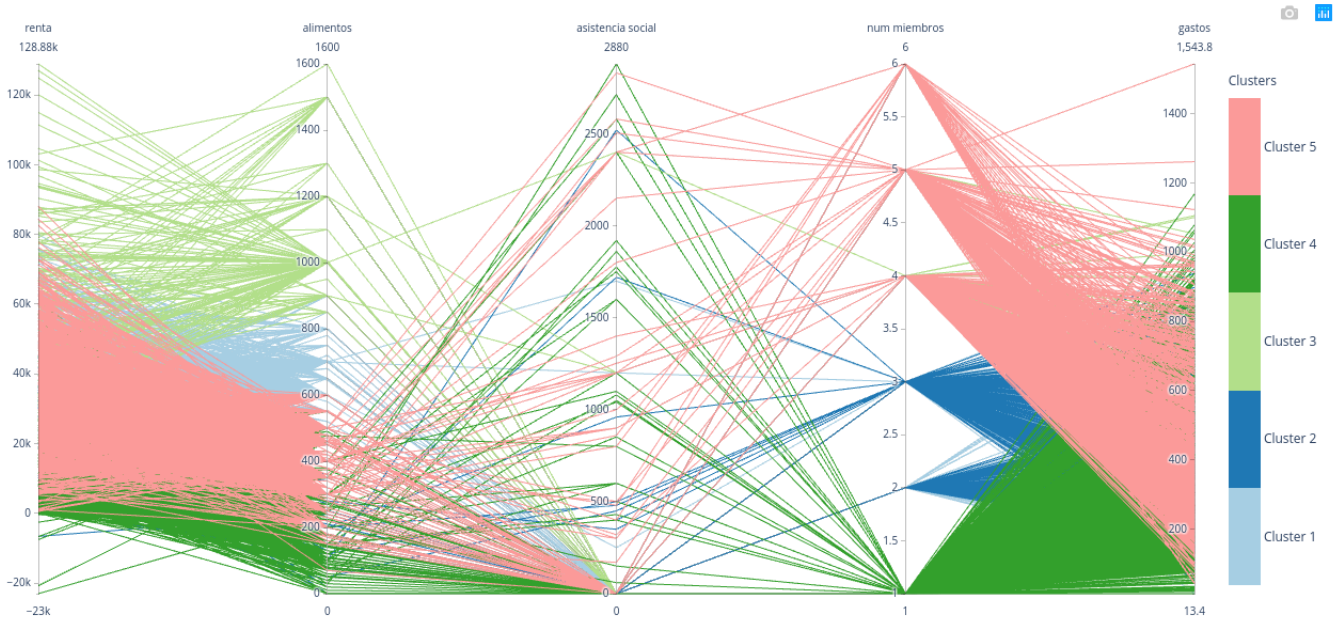
Sin embargo, pese a que en la scatter matrix (22) se distinguieran bien los clusters solamente usando alimentos y número de miembros, en el heatmap esta división no se ve tan clara, aunque si se aprecia que el cluster 4 (que en la scatter matrix es el 3) el número de miembros del hogar es 1. Y el cluster 5 puede distnguirse del 1, 2 y 4 usando el número de miembros, y del 3 mediante el número de alimentos.

Figura 24: Caso 3- MDS de kmeans



En (24) se aprecia como los clusters están notablemente separados y con tamaños bastante similares.

Figura 25: Caso 3- Parallel coordinates de kmean



Finalmente, en (25) es fácil observar como el número de miembros toma valores discretos, pues se agrupan las líneas en los números enteros de esta columna. También se ve claramente como el cluster verde claro, que es el 3, toma valores mucho más altos en alimentos que el resto, aunque es algo disperso. El cluster rosa, correspondiente al 5 es menos disperso y toma valores de número de miembros más altos que los otros clusters.

Si combinamos de nuevo la información de las líneas que llegan a alimentos y número de miembros, obtenemos los mismos resultados que mediante la scatter matrix.

