UNIVERSIDAD
DE GRANADA

PRÁCTICA 2

Segmentación mediante Clustering

Inteligencia de Negocio

DGIIM - 2024/2025

Carmen Azorín Martí

G1

carmenazorin@correo.ugr.es

Índice

Introducción

Caso de estudio 1

En este caso de estudio, hemos analizado los apartamentos turísticos de Granada que cuentan con al menos una reseña en la plataforma Booking.com, con el objetivo de identificar patrones y segmentos de interés. Las características clave consideradas son el número de dormitorios, la capacidad de huéspedes y la distancia al centro de la ciudad, buscando explorar cómo estas variables influyen en la segmentación del mercado turístico. Por ejemplo, se espera que los apartamentos alejados del centro sean más adecuados para grupos grandes, mientras que los más céntricos se orienten principalmente a parejas o pequeños grupos.

El conjunto de datos analizado incluye 2182 registros representativos de la oferta turística de la ciudad. Para llevar a cabo el análisis, utilizamos cinco algoritmos de clustering (K-Means, DBSCAN, Agglomerative Clustering, Gaussian Mixture Models y Mean Shift), cada uno proporcionando una perspectiva única sobre los datos. La evaluación del desempeño de los algoritmos se basó en métricas como el coeficiente Silhouette, el índice Calinski-Harabasz y el tiempo de ejecución, permitiendo identificar el enfoque más adecuado para segmentar los apartamentos.

Los resultados muestran que K-Means se destacó como el algoritmo con mejor rendimiento general, al ofrecer una alta cohesión interna, buena separación entre clústeres y un tiempo de ejecución bajo, identificando seis grupos claramente definidos. Otros algoritmos, como Agglomerative Clustering y Mean Shift, también ofrecieron soluciones viables, especialmente para estructuras de datos más complejas o para identificar clústeres con diferentes densidades. Este análisis proporciona una base sólida para comprender cómo la ubicación y capacidad de los apartamentos influyen en las preferencias de los visitantes y en la oferta turística de Granada.

Caso de estudio 2

En este caso de estudio, nos enfocamos en los alojamientos turísticos del Albaicín, uno de los barrios más emblemáticos y visitados de Granada. Nuestro objetivo es analizar las relaciones entre la calidad de los alojamientos (medida en estrellas), su categoría (básica, destacada, preferente o plus) y su tipo (apartamento, hotel o sin especificar). Este análisis busca comprender si la calidad de los alojamientos depende significativamente de su categoría o tipo. Por ejemplo, queremos evaluar si los alojamientos de categoría "plus" ofrecen consistentemente una calidad superior o si los hoteles se perciben como más lujosos que los apartamentos.

El conjunto de datos incluye 444 registros correspondientes a alojamientos del Albaicín con información sobre estas características clave. Los datos fueron preprocesados, convirtiendo variables categóricas en valores numéricos para facilitar el análisis y visualizando las distribuciones de calidad, tipo y categoría mediante histogramas. De esta forma, identificamos patrones como la prevalencia de alojamientos con 4 y 5 estrellas y la predominancia de apartamentos frente a hoteles.

Para segmentar los alojamientos, se emplearon cinco algoritmos de clustering: K-Means, Agglomerative Clustering, Spectral Clustering, BIRCH y Gaussian Mixture Models (GMM). Estos algoritmos ofrecen perspectivas complementarias, permitiendo evaluar diferentes estructuras de los

datos. Los resultados muestran que K-Means y Spectral Clustering ofrecen la mejor calidad de agrupación, con altos coeficientes de Silhouette y buenos valores del índice Calinski-Harabasz. BIRCH también resulta eficaz, especialmente por su manejo eficiente de datos.

Este análisis proporciona una visión integral de cómo la calidad, el tipo y la categoría de los alojamientos del Albaicín pueden segmentarse para identificar patrones útiles en la gestión y promoción de la oferta turística en este barrio icónico.

Caso de estudio 3

El presente análisis se centra en la segmentación de apartamentos turísticos situados en dos de las áreas más emblemáticas de Granada: el Albaicín y el Centro. Estas zonas poseen características culturales, históricas y comerciales únicas que las convierten en polos turísticos de gran relevancia. El objetivo del estudio es identificar patrones y segmentos clave basados en variables como la calidad de los apartamentos (estrellas), la categoría de alojamiento (como "plus" o "preferente"), el área del apartamento (en m²) y el precio medio por noche, considerando todas las búsquedas en las que los apartamentos aparecen.

El análisis tiene como finalidad explorar cómo estas características influyen en la oferta turística y en la preferencia de los visitantes en cada zona. También se busca proporcionar información útil para los propietarios en la optimización de sus ofertas, y para los turistas en la toma de decisiones al momento de seleccionar un alojamiento.

El conjunto de datos contiene 374 apartamentos en el Albaicín y 904 en el Centro de Granada. Antes de aplicar los algoritmos de segmentación, se realizó un preprocesamiento exhaustivo que incluyó la conversión de variables categóricas a numéricas, la normalización de las variables seleccionadas, y la detección de casos atípicos (como apartamentos con precios excesivamente altos o áreas extremadamente grandes). Este análisis preliminar reveló diferencias significativas entre ambas zonas, tales como:

- Precios: Aunque en ambas zonas los precios oscilan entre 100 y 1300 euros, en el Albaicín existen apartamentos con precios que alcanzan hasta 4000 euros, lo que indica la presencia de alojamientos de lujo.
- Área: En ambas zonas existen apartamentos con áreas de 0 m², lo que probablemente se refiere a habitaciones individuales. Sin embargo, en el Albaicín también se encuentran apartamentos de hasta 700 m², considerados casos excepcionales.
- Calidad: En el Albaicín, los apartamentos están distribuidos exclusivamente entre 3 y 4 estrellas, mientras que en el Centro hay una mayor variabilidad, aunque la mayoría también se encuentra en las categorías de 3 y 4 estrellas.

Se aplicaron cinco algoritmos de clustering distintos para segmentar los apartamentos de ambas zonas, incluyendo K-Means, Agglomerative Clustering, DBSCAN, BIRCH y Gaussian Mixture Models (GMM). Estos algoritmos se evaluaron en función de métricas como el coeficiente Silhouette, el índice Calinski-Harabasz y el tiempo de ejecución. Cada uno aportó una perspectiva diferente sobre los patrones de agrupamiento en los datos.

En las siguientes secciones, se detallan los resultados obtenidos para cada zona y algoritmo, destacando patrones relevantes, diferencias entre el Albaicín y el Centro, y recomendaciones basadas

en los hallazgos. Este análisis busca contribuir al entendimiento de la oferta turística en Granada, ayudando tanto a los gestores turísticos como a los visitantes a tomar decisiones informadas.

Caso de estudio 1

Análisis de apartamentos en función de la ubicación y capacidad de alojamiento

En este caso de estudio, nos centramos en analizar los apartamentos turísticos de Granada que cuentan con al menos una reseña en la plataforma Booking.com. El objetivo es identificar patrones y grupos de interés basándonos en características como el número de dormitorios, el número de huéspedes y la distancia al centro. Este análisis busca explorar cómo estos factores influyen en la segmentación del mercado. Por ejemplo, esperamos encontrar que los apartamentos alejados del centro sean más adecuados para grupos grandes, mientras que los más céntricos puedan estar orientados a parejas.

El conjunto de datos seleccionado para este análisis contiene **2182 registros**, todos ellos correspondientes a apartamentos con al menos una reseña. Este subconjunto representa una muestra significativa de la oferta turística en la ciudad.

