

# **Práctica 2: Análisis Relacional Mediante Segmentación**

Juan Antonio Villegas Recio

10 de Diciembre de 2021

Inteligencia de Negocio  
Curso 2021/2022



**UNIVERSIDAD  
DE GRANADA**



# Índice

<b>Introducción</b>	<b>1</b>
El conjunto de datos . . . . .	1
Metodología . . . . .	2
<b>Caso de estudio: Encuestados que no viven en zonas poco pobladas</b>	<b>3</b>
Discusión de parámetros . . . . .	3
K-Means . . . . .	4
DBSCAN . . . . .	4
Ejecución de los algoritmos . . . . .	7
Interpretación de los resultados . . . . .	8
<b>Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler</b>	<b>13</b>
Discusión de parámetros . . . . .	13
K-Means . . . . .	13
DBSCAN . . . . .	14
Ejecución de los algoritmos . . . . .	15
Interpretación de resultados . . . . .	19
<b>Caso de estudio: Encuestados con mayor renta</b>	<b>23</b>
Discusión de parámetros . . . . .	24
K-Means . . . . .	24
DBSCAN . . . . .	24
Ejecución de los algoritmos . . . . .	25
Interpretación de los resultados . . . . .	28
<b>Contenido adicional</b>	<b>31</b>
<b>Bibliografía</b>	<b>33</b>



# Introducción

En España, y en el mundo en general, las condiciones de vida de la gente son muy variadas y dependen de muchos factores como la renta, posibles hipotecas o alquileres, préstamos, consumo, etc. En este contexto el “INE” lleva a cabo anualmente desde 2004 la Encuesta de Condiciones de Vida. Sobre los datos que dio esta encuesta en el año 2020 hemos aplicado algoritmos de segmentación (clustering) para poder descubrir posibles grupos con características similares.

En este informe se expondrán algunas agrupaciones calculadas por distintos algoritmos de clustering, exponiendo sus resultados en cuanto a medidas de rendimiento e incluyendo análisis e interpretaciones de los resultados obtenidos. También se ha hecho una discusión de los parámetros de dos de los algoritmos, mostrando cuáles son los que dan mejores resultados. En concreto, se han utilizado los siguientes algoritmos de clustering:

- **K-Means**
- **Mean Shift**
- **DBSCAN**
- **Spectral Clustering**
- **Agglomerative Clustering**

Los cuales se han ejecutado sobre los distintos casos de estudio que hemos considerado, recopilando gráficas y medidas de rendimiento que expondremos y discutiremos a continuacion. En concreto, sobre *K-Means* y sobre *DBSCAN* se ha realizado un estudio intensivo de parámetros, ya que *K-Means* necesita el número de clusters (al que nos referiremos como `n_clusters`) como argumento y *DBSCAN* necesita el radio de búsqueda y el mínimo de ejemplos a considerar para incluir en el cluster (`eps` y `min_samples` respectivamente).

## El conjunto de datos

Hablaremos un poco acerca del propio conjunto de datos. Consta originalmente de 15043 filas y 209 columnas, lo que quiere decir que hay 15043 encuestados y 209 variables distintas, aunque aproximadamente la mitad de las columnas son variables “flag” (cuyo formato es `_F`) complementarias a otra variable que simplemente indican si el encuestado respondió o no a dicha variable o si procede que responda. Por tanto, para hacer clustering realmente quedan aproximadamente unas 100 variables de las cuales muchas son categóricas, cuando el clustering nos ofrece mejores resultados cuando usamos variables continuas. Veremos este hecho proximamente.

Por otra parte, el conjunto de datos de por sí tiene muchos valores perdidos, debido entre otras cosas a preguntas que no proceden. Por ejemplo, no tiene sentido preguntar

## *Introducción*

cuánto paga de alquiler a una persona que vive en una casa comprada. Por esto, muchas variables tienen valores perdidos. Sobre esto se ha hecho un estudio que nos revela que un total de 57 variables, el equivalente a un 38% de las variables tienen al menos un valor perdido, y que 21 variables, el equivalente al 10.047% del total tiene al menos 10000 valores perdidos, los cuales son muchos considerando que en total hay aproximadamente 15000 filas en el *dataset*. De todas formas, se filtrarán las filas al especificar un caso de estudio y la concentración de valores perdidos variará con respecto a lo recientemente comentado respecto al *dataset* completo.

Para evitar problemas debido a la variedad de los datos, previo a la ejecución de los algoritmos se ha hecho un preprocessado básico a los datos correspondientes a cada caso de uso una vez seleccionadas las variables sobre las que se iba a realizar el clustering. Este preprocessado consistía en seleccionar únicamente las columnas correspondientes a las variables usadas en el clustering, obteniendo el conjunto de datos que denotaremos X. Sobre el conjunto de datos X, secuencialmente:

1. Se sustituían los valores perdidos por la media de los valores de dicha variable en el conjunto de datos completo.
2. Se eliminaban outliers, entendiendo como *outliers* los valores que normalizando la columna obtenían un z-score mayor que 3.
3. Se normalizaban los valores usando normalización gaussiana para que los órdenes de magnitud no afecten al clustering. Con este paso obtenemos el conjunto de datos X\_normal.

El código completo de estas operaciones puede encontrarse en el fichero `preprocesado.py`.

## **Metodología**

Partiendo de los conjuntos preprocessados X y X\_normal, hemos ejecutado los algoritmos de clustering anteriormente comentados, obteniendo de dicha ejecución algunas medidas de rendimiento, concretamente el **tiempo de ejecución** (en segundos), el **coeficiente de Calinski-Harabasz**, el coeficiente de **silhouette** y el **número de clusters** obtenidos, que en el caso de *K-Means* y *Spectral Clustering* coincide con el que se especifica como argumento. A partir de estas variables podremos aproximar la calidad del clustering y comparar unos algoritmos con otros, además de comparar la influencia de los distintos parámetros ajustables.

Junto con estas medidas se han obtenido gráficas que permiten interpretar los resultados obtenidos. En esta memoria expondremos algunas de las más destacables junto con su correspondiente interpretación.

Procedemos pues a tratar los distintos casos de estudio.

# Caso de estudio: Encuestados que no viven en zonas poco pobladas

Es claro que en España la densidad de población es muy variada, de hecho en cada provincia hay zonas con alta población y zonas más deshabitadas. En general los alquileres suelen centrarse, probablemente debido a la oferta, en zonas más pobladas, pero también sube el precio de estos alquileres. En este caso de estudio veremos posibles relaciones y agrupaciones entre las personas que no viven en zonas poco pobladas, es decir, la variable **DB100: Grado de urbanización** toma un valor inferior a 3. Este subconjunto de datos contiene un total de 10821 filas, lo que supone el 71.93% del total. Dentro de este grupo de personas estudiaremos posibles segmentaciones en torno a las variables:

- HY020: Renta disponible total del hogar en el año anterior al de encuesta. Renombrada como `renta`.
- HH031: Año del contrato o de la compra o de instalación. Renombrada como `anio_contrato`.
- HS130: Ingresos mínimos para llegar a final de mes. Renombrada como `ingresos_minimos`.
- HH060: Alquiler actual por la vivienda ocupada. Renombrada como `alquiler_actual`.

## Discusión de parámetros

Sobre este conjunto de datos y estas variables hemos ejecutado los distintos algoritmos, obteniendo distintos resultados que posteriormente comentaremos, pero antes nos centramos en los algoritmos *K-Means* y *DBSCAN* y en los posibles parámetros que aceptan y cómo afectan al rendimiento del algoritmo.

Para ello, se ha implementado una métrica que busca un balance entre los valores de *Calinski-Harabasz* y de *silhouette* de la siguiente forma:

- Se normalizan los valores de los coeficientes obtenidos variando el parámetro utilizando normalización gaussiana.
- Una vez normalizados se calcula el cuadrado de la norma que formaría el vector (*CH – normalizado, silhouette – normalizado*).

Para ilustrar mejor el procedimiento mostramos el código de esta operación:

```
tabla_aux, scaler = normaliza(tabla_comparativa_parametros)
tabla_aux = tabla_aux[["Calinski-Harabasz", "Silhouette"]]
tabla_comparativa_parametros['Metrica'] = tabla_aux['Calinski-Harabasz']**2
+ tabla_aux['Silhouette']**2
```

Caso de estudio: Encuestados que no viven en zonas poco pobladas

## K-Means

A *K-Means* hay que especificarle el número de clusters que queremos que genere, lo cual es una de sus desventajas, pero podemos realizar varias ejecuciones y ver cuál arroja mejores resultados. La tabla siguiente contiene las medidas de rendimiento en función del valor de `n_clusters`:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
n_clusters = 2	0.833863	3562.18	0.26899	2	0.690638
n_clusters = 3	0.85087	3488.5	0.264078	3	0.246098
n_clusters = 4	1.53865	3653.38	0.292305	4	4.21136
n_clusters = 5	0.665498	3979.64	0.30932	5	10.9099
n_clusters = 6	0.37529	3870.76	0.270715	6	2.53969
n_clusters = 7	0.471179	3859.28	0.277878	7	3.14847
n_clusters = 8	0.462676	3698.22	0.257548	8	0.908438
n_clusters = 9	0.686844	3609.15	0.258347	9	0.496689
n_clusters = 10	0.572541	3491	0.245505	10	0.609119
n_clusters = 11	0.586445	3349.88	0.252772	11	0.0824386
n_clusters = 12	0.498304	3253.9	0.251075	12	0.225621
n_clusters = 13	0.504493	3173.64	0.242217	13	1.00673
n_clusters = 14	0.593639	3150.53	0.246883	14	0.700075
n_clusters = 15	1.51017	3082.3	0.249383	15	0.806567
n_clusters = 16	2.09456	3024.63	0.241419	16	1.67979
n_clusters = 17	0.729023	2967.71	0.24536	17	1.65311
n_clusters = 18	0.621048	2910.03	0.241028	18	2.41192
n_clusters = 19	0.596921	2860.46	0.244102	19	2.49033
n_clusters = 20	0.918963	2786.34	0.243187	20	3.18296

En este caso como podemos observar en general Calinski-Harabasz y silhouette crecen y decrecen conjuntamente. Además, es claro que hay un valor, que es `n_clusters = 5` que aporta el mayor silhouette y el mayor Caliski-Harabasz en un tiempo inferior al de los demás posibles valores del parámetro que se le acercan en cuanto rendimiento. Este hecho se hace notar también en la métrica anteriormente definida. Para clarificarlo, podemos fijarnos en el gráfico de la figura 0.1.

Observamos que el máximo se alcanza claramente en los tres casos cuando *K-Means* crea 5 clusters, por lo que consideramos que lo mejor es ejecutar *K-Means* en este caso de estudio con `n_clusters=5`.