En la tabla (14) se muestran qué variables son necesarias para identificar cada cluster:

Cuadro 14: Caso 3 - Variables necesarias para separar el clustering en kmeans

Cluster	renta	alimentos	asistencia social	num miembros	gastos
0		Medio		Medio	
1		Bajo		Medio	
2		Alto			
3		Bajo		Muy bajo	
4		Bajo		Alto	

Como se aprecia en (14) con solo la variable alimentos distinguimos el cluster 2, y con alimentos y num miembros separamos los demás.

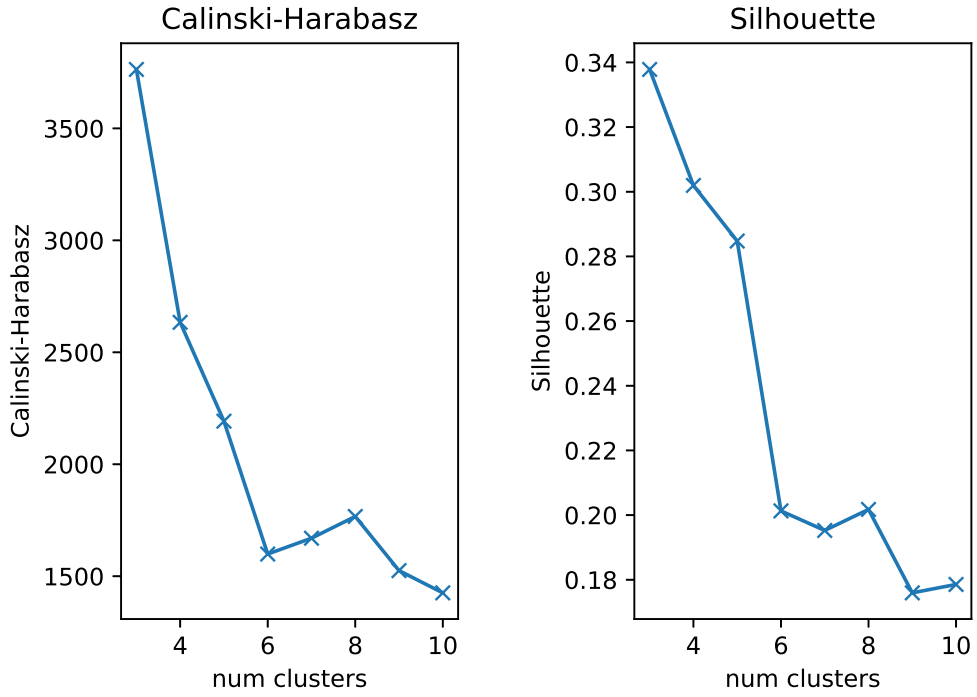
4.4 ESTUDIO DE PARÁMETROS DE SPECTRAL CLUSTER

Vamos a estudiar el comportamiento de spectral cluster variando el número de clusters que va a realizar.

Cuadro 15: Caso 3 - Cambio de parámetros Spectral cluster

Algoritmo	num clusters	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
spectral	3	10.986	3763.346	0.33783	3
spectral	4	10.916	2634.019	0.30196	4
spectral	5	10.949	2192.446	0.28476	5
spectral	6	10.964	1599.167	0.20130	6
spectral	7	11.258	1669.853	0.19525	7
spectral	8	11.180	1766.949	0.20175	8
spectral	9	11.662	1525.245	0.17595	9
spectral	10	11.725	1425.793	0.17855	10

Figura 26: Caso 3- Parámetros de Spectral cluster



Observamos en (15) y (26) que el mayor valor de silhouette se da cuando se realizan 3 clusters, así como el mayor valor de Calinski-harabasz. Ambas medidas son decrecientes según aumenta el número de clusters. Por tanto, en principio este es el mejor agrupamiento que ha realizado spectral cluster. Si miramos los tamaños de los clusters generados vemos que son 40,11 %, 40,08 % y 19,81 %, es decir, están equilibrados dos de ellos y el tercero es algo más pequeño. No parece a priori un mal clustering.