Para realizar la segmentación, utilizamos cinco algoritmos diferentes de clustering. Cada uno de ellos aporta una perspectiva única sobre los datos y nos permite comparar resultados para elegir el enfoque más adecuado:

- **K-Means**: Se usa por su simplicidad y rapidez. Este algoritmo requiere predefinir el número de clústeres y asume que tienen formas esféricas, lo que lo hace limitado para distribuciones más complejas.
- **DBSCAN**: Es excelente para identificar clústeres de diferentes densidades y manejar puntos atípicos. Sin embargo, su desempeño depende de parámetros como eps y el tamaño mínimo de clúster.
- **Clustering jerárquico (Agglomerative Clustering)**: Este método construye una jerarquía de clústeres que puede visualizarse mediante dendrogramas, lo cual facilita interpretar relaciones estructurales entre grupos. Su principal desventaja es el tiempo de ejecución para grandes conjuntos de datos.
- **Gaussian Mixture Models (GMM)**: Este enfoque probabilístico nos ayuda a identificar clústeres no necesariamente esféricos. Es especialmente útil para datos con superposiciones, pero puede ser más costoso computacionalmente.
- **Mean Shift**: Es capaz de determinar automáticamente el número de clústeres al buscar densidades máximas en los datos. Aunque no requiere predefinir el número de grupos, puede ser lento con conjuntos de datos grandes.

A continuación, se presenta una tabla comparativa con los resultados obtenidos al aplicar estos algoritmos al subconjunto seleccionado:

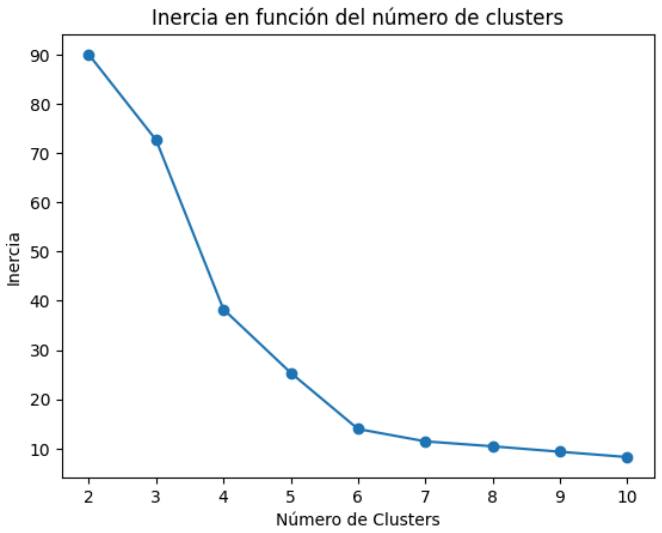
Algoritmo	Número clústeres	Coeficiente Silhouette	Índice Calinski-Harabasz	Tiempo de ejecución (s)
k-Means	6	0.739	8932.388	0.025604963302612305
DBSCAN	23	0.749	1049.945	0.16195273399353027
Agglomerative Clustering	2/6	0.641/ 0.739	734.993/ 734.993	0.7329256534576416/ 0.13674426078796387
GMM	5	0.708	4116.561	0.03392672538757324
Mean Shift	9	0.727	3402.392	5.452937126159668

Podemos observar en la tabla que el algoritmo con mejor rendimiento general es el K-Means, gracias a su alta cohesión (indicada por Silhoutte), mejor separación (indicada por Calinski-Harabasz) y bajo tiempo de ejecución. Una alternativa también válida sería Agglomerative Clustering para 6 clústeres, con resultados similares pero mayor coste computacional. Esto implica que se identifican seis grupos claramente definidos.

K-Means

El algoritmo de K-Means lo he ejecutado para 6 clusters, que es el número óptimo de clústeres.
Para calcular este número he usado el método del codo que consiste en calcular la suma de errores cuadrados dentro del grupo (WSS) para diferentes valores de k y elegir el k para el cual WSS comienza a disminuir primero. En el gráfico de WSS versus k, esto se ve como un codo. Para implementarlo en Python he hecho uso de la biblioteca *sklearn* y he obtenido la siguiente gráfica:

```
inercia = []
for k in range(2, 11):  # Probar con 2 a 10 clusters
    kmeans = KMeans(n_clusters=k, random_state=123456)
    kmeans.fit(X_normal)
    inercia.append(kmeans.inertia_)

# Graficar la inercia para diferentes valores de k
plt.plot(range(2, 11), inercia, marker='o')
plt.title("Inercia en función del número de clusters")
plt.xlabel("Número de Clusters")
plt.ylabel("Inercia")
plt.show()
```



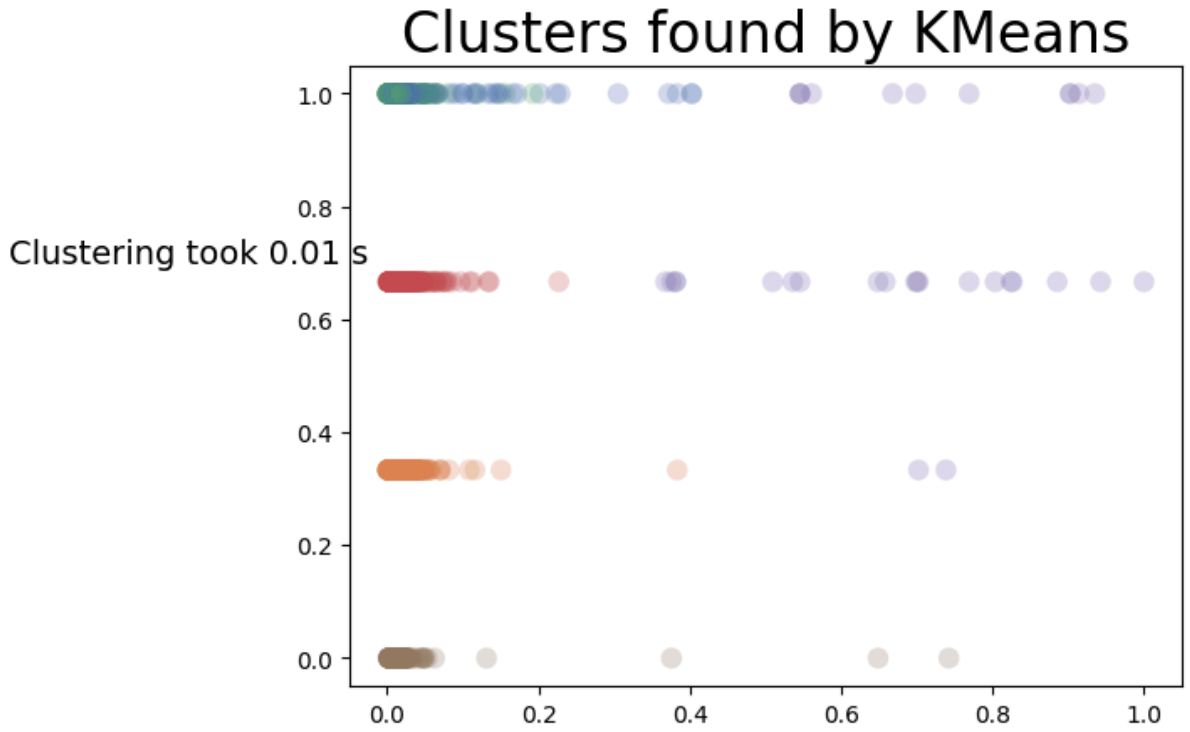
En la gráfica aparece un brazo con un codo en k=6. Al ejecutar el algoritmo con 6 clústeres y las tres variables que queremos estudiar obtenemos las siguientes métricas:

```
from sklearn.cluster import KMeans
from sklearn import metrics
from math import floor

# Ejecutar K-Means
k_means = KMeans(init='k-means++', n_clusters=6, n_init=5,
random_state=123456)
```

```
inicio = time.time()
kmeans_labels = k_means.fit_predict(X_normal)
fin = time.time()

# Calcular métricas
metric_CH = metrics.calinski_harabasz_score(X_normal, kmeans_labels)
metric_SC = metrics.silhouette_score(X_normal, kmeans_labels,
metric='euclidean')

print(f"k-means Calinski-Harabasz Index: {metric_CH:.3f}, Silhouette:
{metric_SC:.3f}, tiempo ejecución: {fin-inicio}")
```