## DBSCAN

El algoritmo *DBSCAN* acepta como parámetros un radio que denotamos como `eps` y un mínimo de puntos que denotamos como `min_samples`. Al ser dos parámetros y uno de ellos continuo, el espacio de posibilidades es mayor que en *K-Means*. Hemos variado los parámetros con valores a distancia 0.5 cada uno de su antecesor y de su predecesor en el caso de `eps` y del 2 al 19 en el caso de `min_samples`, obteniendo la siguiente tabla:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
eps = 0.5, min_samples = 1	0.520944	6.57731	-0.419846	304	17.6125
eps = 1.0, min_samples = 1	1.07568	6.12479	0.116756	32	2.51189
eps = 1.5, min_samples = 1	1.64163	4.84597	0.275662	7	0.854082

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
eps = 2.0, min_samples = 1	2.17582	9.43146	0.587206	2	1.1405
eps = 0.5, min_samples = 3	0.467646	30.5304	-0.0898087	22	6.25095
eps = 1.0, min_samples = 3	1.05965	25.8769	0.421302	3	0.0755977
eps = 1.5, min_samples = 3	1.61535	6.6795	0.486501	2	0.538858
eps = 2.0, min_samples = 3	2.11039	9.43146	0.587206	2	1.1405
eps = 0.5, min_samples = 5	0.465941	54.2495	0.227222	10	0.876537
eps = 1.0, min_samples = 5	1.02348	28.2401	0.434535	2	0.068738
eps = 1.5, min_samples = 5	1.64324	9.24068	0.486501	2	0.483569
eps = 2.0, min_samples = 5	2.07003	9.43146	0.587206	2	1.1405
eps = 0.5, min_samples = 7	0.45091	151.092	0.285299	5	4.44679
eps = 1.0, min_samples = 7	0.976187	27.5209	0.433109	2	0.0727856
eps = 1.5, min_samples = 7	1.49444	9.24068	0.486501	2	0.483569
eps = 2.0, min_samples = 7	2.1454	9.43146	0.587206	2	1.1405
eps = 0.5, min_samples = 9	0.439866	133.181	0.306422	5	3.11659
eps = 1.0, min_samples = 9	0.948044	30.7535	0.42399	2	0.0385151
eps = 1.5, min_samples = 9	1.5013	9.24068	0.486501	2	0.483569
eps = 2.0, min_samples = 9	1.97817	9.43146	0.587206	2	1.1405
eps = 0.5, min_samples = 11	0.44082	110.413	0.300875	5	1.9127
eps = 1.0, min_samples = 11	1.00082	32.2344	0.424698	2	0.0303599
eps = 1.5, min_samples = 11	1.52584	6.99964	0.486501	2	0.531709
eps = 2.0, min_samples = 11	1.99537	9.43146	0.587206	2	1.1405
eps = 0.5, min_samples = 13	0.437803	121.532	0.290904	5	2.52736
eps = 1.0, min_samples = 13	0.938482	31.3639	0.436226	2	0.0491353
eps = 1.5, min_samples = 13	1.50629	7.23129	0.486501	2	0.526578
eps = 2.0, min_samples = 13	1.94632	9.43393	0.587206	2	1.14045
eps = 0.5, min_samples = 15	0.438873	117.533	0.328312	4	2.13361
eps = 1.0, min_samples = 15	0.920527	38.9337	0.433731	2	0.0190873
eps = 1.5, min_samples = 15	1.48381	9.11402	0.486501	2	0.486201
eps = 2.0, min_samples = 15	1.92343	9.43393	0.587206	2	1.14045
eps = 0.5, min_samples = 17	0.426906	258.621	0.373316	2	15.8665
eps = 1.0, min_samples = 17	0.953369	40.9786	0.442682	2	0.0326716
eps = 1.5, min_samples = 17	1.47067	9.11402	0.486501	2	0.486201
eps = 2.0, min_samples = 17	1.9716	9.43393	0.587206	2	1.14045
eps = 0.5, min_samples = 19	0.427922	145.239	0.130104	4	5.58537
eps = 1.0, min_samples = 19	0.954074	47.9379	0.443879	2	0.0536186
eps = 1.5, min_samples = 19	1.46739	11.4433	0.486501	2	0.439511
eps = 2.0, min_samples = 19	1.98121	9.43393	0.587206	2	1.14045

En este caso se puede apreciar como la relación entre los coeficientes no es tan directa, de hecho casi pareciera que cuando crece C-H disminuye silhouette y viceversa. Observemos como hay configuraciones con valores muy altos de silhouette, pero sin embargo sólo genera 2 clusters, lo cual corresponde con agrupar muchos ejemplos en un cluster y los más externos marcarlos como ruido, lo cual no es un buen clustering. Por ello, nos interesan los casos en los que haya al menos 3 clusters. Por otra parte, hay valores con métrica muy alta, pero silhouette negativo. No se debe tomar el coeficiente silhouette como medida definitiva, pero es cierto que un silhouette negativo es objetivamente un síntoma de mala agrupación. Para ayudarnos a elegir unos parámetros definitivos podemos ayudarnos del gráfico de la figura 0.2, en el que las columnas representan el valor de `eps` y las filas el valor de `min_samples` y el valor indicado es el valor de la métrica definida anteriormente.

En este heatmap hay varias celdas muy oscuras y muy llamativas, pero corresponden a configuraciones con silhouette negativo, por lo que las descartaremos. Sin contar con estas, la más razonable es la celda que corresponde a `eps = 0.5, min_samples = 5`, por lo que será la que empleemos.

## Caso de estudio: Encuestados que no viven en zonas poco pobladas

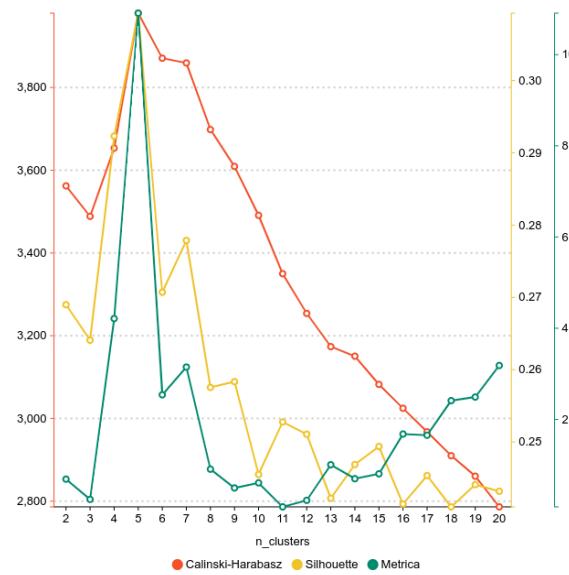


Figura 0.1: Gráfico comparativo del parámetro de K-Means

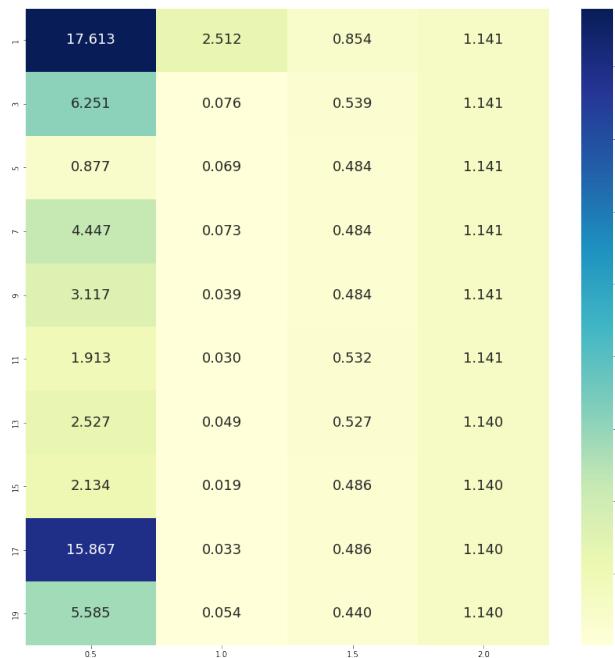


Figura 0.2: Heatmap parámetros DBSCAN

## Ejecución de los algoritmos

Ya elegidos los mejores parámetros para *K-Means* y *DBSCAN*, ejecutamos los algoritmos de clustering. En la siguiente tabla presentamos una comparativa del rendimiento de los distintos algoritmos en base a las distintas métricas que tenemos:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters
K-Means	0.501167	3979.64	0.30932	5
Mean Shift	3.24575	1007.23	0.259081	5
DBSCAN	0.497929	151.092	0.285299	5
Spectral Clustering	39.5008	1349.22	0.451777	3
Agglomerative Clustering	4.56966	2965.18	0.215779	2

Lo primero que salta a la vista es lo rápidos que son *K-Means* y *DBSCAN* en comparación con los demás y lo lento que es *Spectral Clustering*, pues tarda mucho más en ejecutar. A pesar de ello, *Spectral Clustering* obtiene el mayor de los silhouette y un buen coeficiente de *Calinski-Harabasz*. Por su parte, DBSCAN ni en su mejor versión ha conseguido siquiera acercarse a los demás algoritmos en su coeficiente de *Calinski-Harabasz*, a pesar de obtener un *silhouette* decente. Observemos los clusters que ha generado DBSCAN y su tamaño:

	Tamaño	Porcentaje
0	9688	93.8942
1	534	5.17542
2	53	0.513665
3	37	0.358597
4	6	0.0581508

En efecto, ha generado un gran cluster donde ha incluido al 93.89 % del total de ejemplos del conjunto de datos, lo cual facilita un incremento del silhouette que realmente no es representativo.

Nos centraremos ahora en los resultados de *K-Means* para poder interpretarlos. En primer lugar podemos ver en la figura 0.3 un gráfico de coordenadas paralelas en el que podremos diferenciar agrupaciones de manera rápida y clara.

Como podemos observar en la imagen, se diferencian bien en agrupaciones de líneas coloreadas los distintos clusters, aunque también hay varios solapamientos en las distintas variables, por ejemplo, el cluster 0 a penas se puede apreciar a pesar de ser el mayoritario como indica la siguiente tabla:

	Tamaño	Porcentaje
0	4171	40.4245
1	2640	25.5864
2	2604	25.2374
3	478	4.63268

## Caso de estudio: Encuestados que no viven en zonas poco pobladas

Tamaño	Porcentaje
4	425      4.11902

Si nos fijamos ahora en la matriz de dispersión (figura 0.4), como podemos observar, los 5 clusters se diferencian de forma clara, aunque en la matriz `anio_contrato` vs `renta` no se diferencia bien el cluster 4 o en la matriz `alquiler_actual` vs `ingresos_minimos` no se diferencia bien el cluster 2.

Eso se debe a que realmente son proyecciones 2D sobre agrupaciones de ejemplos 4-dimensionales, por lo que es posible que unos ejemplos sean ocultados por otros, provocando dichos efectos. A pesar de ello, generalmente las fronteras intercluster están claras y se diferencian bien, evidenciando que algunas fronteras como las de `renta` vs `ingresos_minimos` no están tan claras.

Esta diversidad en las fronteras se debe a que aunque los clusters están separados en general, realmente hay ejemplos ambiguos y la distancia intercluster no es demasiado grande. Este hecho lo representa de forma muy acertada el gráfico de distancias intercluster de la figura 0.5, que representa el tamaño y la distancia entre sí de los clusters.

Como vemos, los clusters aunque están separados, tienen fronteras ambiguas, sobre todo el cluster 4 y 5 (3 y 4 en la matriz de dispersión) que realmente puede que tengan pocas diferencias con respecto al cluster 1 (0 en la matriz de dispersión).