El resto de ejecuciones de spectral cluster también genera clusters de tamaños relativamente equilibrados.

Finalmente, cabe destacar que el tiempo de ejecución se incrementa a medida que el número de clusters aumenta, pero no en una medida exagerada.

A la vista de estas medidas, parece que todas las agrupaciones son decentes y spectral cluster se comporta de manera robusta, generando buenas agrupaciones independientemente del parámetro empleado.

4.5 ESTUDIO DE PARÁMETROS DE MEANSHIFT

Estudiamos ahora el algoritmo Meanshift. Se ha ido variando el radio (bandwith) del algoritmo, siendo la primera entrada en la tabla el que se calcula automáticamente al ejecutar el algoritmo.

Cuadro 16: Caso 3 - Cambio de parámetros Meanshift

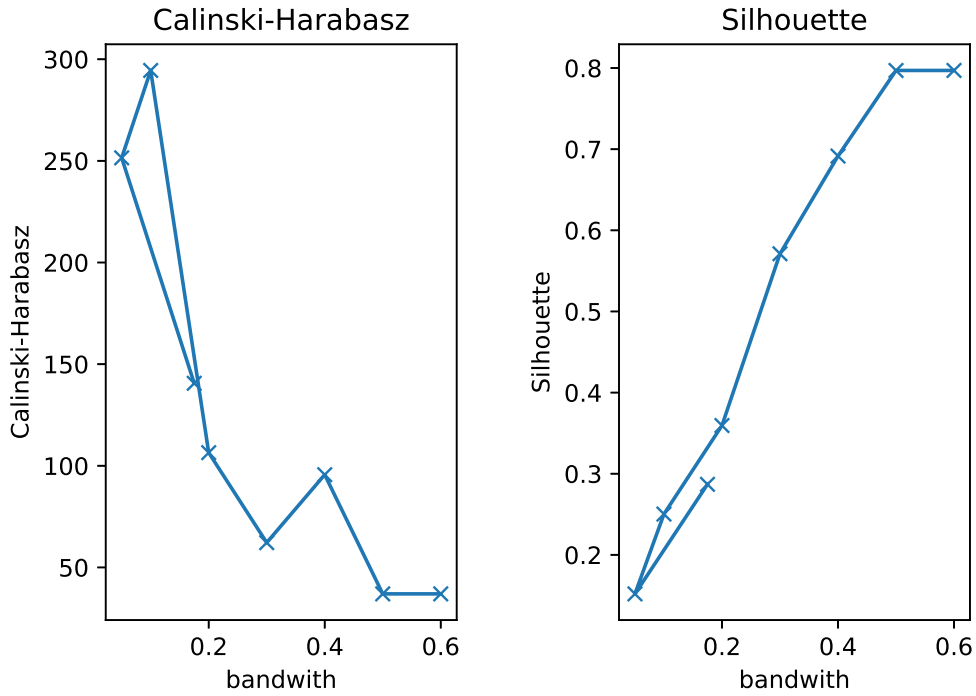
Algoritmo	bandwith	Tiempo(s)	Calinski-Harabasz	Silhouette	n clusters
meanshift	0.175	58.453	140.587	0.28696	15
meanshift	0.050	55.001	251.495	0.15194	445
meanshift	0.100	48.993	294.485	0.25028	120
meanshift	0.200	40.302	106.456	0.35944	13
meanshift	0.300	38.224	62.198	0.57110	7
meanshift	0.400	27.835	95.624	0.69156	3
meanshift	0.500	23.839	36.994	0.79705	2
meanshift	0.600	20.824	36.994	0.79705	2
meanshift	0.700	18.235			2
meanshift	0.800	15.558			2
meanshift	0.900	15.321			1
meanshift	1	15.092			1

En la primera entrada de la tabla (16) se encuentra el radio calculado automáticamente, que genera 15 clusters. Como se comentó al analizar los algoritmos en general

para este caso de estudio, no realiza un buen agrupamiento por los tamaños de los clusters.