Calinski-Harabasz Index: 8932.388, Silhouette: 0.739, tiempo ejecución: 0.025604963302612305

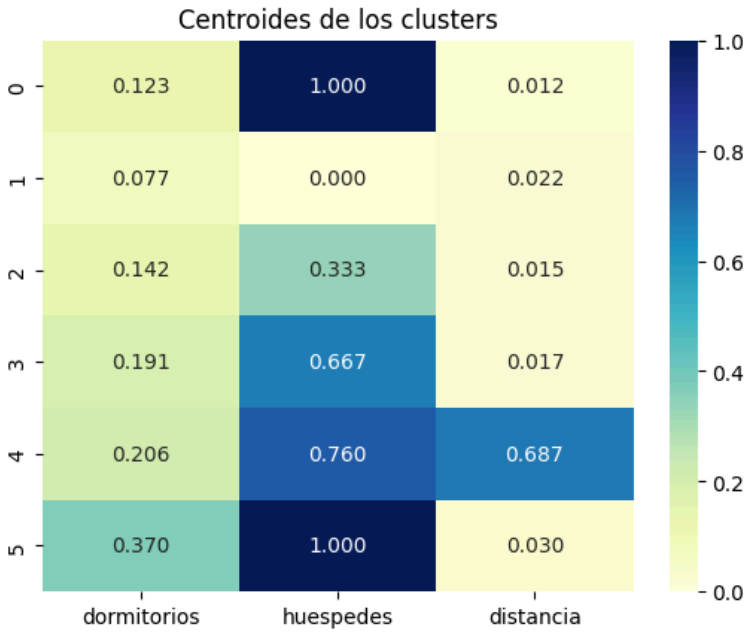
Además, para visualizar el resultado he usado un scatter plot que representa los clusters formados por este algoritmos considerando las variables distancia al centro (eje x) y número de huéspedes (eje y).



Vemos que el algoritmo ha logrado identificar 6 clusters bien diferenciados. Esto indica que los huéspedes tienden a agruparse en función de su distancia al centro y el tamaño de sus reservas. El cluster morado, por ejemplo, refleja grupos de personas que alquilan apartamentos entre 6 y 8 personas en zonas muy alejadas al centro.

Además, para diferenciar mejor los diferentes clusters desarrollados por el algoritmo, he visualizado un heatmap de centroides, que muestra los valores normalizados de los centroides de los clusters en función de las variables dormitorios, huéspedes y distancia al centro. Cada fila representa un cluster y cada columna indica el promedio de la variable correspondiente para ese cluster.

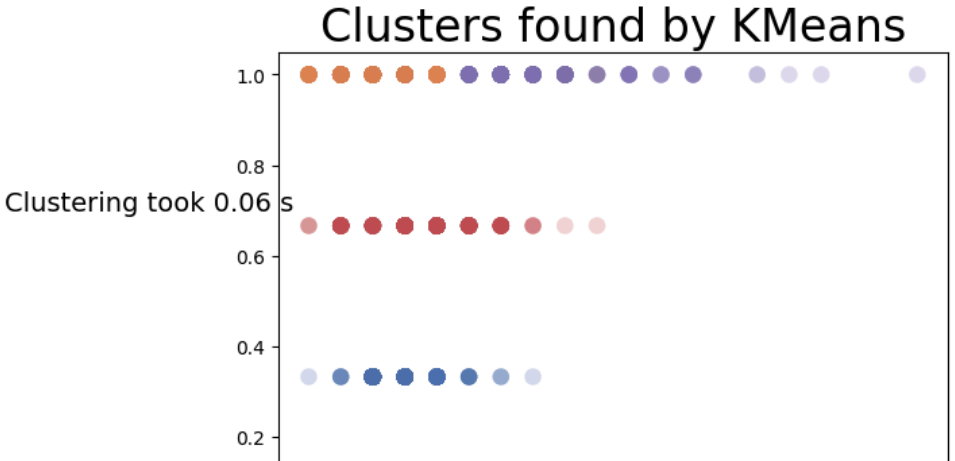
```
import seaborn as sns
import matplotlib.pyplot as plt

# Heatmap de centroides
centers = pd.DataFrame(k_means.cluster_centers_,
columns=X_normal.columns)
sns.heatmap(centers, cmap="YlGnBu", annot=True, fmt='.3f')
plt.title("Centroides de los clusters")
plt.show()
```



El cluster 0 representa las reservas grupales grandes (8 personas) en ubicaciones céntricas pero con pocas habitaciones. Por otro lado, el cluster 1 representa las reservas de 2 personas (el mínimo de personas) con una sola habitación, lo que indica que probablemente se trata de una pareja en zonas también muy céntricas. El cluster 2 representa parejas o pequeños grupos en zonas céntricas. El cluster 3 es representativo de grupos medianos (4-6 personas) en ubicaciones céntricas. El cluster 4 representa grupos grandes en zonas periféricas. Finalmente, el cluster 5 representa reservas grandes (8 personas) en muchas habitaciones cerca del centro.

Como vemos, hay una clara diferenciación en los clusters en función de la distancia al centro y el tamaño de la reserva. Los clusters 0, 3 y 5 destacan por su preferencia por ubicaciones céntricas, mientras que el cluster 4 se asocia con zonas más alejadas

Tenemos otro scatter plot que representa en el eje x los dormitorios y en el eje y los huéspedes, en donde se ve una clara diferencia de clusters.



Observamos que hay dos clusters representando a grupos de 8 personas, aquellos que se alojan en apartamentos más alejados del centro y aquellos que lo hacen en zonas céntricas. De esta forma, podemos concluir con una tabla de diferencia entre los 6 clusters obtenidos por el algoritmo K-Means:

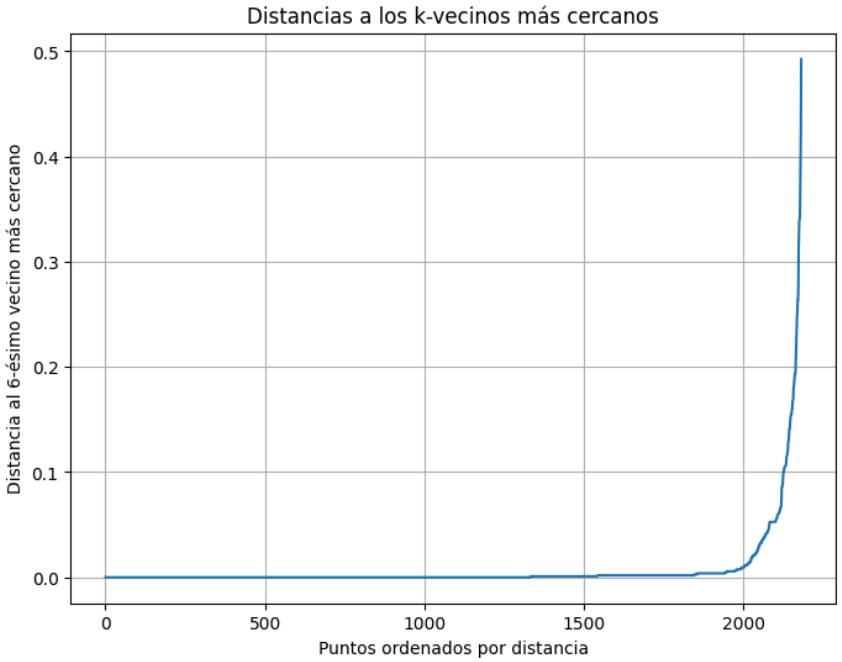
Número de cluster	Número de huéspedes	Número de dormitorios	Distancia al centro
0	8	Pocos	Poca
1	2	Pocos	Poca
2	2-4	Pocos	Poca
3	4-6	Pocos	Poca
4	6-8	Pocos	Mucha
5	8	Muchos	Poca

DBSCAN

El algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) requiere de la especificación de dos parámetros: épsilon (la distancia máxima que pueden tener dos puntos entre sí sin dejar de pertenecer al mismo grupo) y muestras mínimas (la menor cantidad de ùntos necesarios para formar un grupo). Una regla general es establecer min_samples (muestras mínimas) para que sea dos veces la dimensión del conjunto de datos, pero puede ser necesario un valor más grande. Por otro lado, épsilon se puede elegir usando el método de distancia de los k vecinos más cercanos (kNN). En este método, calculamos la distancia promedio de cada punto de datos al k vecino más cercano.