## Interpretación de los resultados

Procedemos a interpretar los resultados que nos ha dado esta ejecución. Recordemos que tratabamos con las variables '`renta`' que corresponde a la renta disponible total del hogar en el año anterior al de encuesta; '`anio_contrato`' que corresponde a año del contrato o de la compra o de instalación; '`ingresos_minimos`' que corresponde a los ingresos mínimos para llegar a final de mes y `alquiler_actual` que corresponde al alquiler actual por la vivienda ocupada.

En la matriz de dispersión, en la posición 4x4 (asumiendo que comenzamos la numeración en el 1) apreciamos la llamativa distribución del alquiler, muy concentrada entorno a los 500€, la cual ya supone una diferencia clara entre los que pagan menos de ese valor(cluster 4, rosa), los que pagan más (cluster 3, verde) y los que sí pagan en torno a 500. En las gráficas superiores se puede apreciar de hecho una clara diferencia entre el rosa, el verde, y una mayor mezcla entre azul claro, azul oscuro y verde claro.

Si observamos el año del contrato comparado con los ingresos mínimos necesarios para llegar a fin de mes se diferencian mucho mejor los clusters que en el alquiler no se apreciaban. Por un lado están los que necesitan una cantidad mayor de ingresos (azul oscuro) y los que necesitan una cantidad menor (verde claro y azul claro), dentro de los cuales están los que tienen un contrato más antiguo (verde claro) y los que tienen uno más reciente (azul, solapado además con el rosa, que se puede apreciar al comparar el año con el alquiler).

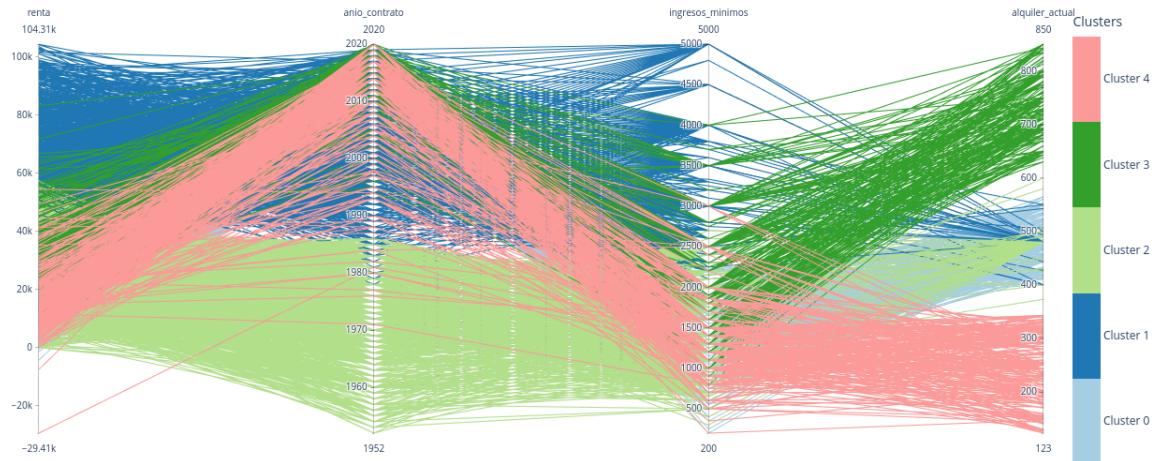


Figura 0.3: Gráfico de coordenadas paralelas según los clusters de K-Means

Por tanto, dentro del grupo de encuestados que paga en torno a unos 500€ al mes están los que requieren una mayor cantidad de ingresos para llegar a fin de mes que además si nos fijamos coinciden con los que tienen la renta total más alta; y los que no requieren tantos ingresos. Los primeros (cluster azul oscuro), si nos fijamos, tienen una cierta tendencia a firmar contratos recientes, quizás porque la gente con renta más alta en 2020 pudo permitirse cambiar de casa pocos años atrás. Por otro lado, la gente que requiere una cantidad menor de ingresos vemos que tiene rentas más bajas, siendo difícil distinguir el cluster azul del verde claro. La principal diferencia se encuentra en el año de firma del contrato.

Todas estas claras diferencias ya comentadas pueden además verse reforzadas y recogidas en el heatmap de centroides de la figura 0.6, aunque recordemos que este gráfico sólo recoge datos respecto a la media, no respecto a la dispersión. Aún así, coincide con las conclusiones extraídas.

Caso de estudio: Encuestados que no viven en zonas poco pobladas

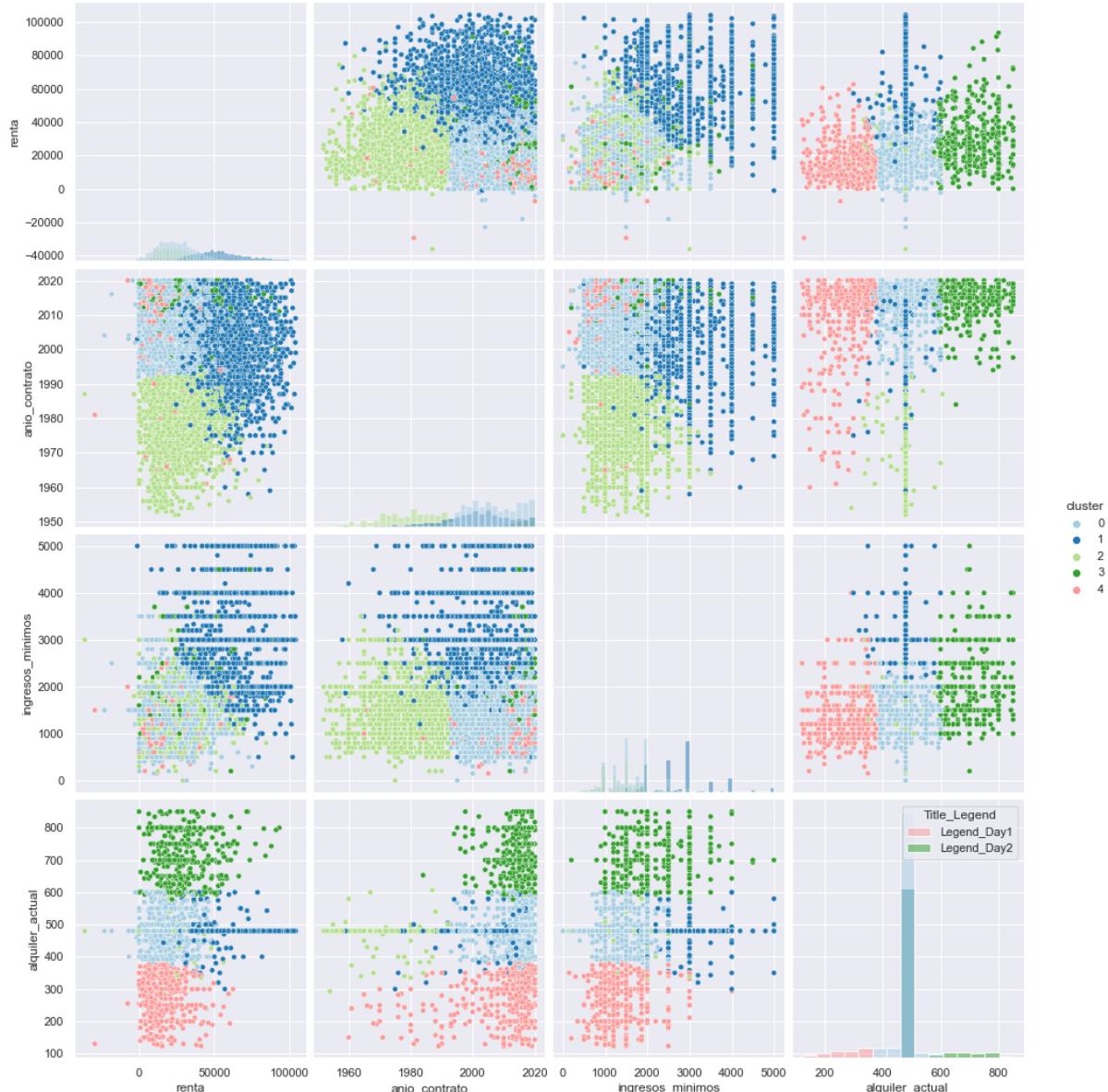


Figura 0.4: Matriz de dispersión coloreada según los clusters de K-Means

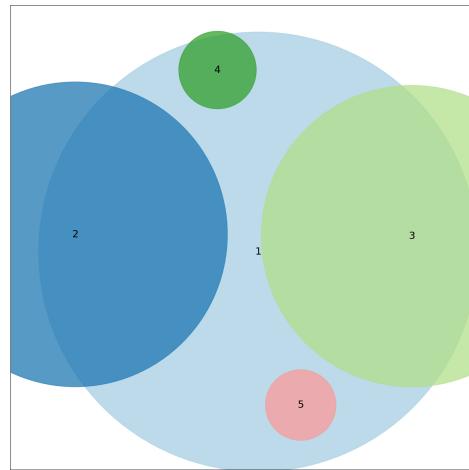


Figura 0.5: Gráfico de distancia intercluster de K-Means

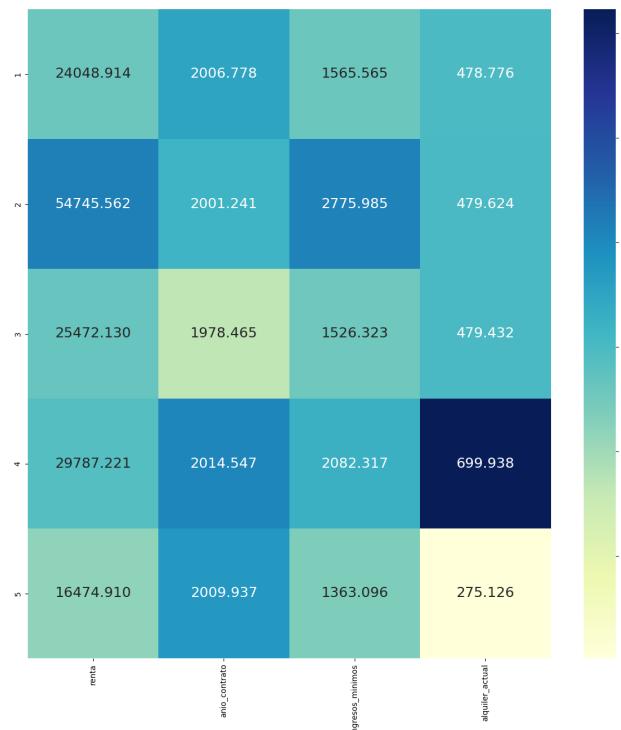


Figura 0.6: Heatmap de centroides de K-Means



# Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler

Nos centraremos ahora en los encuestados que viven en pisos de alquiler o realquiler. Para ello nos ayudaremos filtrando el conjunto de datos completo con la variable **HH021: Régimen de tenencia**, que toma el valor 3 cuando el encuestado vive en un piso de alquiler o realquiler a precio de mercado y el valor 4 en caso de que el precio sea inferior al de mercado. A nosotros nos interesan por tanto las filas del *dataset* en las que esta variable tome el valor 3 ó 4. Es este caso trabajaríamos con un subconjunto de 2315 filas, el 15.39% del total de filas. Se trata de un subconjunto más pequeño que el primer caso, pero veremos que es suficientemente grande para el trabajo que nos ocupa.