En la tabla faltan las medidas cuando meanshift genera un cluster, ya que no tienen sentido, además en las últimas entradas donde genera dos clusters tampoco se muestran, ya que no se podían calcular.

Figura 27: Caso 3- Parámetros de Meanshift



En (27), al no estar los valores en el eje x ordenados de menor a mayor (ya que se desconoce a priori el valor que tomará el radio generado automáticamente) la línea no inicia por el punto más a la izquierda, generando así un cruce de la línea.

Observando la gráfica se aprecia como Calinski-harabasz van decreciendo según se aumenta el radio, salvo cuando este vale 0.1 o 0.4, en el que hay un pico y el silhouette va creciendo. Mirando en la tabla vemos que cuando el radio vale 0.1 se generan 120 clusters, por lo que no parece tampoco un buen agrupamiento.

Es llamativo como a medida que aumenta el radio, el número de clusters se reduce. Los casos más razonables parecen cuando el radio vale 0,3 y se generan 7 clusters y cuando vale 0,4 y se generan 3, pero mirando los tamaños de los clusters generados observamos que en ambos casos se genera un cluster con más del 99,5 % de los objetos y el resto se agrupa en los clusters restantes. Esto puede indicar que no es un buen agrupamiento.

Finalmente, vemos como el tiempo se reduce drásticamente conforme lo hace el número de clusters generados.

4.6 INTERPRETACIÓN DE LA SEGMENTACIÓN

Fijándonos en los resultados obtenidos, podemos concluir que las variables más relevantes en este caso han sido el número de miembros del hogar y el gasto en alimentación, pues solamente con ellas se pueden distinguir todos los clusters. Luego en este caso los gastos y asistencia social no han sido tan relevantes como en los anteriores.

Además, los clusters están bastante separados, lo que puede deberse en gran parte a que el número de miembros solo toma valores enteros naturales, lo que puede estar forzando una separación usando esta variable e incrementando el silhouette artificialmente, pues premia mucho la separación entre los clusters.

Finalmente cabe destacar que al contrario de lo que podríamos haber pensado en un principio, la renta o los gastos en la casa y el número de miembros por familia no producen tan buena agrupación como lo hace la alimentación.

5

CONTENIDO ADICIONAL

5.1 EJECUCIÓN DE CÓDIGO

Para la ejecución de código se ha usado Google Colab usando un cuaderno en línea con python. Por ello, la estructura del fichero de código consta de algunas celdas para cargar el google drive o usar una estructura de carpetas que no va incorporada en la práctica, de cara a comprobar si el código funciona.

Se recomienda la ejecución del código usando google colab en caso de que fuera necesario. La base de datos debe encontrarse subida a drive y en content las carpetas de cada caso con una subcarpeta por algoritmo, teniendo en cuenta que el caso 2 está dividido en dos subcasos, llamados caso2A y caso2B.

Así, en el primer nivel de carpetas tendríamos caso1, caso2A, caso2B y caso3. Y dentro de cada una de ellas una carpeta por algoritmo, llamadas como sigue: kmeans, birch, dbscan, spectral y meanshift.

Además, hay varios scripts para sacar gráficas auxiliares (los gráficos de barras de los tamaños de los clusters y las gráficas de los estudios de parámetros) que se encuentran en ficheros.py y no en el cuaderno.

Como comentario final, para generar unas gráficas u otras se debe comentar o descomentar la sección correspondiente a las gráficas en el código.

5.2 GRÁFICAS

En la memoria se han incluido solamente las gráficas que se han considerado relevantes, pero se pueden encontrar más gráficas dentro de la carpeta figures o graficas, donde también hay varios scripts para generarlas.

6

BIBLIOGRAFÍA

- Diapositivas de la asignatura.
- http://scikit-learn.org/stable/modules/clustering.html
- http://www.learndatasci.com/k-means-clustering-algorithms-python-intro/
- https://pandas.pydata.org/docs/
- https://seaborn.pydata.org/introduction.html
- https://matplotlib.org/