Teniendo en cuenta lo comentado en el párrafo anterior, he elegido un min_samples=6 y un eps=0.05 usando el método comentado y observando un codo en dicho punto. Esto indica un umbral natural para distinguir entre puntos de cluster y puntos de ruido en el conjunto de datos.

```
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import numpy as np

# Número de vecinos más cercanos (puedes probar con diferentes valores, como 4 o 5)
k = 6

# Crear el modelo de vecinos más cercanos
neighbors = NearestNeighbors(n_neighbors=k)
neighbors_fit = neighbors.fit(X_normal)

# Calcular las distancias a los k-vecinos más cercanos
distances, indices = neighbors_fit.kneighbors(X_normal)

# Obtener la distancia al k-vecino más cercano (usamos la última columna que es la distancia al k-ésimo vecino)
distances = np.sort(distances[:, k-1], axis=0)

# Graficar las distancias
plt.figure(figsize=(8, 6))
plt.plot(distances)
plt.title('Distancias a los k-vecinos más cercanos')
plt.xlabel('Puntos ordenados por distancia')
plt.ylabel(f'Distancia al {k}-ésimo vecino más cercano')
plt.grid(True)
plt.show()
```



Como no se aprecia el punto exacto en el que está el codo, ejecutamos un script que nos indica los distintos valores de Silhoutte para los distintos valores de eps:

```
eps_values = [0.01, 0.02, 0.03, 0.04,0.05,0.06,0.07]
for eps in eps_values:
    dbscan = DBSCAN(eps=eps, min_samples=10)
    dbscan_labels = dbscan.fit_predict(X_normal)
    silhouette = metrics.silhouette_score(X_normal, dbscan_labels)
    print(f"DBSCAN - eps={eps} Silhouette: {silhouette}")
```

```
DBSCAN - eps=0.01 Silhouette: 0.7113127831945119
DBSCAN - eps=0.02 Silhouette: 0.7242165366517701
DBSCAN - eps=0.03 Silhouette: 0.7271559671244867
DBSCAN - eps=0.04 Silhouette: 0.7281686648447941
DBSCAN - eps=0.05 Silhouette: 0.7330467761153561
DBSCAN - eps=0.06 Silhouette: 0.7206748380650837
DBSCAN - eps=0.07 Silhouette: 0.7221884148527346
```

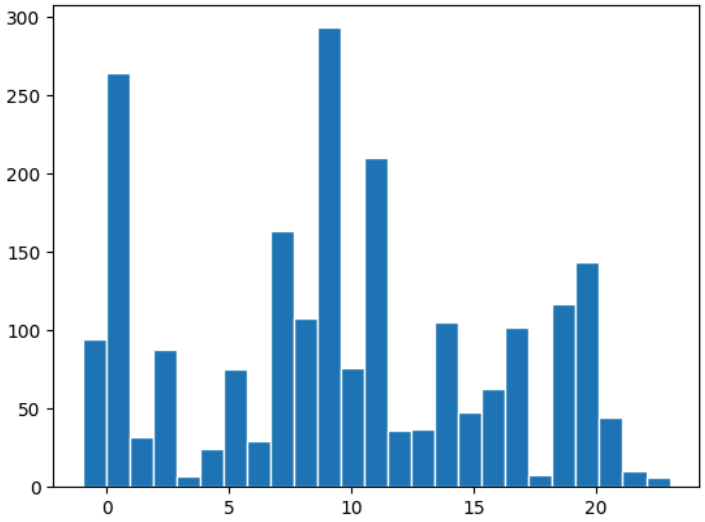
Obtenemos el coeficiente más alto cuando eps=0.05, así que éste será el parámetro que utilicemos. Al ejecutar el algoritmo con las tres variables que queremos estudiar obtenemos las siguientes métricas:

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

# Ejecutar DBSCAN
dbscan = DBSCAN(eps=0.05, min_samples=6)
inicio = time.time()
dbscan_labels = dbscan.fit_predict(X_normal)
```

```
fin = time.time()

# Evaluar la calidad (usualmente no usamos métricas de inercia con DBSCAN)
calinski_harabasz_dbscan = metrics.calinski_harabasz_score(X_normal, dbscan_labels)
silhouette_dbscan = metrics.silhouette_score(X_normal, dbscan_labels)
print(f"dbscan Calinski-Harabasz Index: {calinski_harabasz_dbscan:.3f}, k-means Silhouette: {silhouette_dbscan:.3f}, tiempo ejecución: {fin-inicio}")
```

Calinski-Harabasz Index: 1049.945, Silhouette: 0.749, tiempo ejecución: 0.16195273399353027

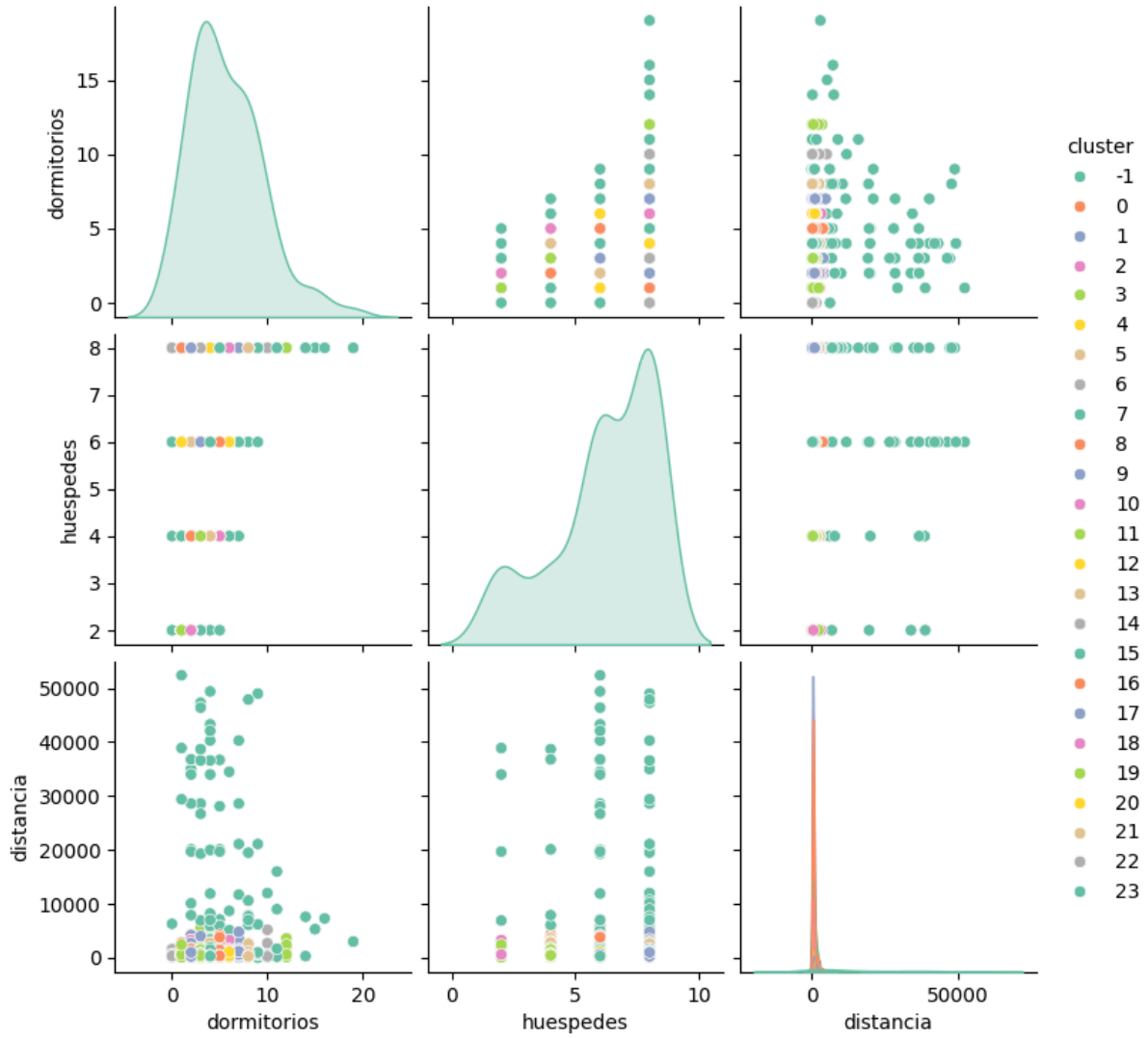
Veamos la cantidad de clusters y el número de elementos en cada uno mediante un histograma:



Vemos que hay 25 clusters, algunos con muy pocos elementos y otros con bastante elementos. El cluster -1 suele asociarse al de los outliers, lo que supone que hay algo más de 150 elementos que se consideran ruido y no pertenece a ninguno de los otros clusters.

Además, para visualizar los resultados del algoritmo he usado un scatter matrix considerando las tres variables.