En este caso haremos clustering sobre las siguientes variables:

- HS130: Ingresos mínimos para llegar a final de mes. Renombrada como **ingresos\_minimos**.
- vhRentaa: Renta disponible total del hogar en el año anterior al de la entrevista. Renombrada como **rentaa**.
- HH060: Alquiler actual por la vivienda ocupada. Renombrada como **alquiler**.
- HH061: ¿Cuál cree usted que sería el importe mensual que tendría que pagar por el alquiler de una vivienda como ésta a precio de mercado? Renombrada como **importe justo**.
- HY070G: Ayuda para vivienda en el año anterior al de encuesta. Renombrada como **ayuda**.

## Discusión de parámetros

Hemos seguido la misma metodología en este conjunto de datos que en el anterior, y también en el siguiente. Mostramos los resultados obtenidos:

### K-Means

Mostramos las medidas de rendimiento en función del valor de **n\_clusters**:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
n_clusters = 2	0.822764	717.129	0.272167	2	0.0982667
n_clusters = 3	0.811976	753.32	0.331899	3	3.75077
n_clusters = 4	1.09546	959.058	0.37878	4	14.4273
n_clusters = 5	1.00909	931.614	0.278027	5	3.63
n_clusters = 6	1.14646	883.803	0.283549	6	2.42107

## Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
n_clusters = 7	0.434081	841.572	0.277827	7	1.336
n_clusters = 8	0.466454	771.127	0.237115	8	0.591056
n_clusters = 9	0.48834	776.735	0.253786	9	0.274296
n_clusters = 10	0.470694	728.62	0.245741	10	0.162816
n_clusters = 11	0.50352	689.709	0.239095	11	0.404486
n_clusters = 12	1.1739	701.804	0.256559	12	0.0384499
n_clusters = 13	1.16819	681.57	0.242197	13	0.359095
n_clusters = 14	0.960659	666.254	0.244005	14	0.423111
n_clusters = 15	0.492634	641.779	0.245845	15	0.631211
n_clusters = 16	0.981674	603.633	0.234732	16	1.52737
n_clusters = 17	1.63296	582.134	0.227412	17	2.25868
n_clusters = 18	0.664905	601.196	0.232778	18	1.64689
n_clusters = 19	0.643515	582.999	0.236843	19	1.8542
n_clusters = 20	0.458785	577.096	0.231967	20	2.16491

Al igual que en el caso anterior, parece que el coeficiente de Calinski-Harabasz y el Silhouette aumentan o disminuyen simultáneamente, marcando un claro máximo en `n_clusters = 4`:

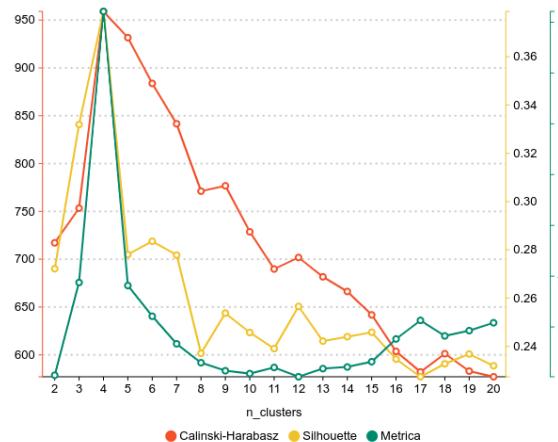


Figura 0.1: Gráfico comparativo del parámetro de K-Means

La métrica descrita también nos marca un máximo en `n_clusters = 4`, por lo que consideramos dicho valor como el óptimo.

## DBSCAN

Procediendo como en el caso anterior, hemos obtenido la siguiente tabla:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
eps = 0.5, min_samples = 1	0.0861559	9.96026	-0.343908	319	10.613
eps = 1.0, min_samples = 1	0.100555	4.34865	-0.353386	62	11.0455
eps = 1.5, min_samples = 1	0.244581	5.63604	0.0728386	15	1.96614
eps = 2.0, min_samples = 1	0.299835	5.47125	0.489777	2	1.55764
eps = 0.5, min_samples = 3	0.100063	28.5634	-0.216601	26	6.70263
eps = 1.0, min_samples = 3	0.100924	19.0079	0.274985	5	0.429379
eps = 1.5, min_samples = 3	0.140998	16.1192	0.363811	3	0.560832
eps = 2.0, min_samples = 3	0.138024	5.47125	0.489777	2	1.55764
eps = 0.5, min_samples = 5	0.05844	43.5738	0.122698	11	0.847498

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
eps = 1.0, min_samples = 5	0.107312	117.385	0.33771	4	0.703838
eps = 1.5, min_samples = 5	0.139446	16.82	0.357222	3	0.528148
eps = 2.0, min_samples = 5	0.187871	5.74286	0.474304	2	1.41707
eps = 0.5, min_samples = 7	0.107244	72.7831	0.271347	5	0.0472372
eps = 1.0, min_samples = 7	0.106187	159.751	0.388328	3	2.2821
eps = 1.5, min_samples = 7	0.097234	20.311	0.424804	2	0.749844
eps = 2.0, min_samples = 7	0.111909	5.74286	0.474304	2	1.41707
eps = 0.5, min_samples = 9	0.0552914	89.0498	0.266319	4	0.189832
eps = 1.0, min_samples = 9	0.0761926	164.351	0.386208	3	2.47638
eps = 1.5, min_samples = 9	0.101239	22.2303	0.41707	2	0.670958
eps = 2.0, min_samples = 9	0.113905	10.3712	0.41514	2	0.900843
eps = 0.5, min_samples = 11	0.0621951	63.0707	0.109216	5	0.891458
eps = 1.0, min_samples = 11	0.174437	157.108	0.380851	3	2.14002
eps = 1.5, min_samples = 11	0.108469	22.2303	0.41707	2	0.670958
eps = 2.0, min_samples = 11	0.123471	10.3712	0.41514	2	0.900843
eps = 0.5, min_samples = 13	0.0668175	161.037	0.300675	2	2.15877
eps = 1.0, min_samples = 13	0.0911279	152.614	0.377042	3	1.94192
eps = 1.5, min_samples = 13	0.151298	17.8838	0.412903	2	0.729263
eps = 2.0, min_samples = 13	0.118615	10.3712	0.41514	2	0.900843
eps = 0.5, min_samples = 15	0.0538967	165.345	0.288784	2	2.35516
eps = 1.0, min_samples = 15	0.0887775	136.822	0.362458	3	1.31856
eps = 1.5, min_samples = 15	0.097997	21.4271	0.43087	2	0.766124
eps = 2.0, min_samples = 15	0.110973	10.3712	0.41514	2	0.900843
eps = 0.5, min_samples = 17	0.0591142	104.629	0.120639	3	1.18668
eps = 1.0, min_samples = 17	0.118321	227.05	0.35485	4	6.07201
eps = 1.5, min_samples = 17	0.134829	20.73	0.426388	2	0.75153
eps = 2.0, min_samples = 17	0.154063	10.3712	0.41514	2	0.900843
eps = 0.5, min_samples = 19	0.0593348	103.875	0.112474	3	1.24533
eps = 1.0, min_samples = 19	0.0857577	225.088	0.344769	4	5.9068
eps = 1.5, min_samples = 19	0.110563	34.1105	0.449613	2	0.697659
eps = 2.0, min_samples = 19	0.16902	10.3712	0.41514	2	0.900843

Podemos en este caso observar que al igual que en el caso anterior las configuraciones que consiguen un coeficiente silhouette más alto muestran un C-H más bajo (y un número de clusters más pequeño) y viceversa. Por ello de nuevo debemos utilizar la métrica para poder decidir un equilibrio, ignorando aquellas que muestren silhouette negativo, nos volvemos a ayudar del heatmap de la figura 0.2.

Si nos fijamos en el mapa de calor, la configuración con mayor métrica y con silhouette positivo es `eps = 1, min_samples = 17`, que ofrece un silhouette considerable, de 0.3548 y un coeficiente de C-H que de hecho es el mayor de todos: 227.05, por lo que parece ser la mejor configuración posible.

## Ejecución de los algoritmos

Con los datos y las variables seleccionadas y elegidos parámetros óptimos para *K-Means* y *DBSCAN* procedemos a ejecutar y comparar los distintos algoritmos, obteniendo la siguiente tabla comparativa de las distintas medidas de rendimiento:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters
K-Means	0.484145	959.058	0.37878	4
Mean Shift	0.836446	310.047	0.329927	9
DBSCAN	0.101328	227.05	0.35485	4
Spectral Clustering	2.38311	558.844	0.471986	3
Agglomerative Clustering	0.163604	555.199	0.258053	2

## Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler

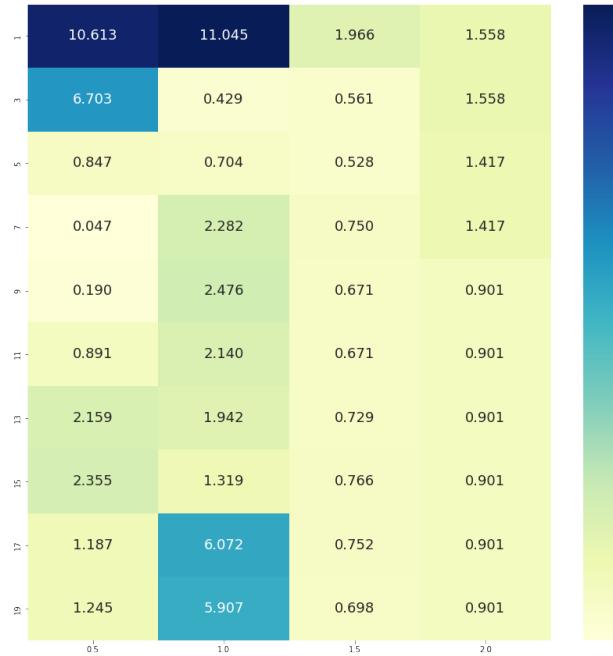


Figura 0.2: Heatmap parámetros DBSCAN

En este caso se nota bastante que el conjunto de datos es más pequeño, pues los tiempos de ejecución no llegan al segundo salvo *Spectral Clustering* que vimos en el primer caso de estudio que era más lento. Aún así, estos 2.38 segundos distan mucho de los cerca de 40 segundos que tardó e ejecutar el clustering en el primer caso. *DBSCAN* ha sido el más rápido en este caso de nuevo, pero tiene un coeficiente de C-H muy bajo y un silhouette moderado, veamos cómo queda su distribución

Como vemos de nuevo concentra una gran mayoría de los ejemplos en uno de los clusters, pero hay un mejor reparto que en el caso anterior, veamos la tabla:

	Tamaño	Porcentaje
0	1748	82.2588
1	203	9.55294
2	115	5.41176
3	59	2.77647

La gran mayoría de los items está en el cluster 0, el azul oscuro, pero no es un desbalanceo tan pronunciado como el 93.89% conseguido en el primer caso.