```
# Scatter matrix
clusters = pd.DataFrame(dbscan_labels, index=X_normal.index, columns=['cluster'])
datos_cluster['cluster'] = clusters['cluster']
sns.pairplot(datos_cluster,
vars=['dormitorios','huespedes','distancia'], hue="cluster", palette="Set2")
plt.show()
```



Como podemos observar, los puntos más visibles son los outliers, puesto que no se consideran de ningún otro cluster. La mayoría de los clusters contienen huéspedes que se hospedan en apartamentos muy céntricos y, por tanto, los clusters se diferencian por la relación entre el número de huéspedes y la cantidad de dormitorios que alquilan. Sin embargo, no podemos encontrar una diferencia muy clara entre los clusters, a diferencia de cómo habíamos conseguido con K-Means.

Agglomerative Clustering

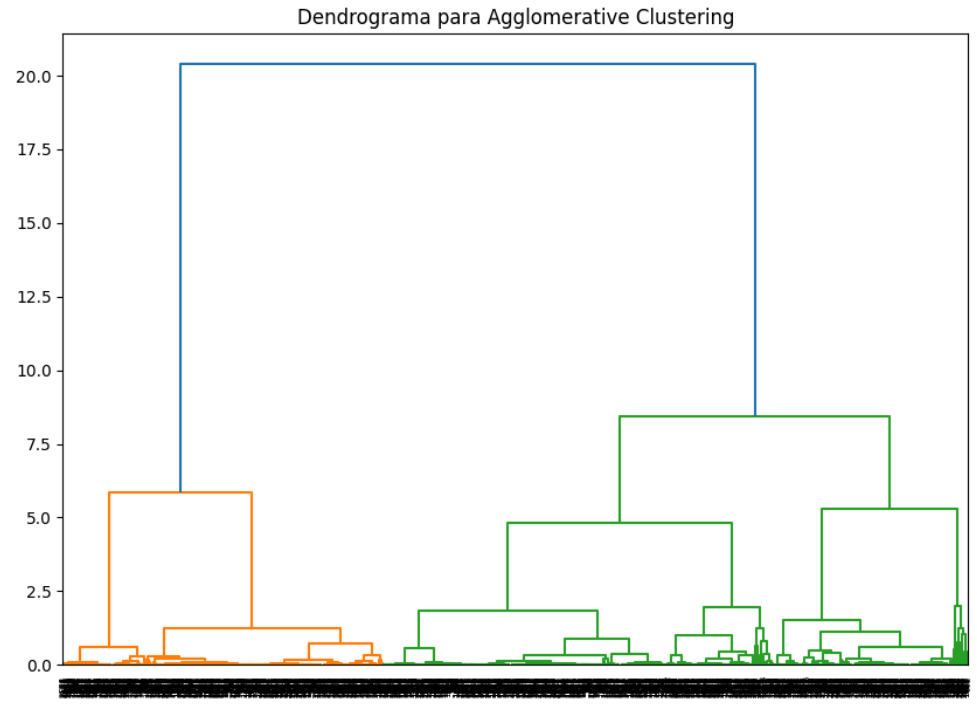
El clustering aglomerativo es una técnica jerárquica, es decir, supone que los puntos de datos que están más cerca entre sí son más similares o están más relacionados que los puntos de datos que están más separados.

Para visualizar un clustering jerárquico se usa el dendograma, una figura con forma de árbol que representa las relaciones jerárquicas entre grupos. Los puntos de datos individuales se ubican en la parte inferior del dendograma, mientras que los grupos más grandes, que incluyen todos los puntos de datos, se ubican en la parte superior. Para generar distintas cantidades de clusters, el dendograma se puede dividir a distintas alturas. El número óptimo de clusters se consigue:

1. Encontrar la diferencia vertical más grande entre las líneas horizontales del dendograma
2. En el punto medio de la diferencia trazas una línea horizontal
3. El número óptimo de clusters es la cantidad de intersecciones que tiene dicha línea

Sabiendo esto, representemos el dendograma y encontremos el número óptimo de clusters.

```
from scipy.cluster.hierarchy import dendrogram, linkage

# Generar el linkage para Agglomerative Clustering
Z = linkage(X_normal, 'ward')

# Visualizar el dendrograma
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title("Dendrograma para Agglomerative Clustering")
plt.show()
```



Vemos que la diferencia vertical más grande se encuentra entre la recta en y=20.0 y la recta en y=8.5 y, por tanto, el número óptimo de clusters es 2. La siguiente diferencia más grande se encuentra entre la recta en y=4.5 (verde) y la siguiente recta verde en y=2.0, lo que supone 6 clusters. Interpretemos los resultados para ambos valores:

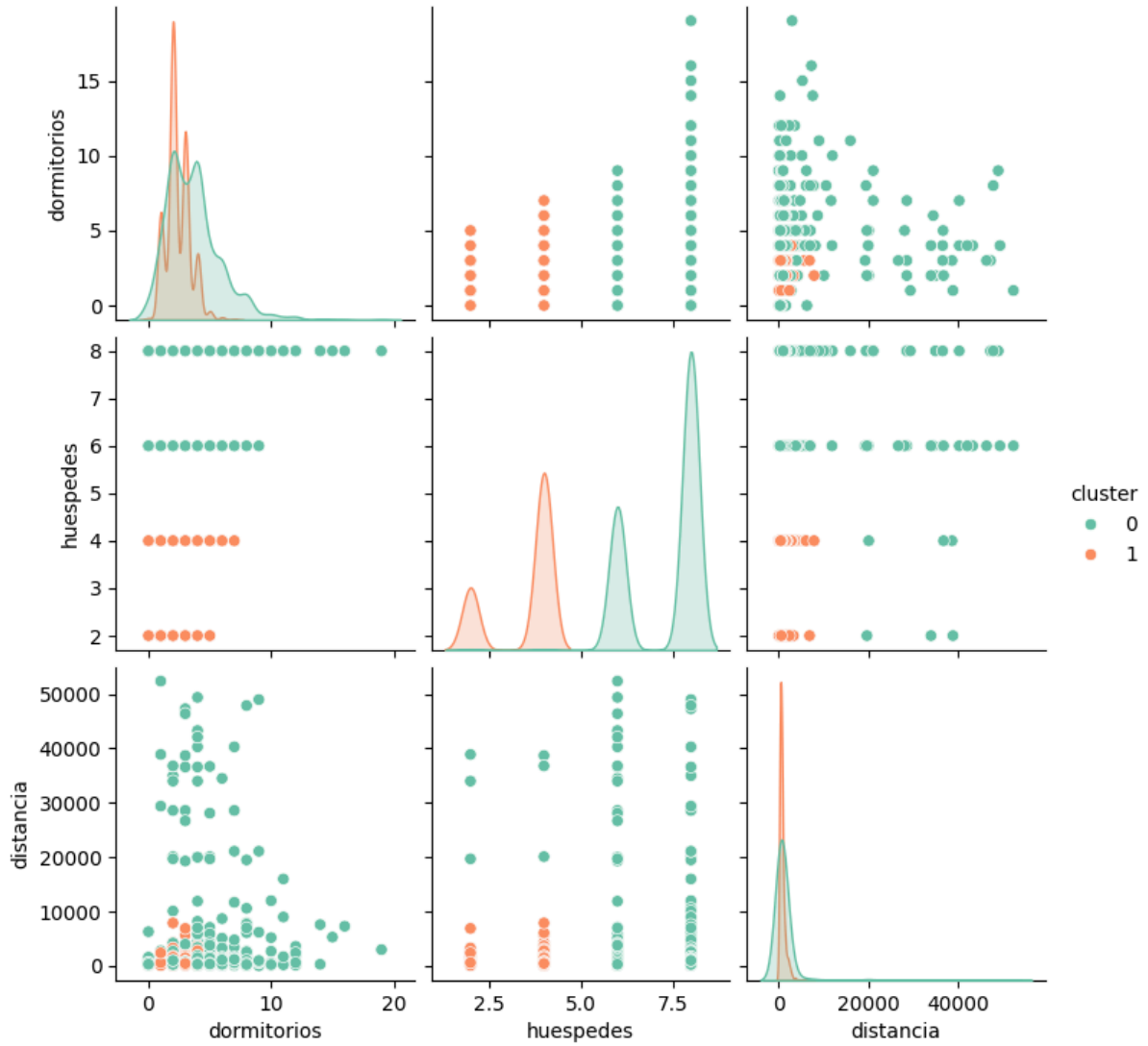
Número de clusters = 2

Al ejecutar el algoritmo para 2 clusters obtenemos los siguientes valores en las medidas de rendimiento:

```
from sklearn.cluster import AgglomerativeClustering

# Ejecutar Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=2)
inicio = time.time()
agglo_labels = agglo.fit_predict(X_normal)
fin = time.time()

# Evaluar la calidad
calinski_harabasz_agglo = metrics.calinski_harabasz_score(X_normal, dbscan_labels)
silhouette_agglo = metrics.silhouette_score(X_normal, agglo_labels)
print(f"Agglomerative Clustering Calinski-Harabasz Index: {calinski_harabasz_agglo:.3f}, k-means Silhouette: {silhouette_agglo:.3f}, tiempo ejecución: {fin-inicio}")
```

Calinski-Harabasz Index: 734.993, Silhouette: 0.641, tiempo ejecución: 0.26859354972839355

Vemos que el tiempo de ejecución es considerablemente más alto que en el resto de algoritmos y, además, el coeficiente de Silhouette es más bajo. Veamos la representación de los clusters con un scatter matrix.



Efectivamente, obtenemos dos clusters. La principal diferencia entre ellos es el número de huéspedes: el cluster 0 contiene apartamentos que alojan entre 2/4 personas y el cluster 1 contiene apartamentos que alojan entre 6/8 personas. Esto coincide con la variable del número de habitaciones, puesto que el cluster 0 contiene apartamentos con pocas habitaciones y el cluster 1 contiene apartamentos con muchas habitaciones. En cuanto a la distancia al centro, observamos que algunos puntos que creíamos que pertenecían al cluster 0 (por tener pocos dormitorios), resulta que pertenecen al cluster 1, puesto que viven en zonas alejadas del centro. Con esto, ya podemos rellenar la tabla:

Cluster	Número de huéspedes	Número de dormitorios	Distancia al centro
0	Entre 2 y 8	0 ó más	Indiferente
1	2/4	0/1	Poca

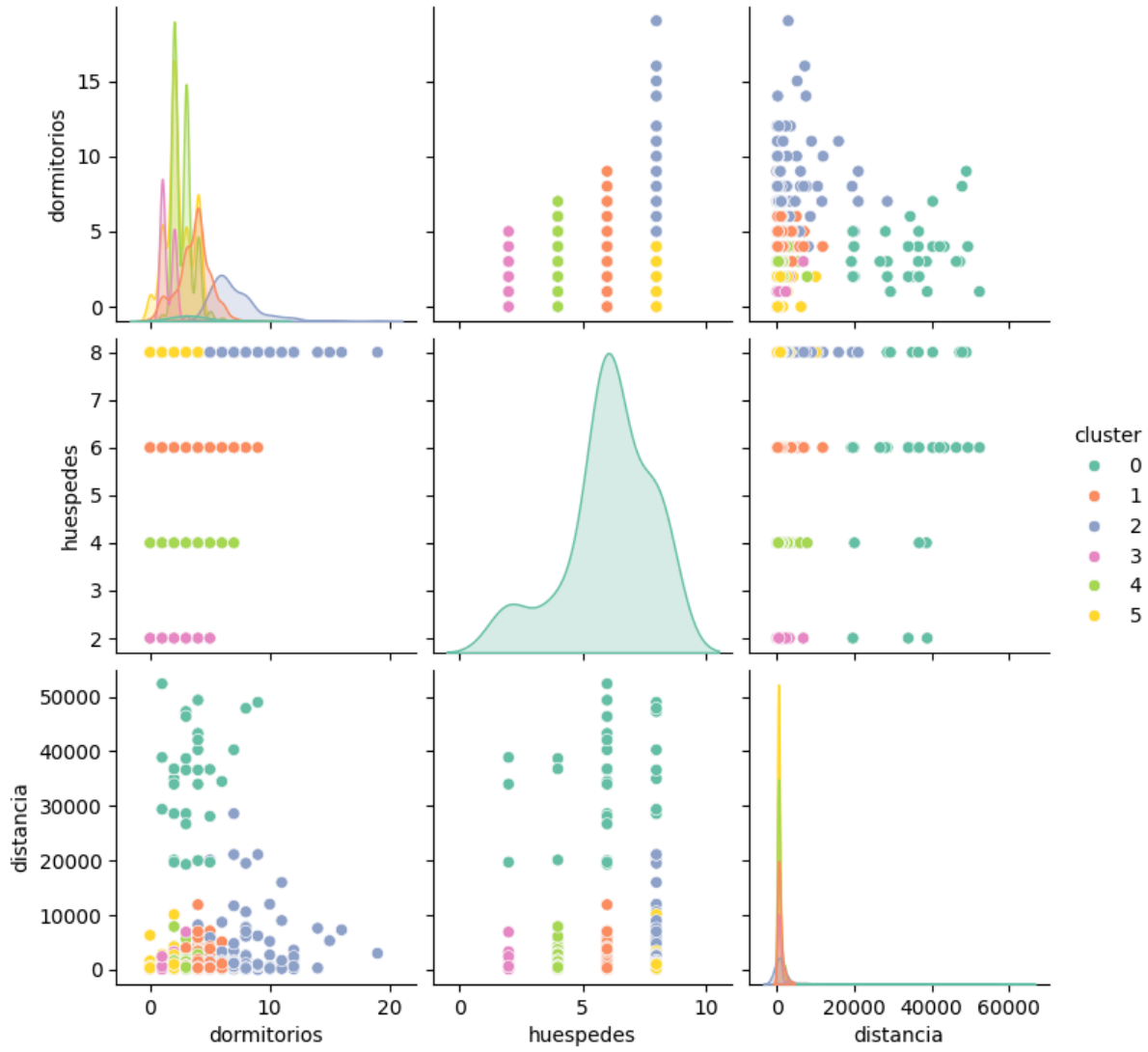
Número de clusters = 6

Ahora ejecutamos el algoritmo de clustering aglomerativo para 6 clústeres. Los valores en las medidas de rendimiento son:

```
from sklearn.cluster import AgglomerativeClustering

# Ejecutar Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=6)
inicio = time.time()
agglo_labels = agglo.fit_predict(X_normal)
fin = time.time()