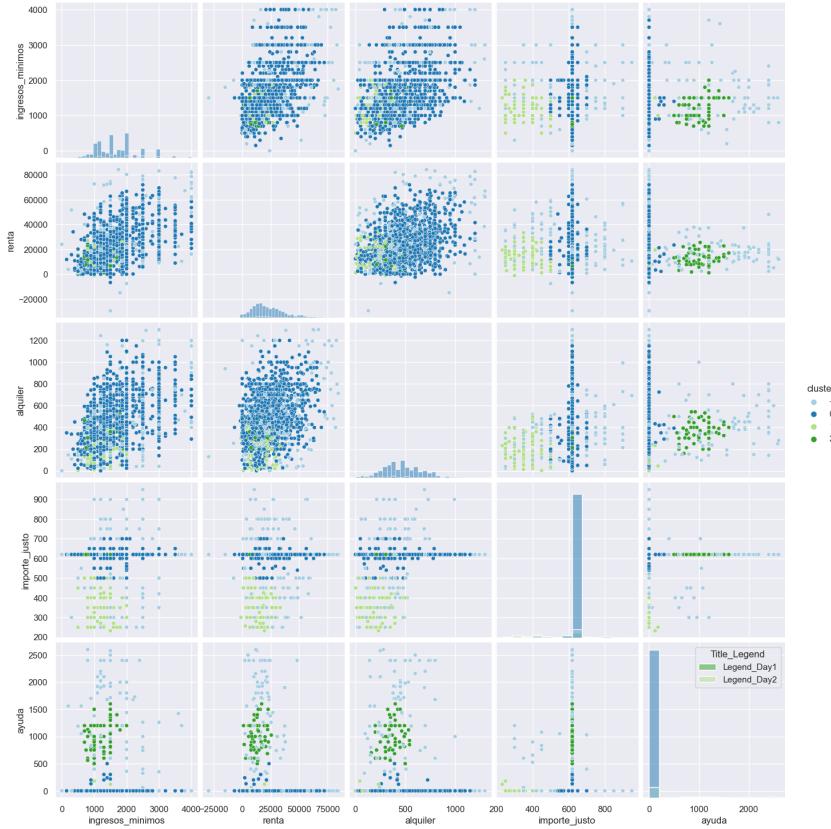


Figura 0.3: Matriz de dispersión de DBSCAN

Respecto a los demás algoritmos, entraremos en detalle en el rendimiento de *Spectral Clustering*, pero antes comentaremos el gran número de clusters conseguidos por *Mean Shift*, que si miramos la tabla de distribución de los items por clusters vemos que muchos de estos clusters son muy pequeños, por lo que igual no merece la pena considerarlos por separado:

	Tamaño	Porcentaje
0	1678	78.9647
1	159	7.48235
2	149	7.01176
3	62	2.91765
4	46	2.16471
5	15	0.705882
6	10	0.470588
7	4	0.188235
8	2	0.0941176

En efecto, a partir del cluster 4 no llegan a agrupar un 1% cada cluster. Este resultado se puede ver reflejado en unos coeficientes de C-H y silhouette bajos comparados con los otros algoritmos.

## Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler

Por su parte, *K-Means* ha conseguido un buen rendimiento en lo que se refiere al coeficiente de C-H y un silhouette razonable, ofreciéndonos una distribución que en el siguiente gráfico se puede ver que en lo respectivo a tamaño de los clusters y distancia inter-cluster es muy buena.

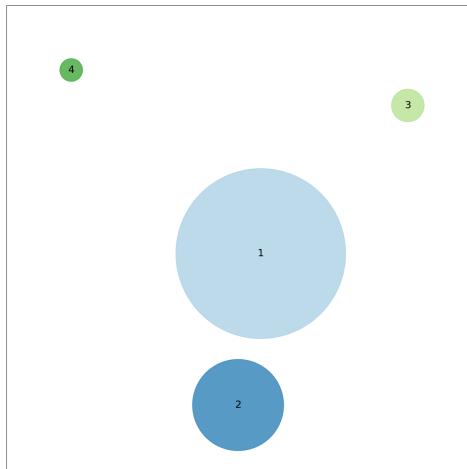
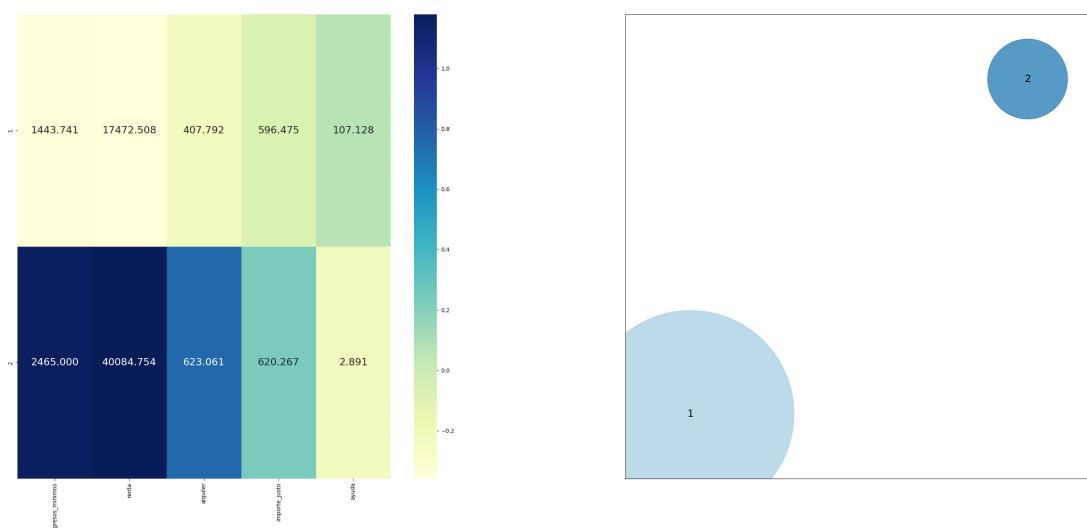


Figura 0.4: Distancia y tamaño de los clusters en K-Means

Respecto al clustering aglomerativo salta a la vista que el resultado es muy pobre, tiene la medida de silhouette más baja y el segundo C-H más bajo. Sin embargo, aunque solo ha hecho dos clusters los ha separado bastante bien. Vemos en los siguientes gráficos que ha agrupado a los items con valores de `ingresos_minimos`, `renta` y `alquiler` más altos y a los que tienen medidas más moderadas en otro cluster, estando estos dos grupos bien separados.



## Interpretación de resultados

Pasaremos a interpretar los resultados que nos ha dado el clustering espectral. Para ello en primer lugar presentamos la matriz de dispersión:



Figura 0.5: Matriz de dispersión de Spectral Clustering

Si consultamos el conjunto de datos, la variable `importe_justo` tiene un alto porcentaje de valores perdidos, casi el 84% de hecho, por eso la media supone una concentración tan alta. Por otra parte, vemos que la ayuda para la vivienda está muy sesgada hacia el valor 0, ya que lo normal es no tener ninguna ayuda para la vivienda.

Por otra parte y como veníamos comentando, vemos una alta presencia de puntos azules que corresponden al cluster 0, que es de hecho el mayoritario, podemos observarlo en el gráfico siguiente:

## Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler

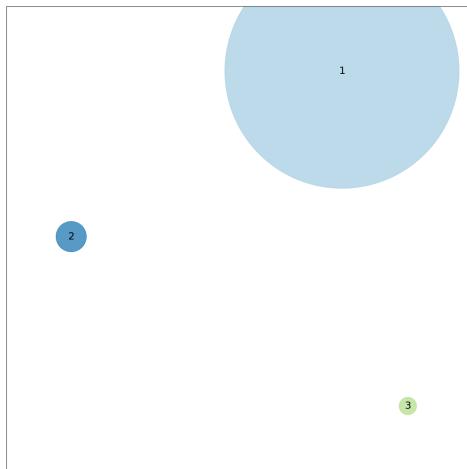


Figura 0.6: Distancia intercluster de Spectral Clustering

Observamos que los clusters están muy separados, pero el cluster 3 (2 en la scatter matrix) es muy pequeño en comparación con el 1. Sin embargo, estos clusters aunque minoritarios tienen sus características propias y distinguidas. La diferencia principal del cluster azul oscuro se observa en la variable `ayuda`, y vemos que son los que necesitan una mayor cantidad de ayudas para la vivienda, ya que los vemos muy presentes en la parte alta de la última fila de gráficos de la matriz de dispersión. Sin embargo, en el resto de celdas se ve más disperso y se confunde más con el cluster azul claro.

Por su parte, el cluster verde claro es el que tiene la variable `importe_justo` más baja. Es decir, los que piensan que su casa a precio de mercado tiene un precio barato. Es curioso cómo si nos fijamos en la celda de `importe_justo` vs `alquiler` la mayoría de los items están por debajo de la diagonal, es decir, piensan que el importe a precio de mercado es mayor que su alquiler actual. De todas formas recordamos que las conclusiones que impliquen la variable `importe_justo` no son del todo fiables por la gran cantidad de valores perdidos, aunque los items del cluster verde claro no tienen esa variable como perdida.

Y el resto de valores han sido agrupados todos en el cluster azul claro, que es de hecho el más grande. Podemos ver si nos fijamos en el menor principal de orden 3 que los puntos azules claro se distribuyen por todo el plano. Hay que tener en cuenta que a Spectral Clustering, al igual que a *K-Means* hay que especificarle cuántos clusters queremos que genere, y en este caso se han generado 3, pero parece que el cluster azul pide una posible separación.

Más allá de la separación por importe justo y por ayuda para la vivienda no parece que nos haya dejado buen sabor de boca esta interpretación, por lo que de nuevo consultaremos los resultados de *K-Means* para cerciorarnos de lo concluido hasta ahora y ver si podemos extraer más conclusiones, pues *K-Means* separó en cuatro clusters, no en tres, obsérvese la matriz de dispersión en la figura 0.7.

Vemos que *K-Means* está de acuerdo con separar un cluster con los que tuvieron mayor

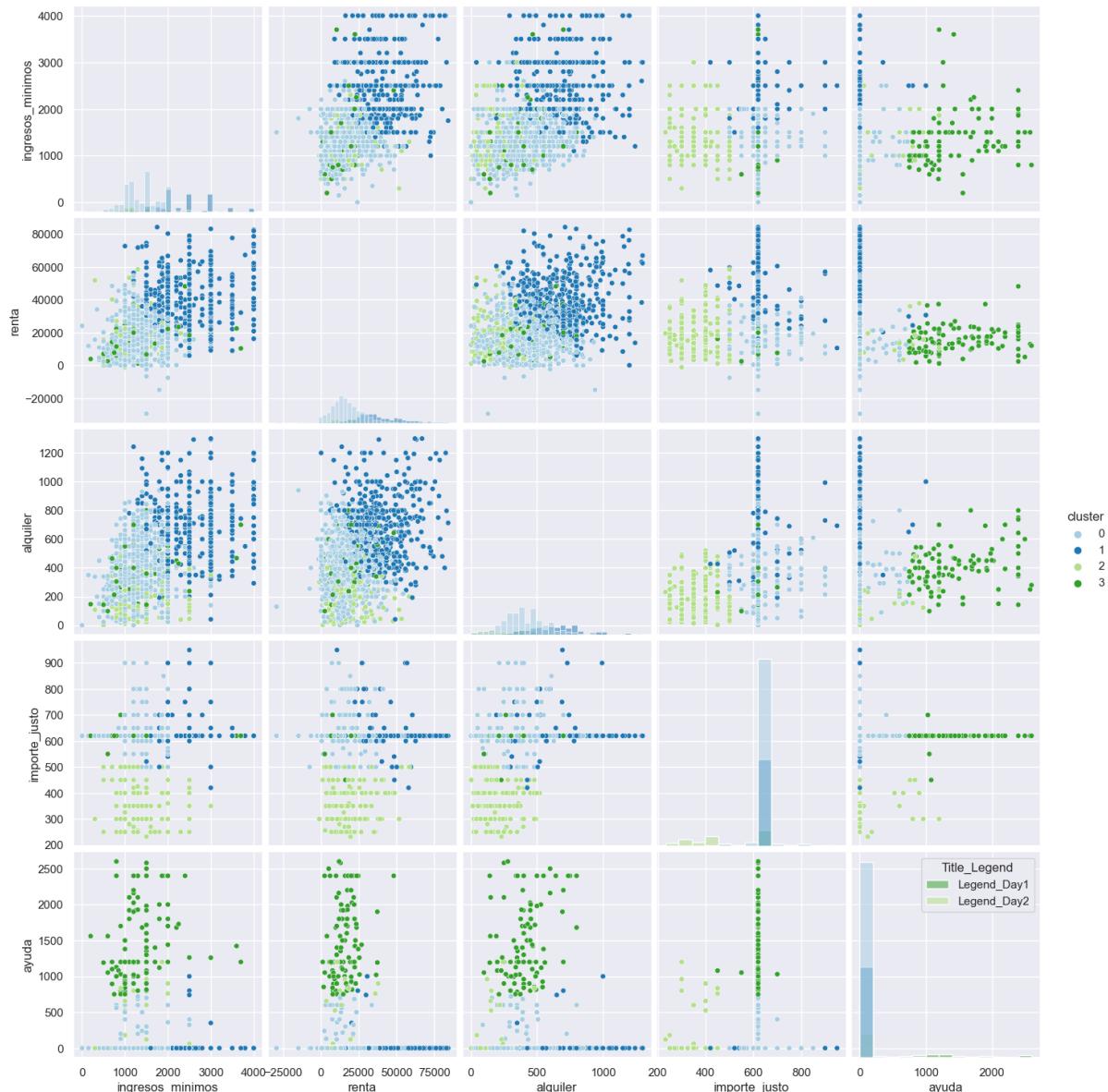


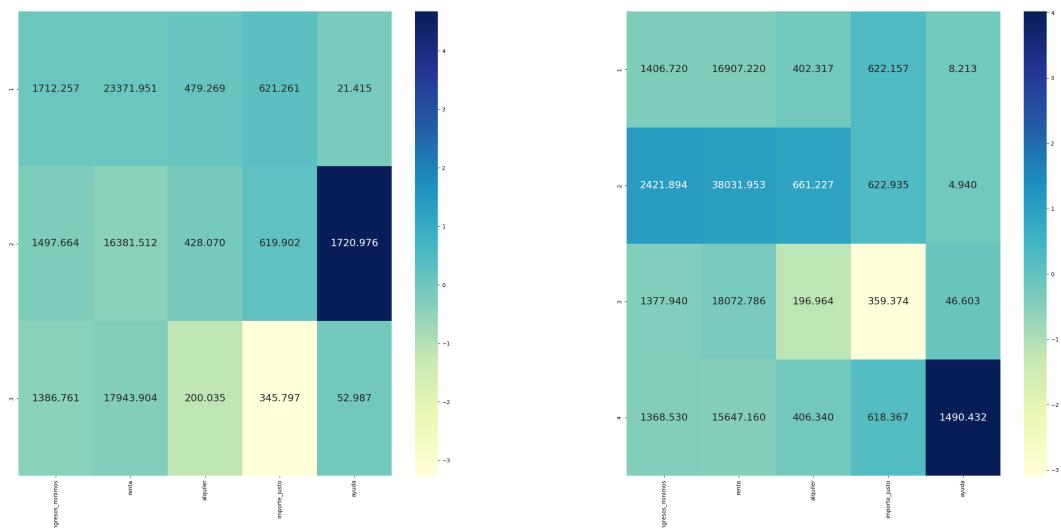
Figura 0.7: Matriz de dispersión de K-Means

## Caso de estudio: Encuestados que viven en pisos de alquiler o realquiler

ayuda para la vivienda y los que consideran que el valor de mercado del alquiler es bajo. Sin embargo, parece que el cuarto cluster que a *Spectral Clustering* no se le especificó y a *K-Means* sí se hace notar en este caso, pues vemos una nube de puntos azul oscuro diferenciarse de la nube de puntos azul claro, que juntos forman (aproximadamente) el cluster mayoritario en los resultados del clustering espectral. La principal distinción, de forma similar a la que ocurría en el primer caso de estudio, se presenta en los ingresos mínimos y la renta, que toman valores más altos, pero en este caso los alquileres también son más altos.

Es decir, se ha dividido el cluster azul claro de *Spectral Clustering* en 2 clusters, uno que agrupa a los items con mayor renta, y por tanto que se pueden permitir pagar un mayor alquiler por un piso (o casa) mejor y que por tanto puede requerir unos ingresos mayores a fin de mes. Por otro lado tenemos el cluster azul claro que aunque cuesta más distinguirlo entre los demás, puede destacar porque en la ayuda para la vivienda se pueden diferenciar fácilmente los puntos azules en las zonas más bajas.

Toda esta información podemos verla recogida en los heatmap de *Spectral Clustering* y de *K-Means*, respectivamente:



Como conclusión quiero remarcar que las conclusiones pueden extraerse no solo a raíz de los resultados de un algoritmo por muy buenos que sean, sino que con las salidas de varios algoritmos podemos también extraer información valiosa.

# Caso de estudio: Encuestados con mayor renta

Bien es sabido que la renta total es muy variada y condiciona muchas de las demás variables, porque siendo sinceros, las condiciones de vida muchas veces dependen del dinero que se tenga y del que se ingrese a corto plazo. Por ello, ahora hemos querido estudiar a aquellos individuos que tengan la renta más alta. Para ello hemos vuelto a usar la variable **vhRentaa: Renta disponible total del hogar en el año anterior al de la entrevista**, seleccionando aquellos individuos cuyo valor de esta variable sea superior a la mediana. Es decir, el 50% con la renta más alta, asegurando así que tendremos una muestra suficientemente representativa. Trabajamos por tanto ahora con un *dataset* de 7521 filas, que naturalmente representan un 50% del total.

Para hacer clustering, emplearemos las variables que presentamos a continuación:

- HH031: Año del contrato o de la compra o de instalación. Renombrada como `anio_contrato`.
- HY040N: Renta neta procedente del alquiler de una propiedad o terreno en el año anterior al de encuesta. Renombrada como `renta_alquiler`.
- HH030: Numero de habitaciones de la vivienda. Renombrada como `n_habitaciones`.
- 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.). Renombrada como `gastos_vivienda`.
- HV010: ¿Cuál es el valor actual de la vivienda principal? Renombrada como `valor_actual`.

Hemos hecho un estudio de valores perdidos considerando que más de 3000 valores perdidos sería una cantidad significativa y ninguna de las variables elegidas tiene 3000 o más valores perdidos sobre este subconjunto de datos.

## Discusión de parámetros

Al igual que en los casos anteriores, hemos hecho ejecuciones de *K-Means* y de *DBSCAN* modificando los parámetros y nos hemos quedado con las mejores opciones. A continuación exponemos los resultados.

### K-Means

Las medidas de rendimiento en función del parámetro `n_clusters` fueron:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
<code>n_clusters = 2</code>	0.652612	1668.75	0.274074	2	5.80194
<code>n_clusters = 3</code>	0.737241	1764.78	0.271731	3	5.88365
<code>n_clusters = 4</code>	0.876824	1792.46	0.206926	4	1.45126
<code>n_clusters = 5</code>	0.584424	1856.37	0.235795	5	2.07841
<code>n_clusters = 6</code>	0.884014	2028.02	0.25617	6	6.3121
<code>n_clusters = 7</code>	1.49973	1866.63	0.237171	7	2.26907
<code>n_clusters = 8</code>	1.54012	1726.72	0.222993	8	0.594136
<code>n_clusters = 9</code>	1.58544	1659.89	0.216947	9	0.28144
<code>n_clusters = 10</code>	1.62969	1555.88	0.199263	10	0.938645
<code>n_clusters = 11</code>	0.925318	1494.83	0.20982	11	0.274204
<code>n_clusters = 12</code>	0.649244	1459.82	0.203523	12	0.695063
<code>n_clusters = 13</code>	1.20058	1409.11	0.205179	13	0.750843
<code>n_clusters = 14</code>	2.37154	1355.89	0.205281	14	1.00576
<code>n_clusters = 15</code>	1.64016	1336.35	0.210843	15	0.840846
<code>n_clusters = 16</code>	1.62435	1299.93	0.210787	16	1.09705
<code>n_clusters = 17</code>	1.27216	1267.5	0.200714	17	1.96414
<code>n_clusters = 18</code>	1.1544	1230.45	0.215473	18	1.55391
<code>n_clusters = 19</code>	1.12562	1204.75	0.207949	19	2.09191
<code>n_clusters = 20</code>	2.05847	1196.01	0.209394	20	2.11561

En este caso vemos mayor variedad y duda que en los anteriores, pues el caso con mayor coeficiente de C-H y el caso con mayor coeficiente silhouette no coinciden. Observemos ahora el gráfico comparativo de la figura 0.1.

Ahora el pico de la métrica y el coeficiente de C-H coinciden, pero no con el de silhouette. Además, coinciden en un número de clusters quizás muy alto, 6. La otra opción es utilizar 3 clusters, ya que es un número más interpretable, más manejable y que presenta el segundo valor más alto de la métrica. Por tanto ejecutaremos *K-Means* con el parámetro `n_clusters = 3`.

## DBSCAN

De nuevo variando los parámetros `eps` y `min_samples`, hemos obtenido la siguiente tabla:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
<code>eps = 0.5, min_samples = 1</code>	0.375295	10.2692	-0.366891	872	6.27831
<code>eps = 1.0, min_samples = 1</code>	0.474448	19.0186	-0.097054	160	2.80406
<code>eps = 1.5, min_samples = 1</code>	1.09341	8.63812	0.261284	21	1.28621

## Ejecución de los algoritmos

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters	Metrica
eps = 2.0, min_samples = 1	1.5578	8.28276	0.47997	4	1.90333
eps = 0.5, min_samples = 3	0.400427	61.3052	-0.231829	60	3.86752
eps = 1.0, min_samples = 3	0.808973	174.507	0.0436174	14	0.720147
eps = 1.5, min_samples = 3	1.72104	51.1726	0.485254	3	1.51316
eps = 2.0, min_samples = 3	2.02726	14.3377	0.564107	2	2.39929
eps = 0.5, min_samples = 5	0.29938	150.358	-0.127777	22	2.13302
eps = 1.0, min_samples = 5	0.638903	560.277	0.104288	5	2.49425
eps = 1.5, min_samples = 5	1.21877	118.643	0.531064	2	1.30814
eps = 2.0, min_samples = 5	1.01357	18.2271	0.55204	2	2.26743
eps = 0.5, min_samples = 7	0.5315	270.204	-0.0158216	11	0.982484
eps = 1.0, min_samples = 7	0.617375	587.496	0.105424	5	2.88725
eps = 1.5, min_samples = 7	1.73828	134.127	0.532385	2	1.23272
eps = 2.0, min_samples = 7	1.74406	32.8432	0.567122	2	2.23638
eps = 0.5, min_samples = 9	0.264134	300.861	-0.0589531	9	1.36088
eps = 1.0, min_samples = 9	0.655114	383.218	0.0621181	8	0.909617
eps = 1.5, min_samples = 9	1.30768	201.446	0.524364	2	0.935402
eps = 2.0, min_samples = 9	2.08538	32.8432	0.567122	2	2.23638
eps = 0.5, min_samples = 11	0.509683	460.571	-0.00189462	6	1.88751
eps = 1.0, min_samples = 11	0.550314	524.133	0.0802811	6	2.12052
eps = 1.5, min_samples = 11	1.07313	255.895	0.527034	2	0.905837
eps = 2.0, min_samples = 11	1.61496	32.8432	0.567122	2	2.23638
eps = 0.5, min_samples = 13	0.594752	450.414	0.0234946	6	1.63173
eps = 1.0, min_samples = 13	0.526653	662.751	0.107131	5	4.15245
eps = 1.5, min_samples = 13	1.21475	279.293	0.522934	2	0.898603
eps = 2.0, min_samples = 13	1.28873	41.4121	0.571592	2	2.19045
eps = 0.5, min_samples = 15	0.31228	374.553	-0.00747495	7	1.27187
eps = 1.0, min_samples = 15	0.922935	542.087	0.0878424	6	2.31594
eps = 1.5, min_samples = 15	0.974567	289.815	0.514148	2	0.85805
eps = 2.0, min_samples = 15	0.943222	41.4121	0.571592	2	2.19045
eps = 0.5, min_samples = 17	0.252476	432.681	0.0314966	6	1.43052
eps = 1.0, min_samples = 17	0.832875	629.797	0.0941775	5	3.61692
eps = 1.5, min_samples = 17	1.11783	296.512	0.51061	2	0.848872
eps = 2.0, min_samples = 17	1.52231	49.0627	0.55392	2	1.98485
eps = 0.5, min_samples = 19	0.266361	415.929	0.0268887	6	1.32535
eps = 1.0, min_samples = 19	0.785456	632.37	0.0927353	5	3.66723
eps = 1.5, min_samples = 19	1.08394	310.336	0.506687	2	0.858333
eps = 2.0, min_samples = 19	1.30832	66.7265	0.556318	2	1.85215

En este caso las diferencias entre silhouette y C-H son muy abruptas, por lo que la métrica toma valores poco fiables. Muchas configuraciones toman valores muy altos de C-H pero silhouette muy cercano a cero o incluso negativos. Por ello, en la búsqueda de un término medio entre ambos casos hemos elegido la configuración `eps = 1.5, min_samples = 19`, ya que tiene un silhouette de 0.5 y un C-H de 310.336, valores que aunque no sean demasiado altos en comparación con otros sí son los más parejos, ofreciendo posiblemente un mejor rendimiento.

## Ejecución de los algoritmos

Para este caso los resultados obtenidos son los recogidos por la siguiente tabla:

	Tiempo	Calinski-Harabasz	Silhouette	Número de clusters
K-Means	0.480617	1764.78	0.271731	3
Mean Shift	2.36811	511.638	0.444564	3
DBSCAN	0.740452	310.336	0.506687	2
Spectral Clustering	16.6894	450.73	0.471258	3
Agglomerative Clustering	2.65285	1139.17	0.296723	2

## Caso de estudio: Encuestados con mayor renta

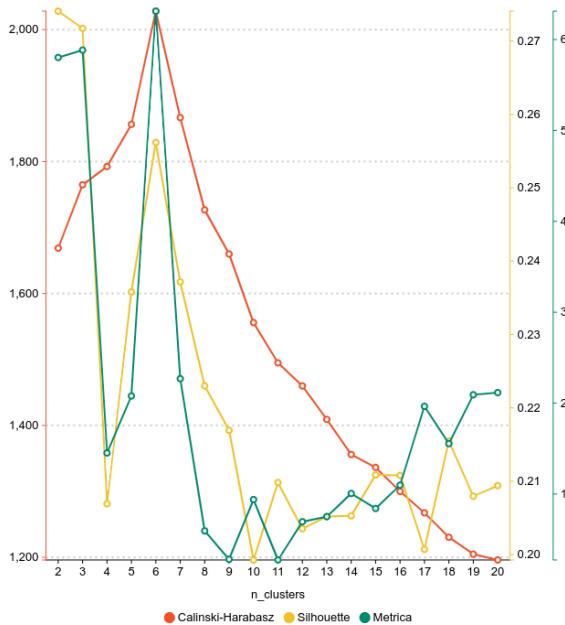


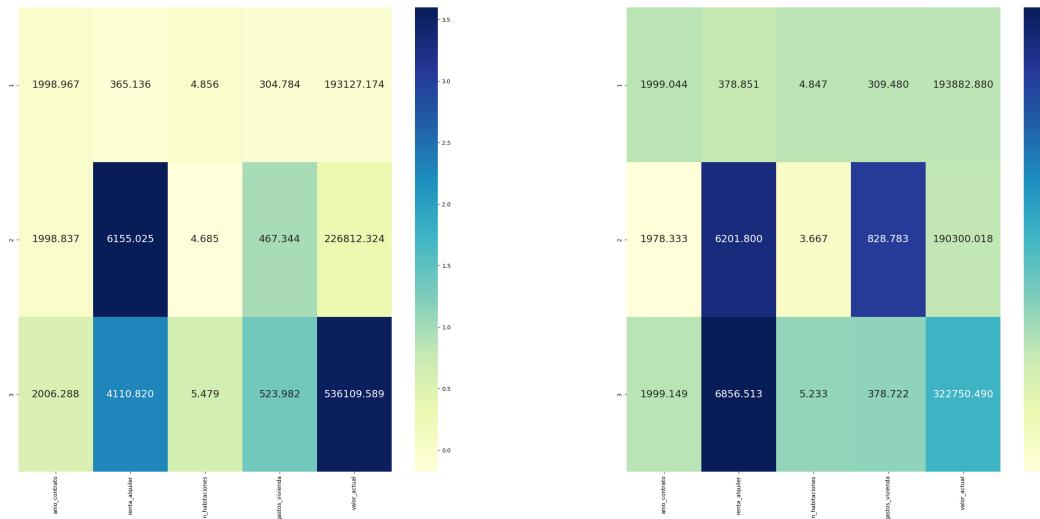
Figura 0.1: Gráfico comparativo del parámetro de K-Means

En este caso los tiempos de ejecución están más dispersos. Por ejemplo, *K-Means* tarda poco menos de medio segundo en ejecutar mientras que *Spectral Clustering* tarda aproximadamente 16.68 segundos. Si nos fijamos propiamente en las medidas, aparentemente *DBSCAN* tiene un silhouette muy alto, pero sólo dos clusters, lo cual nos dice que ha agrupado muchos items en un solo cluster y ha dejado los más lejanos como ruido. Recordemos que como `min_samples` tomamos el valor 19, por lo que probablemente en las ‘zonas más externas’ se hayan quedado muchos items sin asignar al cluster mayoritario. Vemos en la siguiente tabla que en efecto hay un cluster mayoritario:

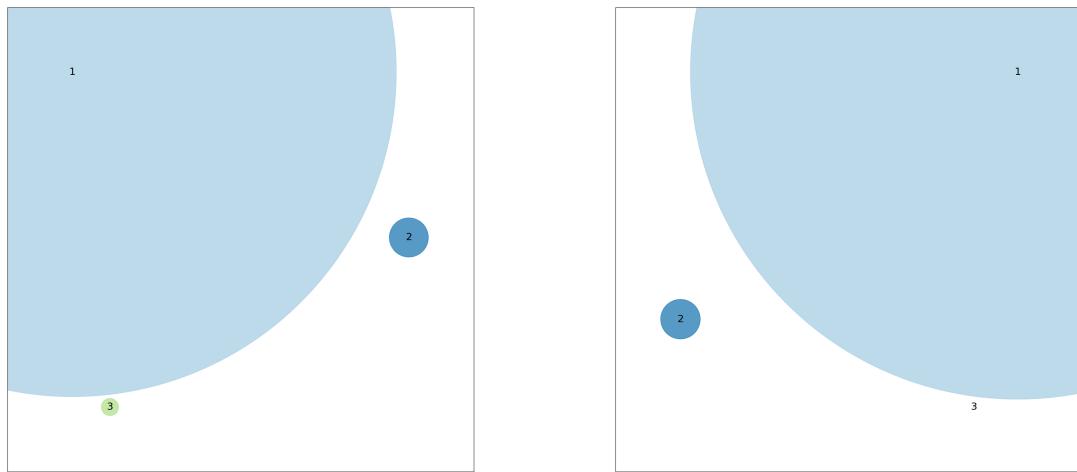
	Tamaño	Porcentaje
0	6983	98.8254
1	83	1.17464

Como conclusión, *DBSCAN* es un algoritmo que no nos ha ofrecido en estos casos buenos resultados en ningún caso ni ajustando los parámetros, aunque hay que destacar que aunque sea poco preciso es muy rápido.

Por su parte, *Mean Shift* y *Spectral Clustering* han obtenido resultados llamativamente similares en lo que corresponde a coeficientes de C-H y silhouette. De hecho, si comparamos el heatmap de *Mean Shift* y el de *Spectral Clustering*, representados abajo respectivamente, podemos ver ciertas similitudes entre uno y otro:



Podemos también presentar los gráficos de distancias intercluster para comparar el tamaño de los clusters y las distancias entre ellos:



Vemos que en efecto tienen la misma tendencia a agrupar la mayoría en un único cluster y el resto en dos clusters. Proximamente estudiaremos qué variables diferencian los clusters, pero podemos adelantar fijándonos en los heatmap que son principalmente **renta\_alquiler**, **gastos\_vivienda** y **valor\_actual**.

Sobre el clustering aglomerativo, confirmamos su tendencia a separar en dos clusters, aunque desde luego con más criterio que *DBSCAN*, y eso se puede ver reflejado en unas medidas de rendimiento cercanas a las de *K-Means*, que una vez más nos dará las más fáciles de interpretar. Para comprobarlo vemos el gráfico de coordenadas paralelas:

## Caso de estudio: Encuestados con mayor renta

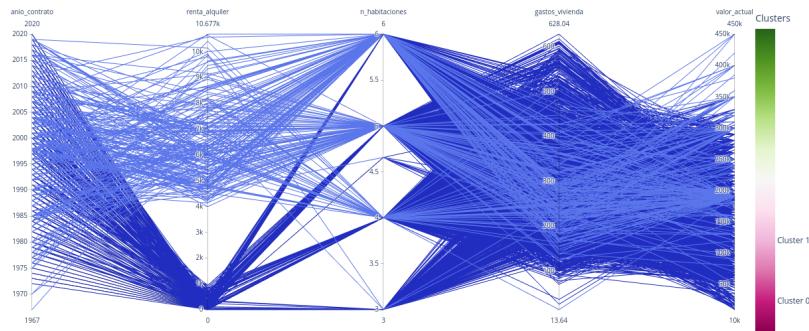


Figura 0.2: Gráfico de coordenadas paralelas de clustering aglomerativo

Vemos solapamientos, separaciones y una mayoría de líneas oscuras, que indican que el cluster 0 es el más grande.

## Interpretación de los resultados

Como en los casos anteriores, *K-Means* ha tenido un rendimiento muy bueno. Aunque no tenga un silhouette demasiado alto, lo cual también en cierto modo nos dice que no ha agrupado muchos items en un único cluster como *DBSCAN*, tiene el coeficiente de *Caliski-Harabasz* más alto, y comprobaremos en la matriz de dispersión que los resultados son razonables e interpretables (figura 0.3).

Lo primero que salta a la vista son las celdas que implican a la variable número de habitaciones, que al ser discreta tienen ese aspecto, pues dicha variable sólo toma 5 valores distintos.

Si intentamos distinguir propiedades de los clusters, podemos observar primero la variable `renta_alquiler`. Si observamos las celdas con dicha variable vemos que se diferencian muy bien los puntos azul claro, que representan a aquellos que reciben una renta alta debida al alquiler de una propiedad o terreno. Es llamativo también porque ese cluster es de hecho el mayoritario, fijémonos en la tabla:

	Tamaño	Porcentaje
0	4711	66.6714
1	1731	24.4976
2	624	8.83102

Por lo que podemos sacar la conclusión de que dentro de los encuestados con mayor renta total, la mayoría perciben una renta neta debida a propiedades alquiladas alta. Sin embargo, también hay muchos items agrupados en torno al 0, es decir, que reciben poco de las propiedades alquiladas o directamente no tienen propiedades alquiladas.

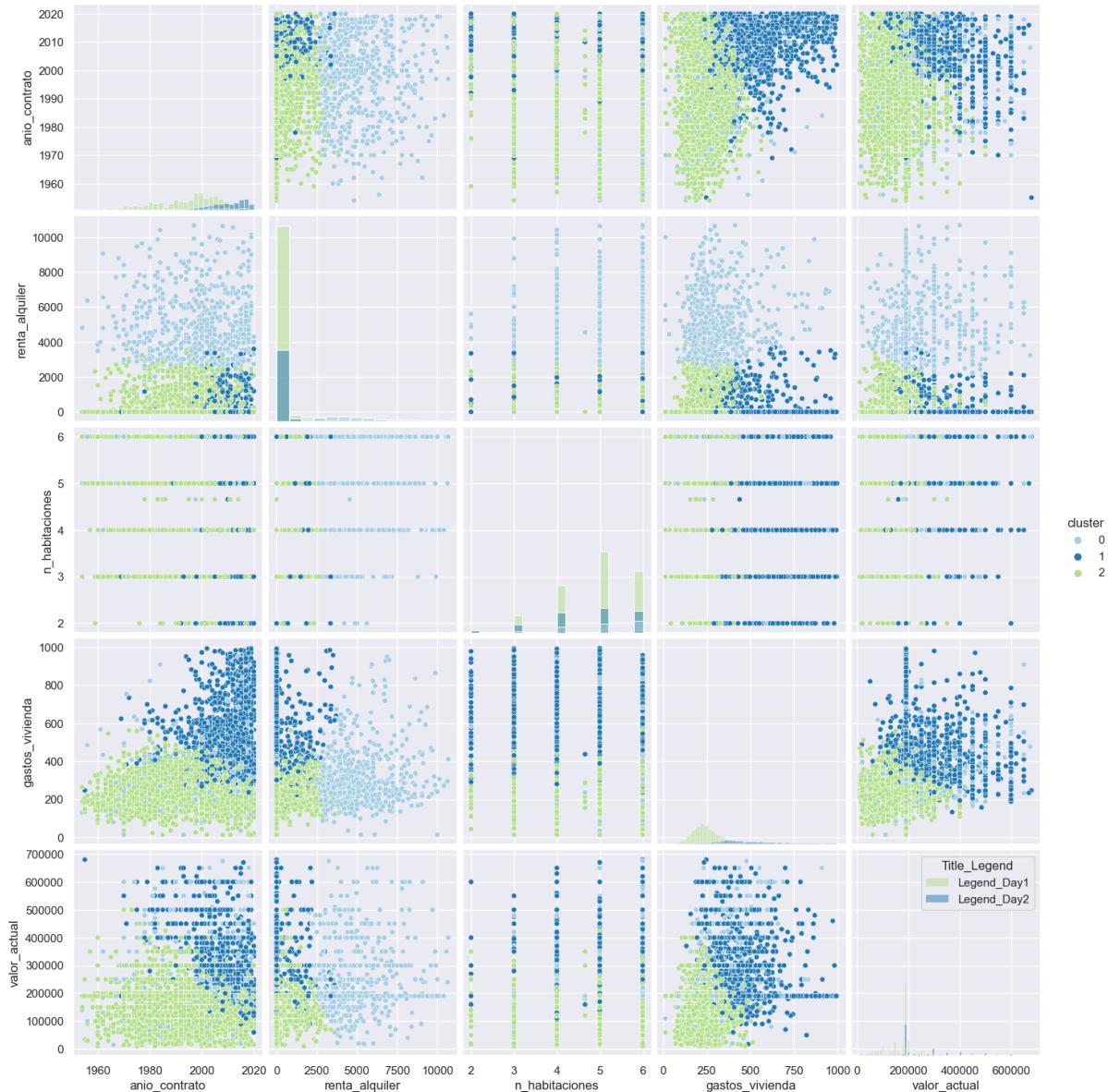


Figura 0.3: Matriz de dispersión de K-Means

## Caso de estudio: Encuestados con mayor renta

Los dos clusters restantes, que son el azul oscuro y el verde claro, podemos distinguirlos bien con los gastos asociados a la vivienda y el año del contrato. En las celdas correspondientes a estas dos variables vemos una alta agrupación de puntos azul oscuro entre los que tienen muchos gastos correspondientes a la vivienda y un año de contrato reciente, mientras que independientemente del año el cluster verde claro tiene menores gastos. Algo similar ocurre al observar los gastos de la vivienda con el valor actual de la vivienda principal, pero intercambiando los papeles. Observamos que los que tienen menos gastos pero también menor valor de la vivienda se agrupan en el cluster verde claro mientras que independientemente del valor actual de la vivienda los items del cluster azul oscuro tienden a tener mayores gastos, aunque la frontera no es recta, de hecho es más bien diagonal.

Donde podemos observar una muy buena segmentación es en la celda que encara a `renta_alquiler` con `gastos_vivienda`, donde podemos observar claramente los tres clusters bien separados con fronteras poco ambiguas.

Respecto al número de habitaciones realmente se puede comentar poco, ya que no parece suponer una diferencia representativa entre los distintos clusters.

Las ideas expuestas de nuevo se pueden ver recogidas en el heatmap presentado.

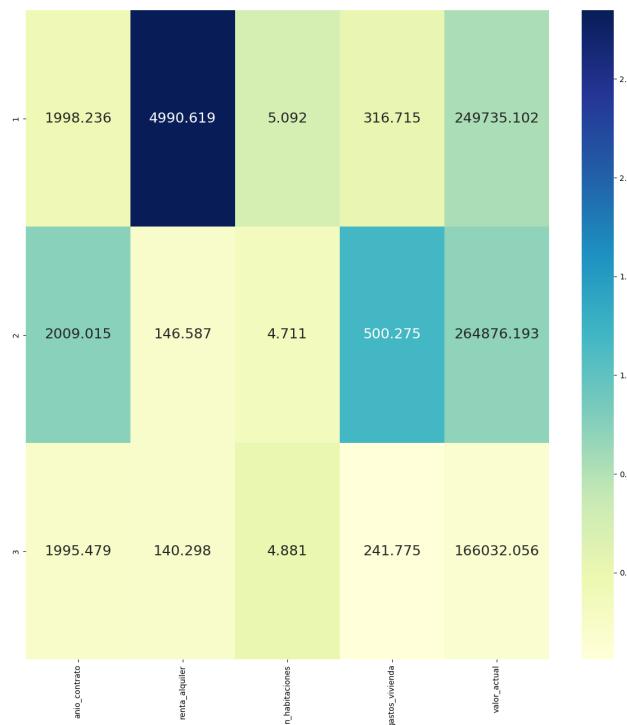


Figura 0.4: Heatmap presentado por K-Means

# Contenido adicional

Junto a esta memoria y el código utilizado se encuentran adjuntas las carpetas “caso1”, “caso2” y “caso3”, que contienen todo el material proporcionado como salida de las celdas del archivo P2.ipynb. En particular cada carpeta “casoX” contiene subcarpetas “img” y “tablas”:

- La carpeta `tablas` contiene en su raíz tablas comparativas de los parámetros de *K-Means* y *DBSCAN* en formato `.csv` y `.md` junto con la tabla comparativa con las medidas de todos los algoritmos. Contiene también 5 subcarpetas, una por algoritmo, con archivos en formato `.md` de las tablas de centroides generados por dicho algoritmo junto con una tabla con el tamaño de los clusters generados.
- La carpeta `img` contiene todos los gráficos que es posible crear mediante la celda que crea las gráficas. En concreto, en la raíz podemos encontrar los gráficos comparativos de los parámetros de *K-Means* y *DBSCAN* y 5 subcarpetas, una por cada algoritmo. Cada carpeta contiene, si es que es posible crearlas: matriz de dispersión, *heatmap* de centroides, gráfico de distancia intercluster, gráfico de distribución por variable y cluster y gráfico de coordenadas paralelas.

Se incluye también una estructura de carpetas vacía idéntica a la descrita pero sin estar encapsulada en ningún caso de estudio. Su propósito es permitir la ejecución de las celdas del archivo P2.ipynb sin obtener errores porque no existan las carpetas configuradas.



# Bibliografía

- colaboradores de Wikipedia. Wikipedia, la enciclopedia libre. Wikipedia. <https://es.wikipedia.org/wiki/Wikipedia:Portada>
- contextlib — Utilities for with-statement contexts — Python 3.10.0 documentation. contextlib. <https://docs.python.org/3/library/contextlib.html#LINE>
- Instituto Nacional de Estadística. (2020). INE. <https://www.ine.es/dyngs/INEbase/es/operacion>
- Linear Algebra (scipy.linalg) — SciPy v1.7.1 Manual. Scipy. <https://docs.scipy.org/doc/scipy/reference>
- pandas documentation — pandas 1.3.4 documentation. Pandas.Pydata.Org. <https://pandas.pydata.org/docs/>
- scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. Scikit-learn. <https://scikit-learn.org/>
- seaborn: statistical data visualization — seaborn 0.11.2 documentation. Seaborn. <https://seaborn.pydata.org/>
- Stack Overflow - Where Developers Learn, Share, & Build Careers. Stack Overflow. <https://stackoverflow.com/>