# Evaluar la calidad
calinski_harabasz_agglo = metrics.calinski_harabasz_score(X_normal,
dbscan_labels)
silhouette_agglo = metrics.silhouette_score(X_normal, agglo_labels)
print(f"Agglomerative Clustering Calinski-Harabasz Index:
{calinski_harabasz_agglo:.3f}, k-means Silhouette:
{silhouette_agglo:.3f}, tiempo ejecución: {fin-inicio}")
```

Calinski-Harabasz Index: 734.993, Silhouette: 0.739, tiempo ejecución: 0.13674426078796387

Observamos un incremento en el coeficiente de Silhouette y un descenso en el tiempo de ejecución. Veamos la scatter matrix para estos clusters.



Efectivamente obtenemos 6 clusters diferentes dependiendo del número de dormitorios, el número de huéspedes y la distancia al centro. Nos encontramos con que el cluster 0 depende mayoritariamente de la distancia al centro, concretamente de los apartamentos más alejados. El resto de clusters contienen apartamentos céntricos y se diferencian, en gran medida, por el número de huéspedes. El cluster 1 contiene apartamentos de 6 huéspedes; el cluster 3, apartamentos de 2 huéspedes; y el cluster 4, apartamentos de 4 huéspedes. Tenemos entonces que los apartamentos de 8 huéspedes se dividen entre los clusters 2 y 5, dependiendo del número de dormitorios: pocos dormitorios implica cluster 5 y muchos dormitorios, cluster 2. Cabe destacar que los apartamentos del cluster 2 están a una distancia media-baja del centro. Obtenemos la siguiente tabla:

Cluster	Número de huéspedes	Número de dormitorios	Distancia al centro
0	Indefinido	Indefinido	Mucha
1	6	Entre 1 y 3	Poca
2	8	2 ó más	Poca - Media
3	2	0/1	Poca
4	4	1/2	Poca

5	8	0/1	Poca

Obtenemos unos clusters muy bien diferenciados y similares a los encontrados por K-Means: un único cluster para los alejados del centro, dos clusters para apartamentos de 8 huéspedes y el resto de apartamentos repartidos según el número de huéspedes.

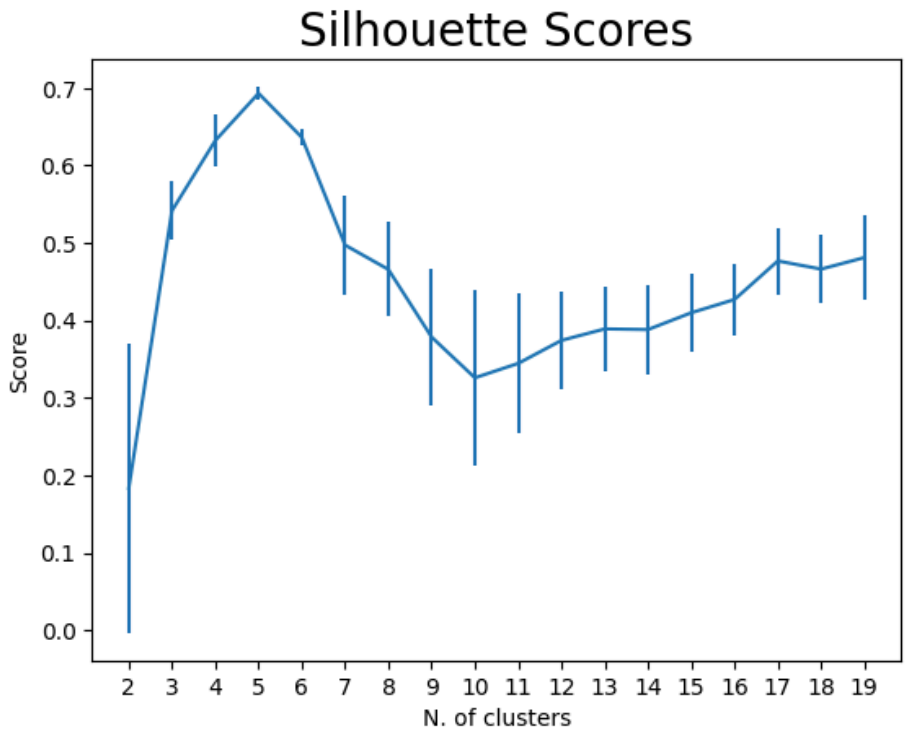
GMM

Al algoritmo de clustering GMM se puede ver como un K-Means que es capaz de formar grupos estirados. Para encontrar el parámetro de número de clusters óptimo usaremos la puntuación de Silhouette que considera dos medidas:

- La distancia media entre una muestra y todos los demás puntos del mismo grupo.
- La distancia media entre una muestra y todos los demás puntos del siguiente cluster más cercano.

Es decir, comprueba en qué medida los grupos están compactos y bien separados. El procedimiento de ajuste no es determinista, realizamos 20 ajustes para cada número de clusters y luego consideramos el valor medio y la desviación estándar de las cinco mejores ejecuciones. Los resultados son los siguientes:

```
# encontrar numero optimo de clusters por puntuacionde silueta
def SelBest(arr:list, X:int)->list:
    '''
    returns the set of X configurations with shorter distance
    '''
    dx=np.argsort(arr)[:X]
    return arr[dx]

n_clusters=np.arange(2, 20)
sils=[]
sils_err=[]
iterations=20
for n in n_clusters:
    tmp_sil=[]
    for _ in range(iterations):
        gmm=GaussianMixture(n, n_init=2).fit(X_normal)
        labels=gmm.predict(X_normal)
                        sil=metrics.silhouette_score(X_normal,   labels,
metric='euclidean')
        tmp_sil.append(sil)
    val=np.mean(SelBest(np.array(tmp_sil), int(iterations/5)))
    err=np.std(tmp_sil)
    sils.append(val)
    sils_err.append(err)
```

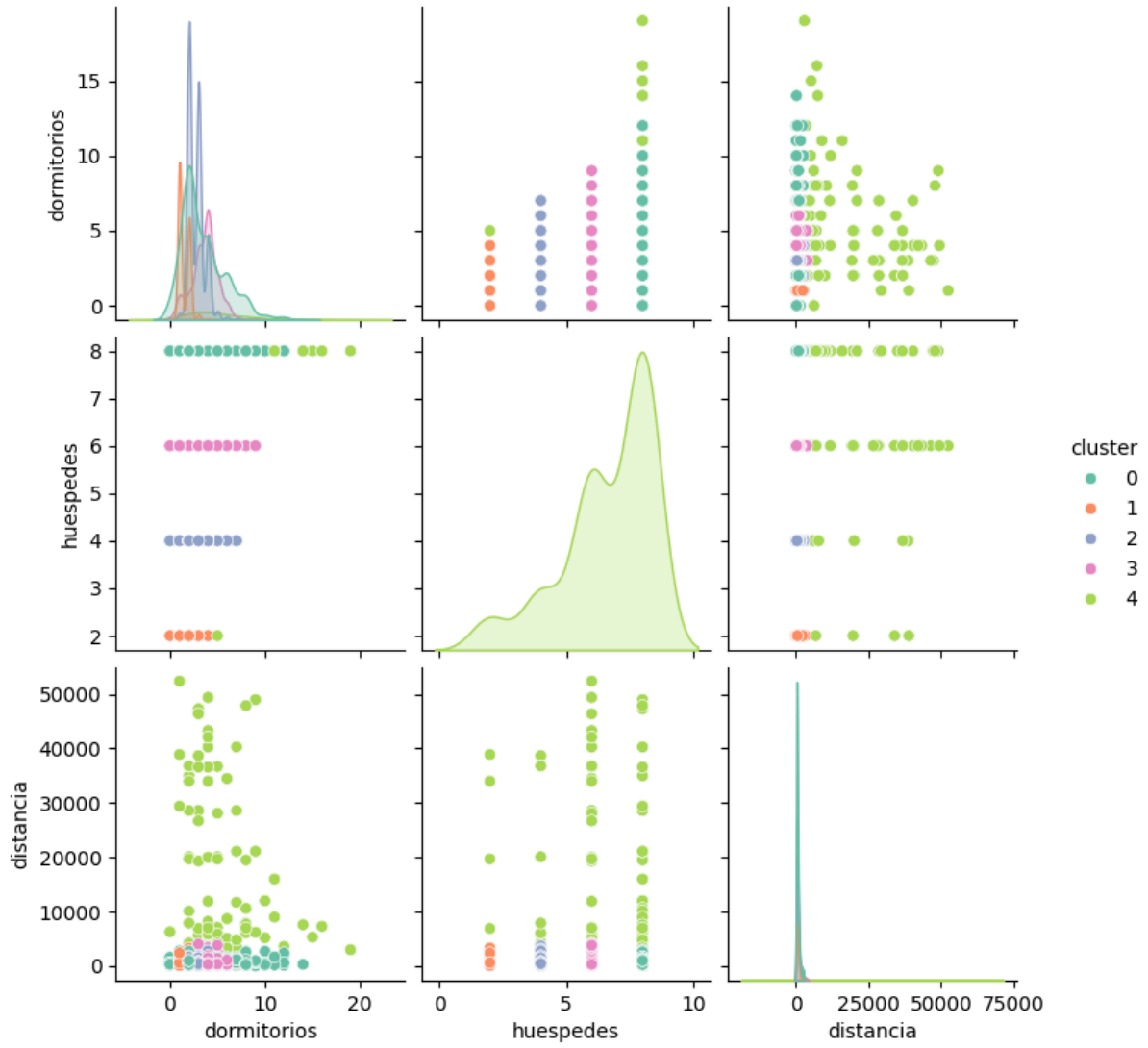


Resulta que obtenemos la mejor puntuación con 5 clusters. Aunque la configuración de 4 y 6 grupos es también buena. Con el número de clusters obtenido, podemos ejecutar el algoritmo de Gaussian Mixture Models (GMM) y obtener los siguientes valores para las medidas de rendimiento:

```
from sklearn.mixture import GaussianMixture

# Ejecutar GMM
gmm = GaussianMixture(n_components=5, random_state=123456)
inicio = time.time()
gmm_labels = gmm.fit_predict(X_normal)
fin = time.time()

# Evaluar la calidad
calinski_harabasz_gmm = metrics.calinski_harabasz_score(X_normal,
gmm_labels)
silhouette_gmm = metrics.silhouette_score(X_normal, gmm_labels)
print(f"GMM Calinski-Harabasz Index: {calinski_harabasz_gmm:.3f},
k-means Silhouette: {silhouette_gmm:.3f}, tiempo ejecución:
{fin-inicio}")
```

Calinski-Harabasz Index: 4116.561, Silhouette: 0.708, tiempo ejecución: 0.03392672538757324

Los valores son muy buenos, aunque siguen sin superar a los de K-Means. Para visualizar los apartamentos de cada cluster he mostrado un scatter matrix.



Vemos los 5 clusters distribuidos de forma similar a la de los anteriores algoritmos, solo que elimina un cluster. Tenemos por un lado el cluster 4, que contiene a los apartamentos que no son céntricos y que, en el caso de tener 8 huéspedes, tienen más habitaciones. El resto de clusters se diferencian por el número de huéspedes: el cluster 0 tiene 8 huéspedes, el cluster 1 tiene 2 huéspedes, el cluster 2 tiene 4 huéspedes y el cluster 3 tiene 6 huéspedes. Podemos crear la siguiente tabla:

Cluster	Número de huéspedes	Número de dormitorios	Distancia al centro
0	8	Entre 0 y 4	Poca
1	2	0/1	Poca
2	4	Entre 0 y 2	Poca
3	6	Entre 0 y 3	Poca
4	Indefinido	Indefinido	Mucha

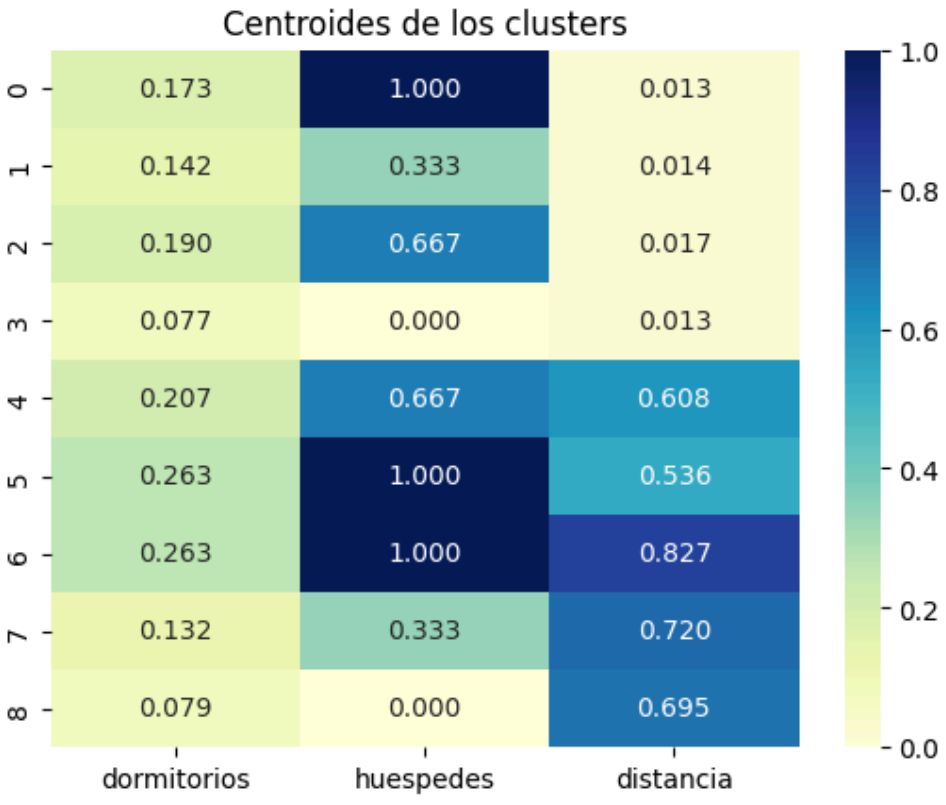
Meanshift

Meanshift es un método de clustering basado en la densidad que se centra en encontrar regiones de alta densidad y desplazar iterativamente los puntos de datos hacia mayor densidad de puntos. No requiere un parámetro que indique la cantidad de clusters. Al ejecutar este algoritmo obtenemos los siguientes valores en las medidas de rendimiento:

```
from sklearn.cluster import MeanShift

# Ejecutar MeanShift
meanshift = MeanShift()
inicio = time.time()
meanshift_labels = meanshift.fit_predict(X_normal)
fin = time.time()

# Evaluar la calidad
calinski_harabasz_meanshift = metrics.calinski_harabasz_score(X_normal,
meanshift_labels)
silhouette_meanshift = metrics.silhouette_score(X_normal,
meanshift_labels)
print(f"MeanShift Calinski-Harabasz Index:
{calinski_harabasz_meanshift:.3f}, k-means Silhouette:
{silhouette_meanshift:.3f}, tiempo ejecución: {fin-inicio}")
```

Calinski-Harabasz Index: 3402.392, Silhouette: 0.727, tiempo ejecución: 5.452937126159668

Obtenemos unos valores altos en los coeficientes, pero también en el tiempo de ejecución. Veamos un heatmap que represente cada cluster.



Observamos dos tipos de clusters muy diferenciados: los cuatro primeros son céntricos y los cinco siguientes son lejanos. Los clusters céntricos se diferencian entre ellos por el número de huéspedes. Entre los clústeres lejos del centro hay uno para cada número de huéspedes excepto para 8 huéspedes, que se diferencian entre distancia media y alta del centro. Como conclusión, rellenaremos la tabla:

Cluster	Número de huéspedes	Número de dormitorios	Distancia al centro
0	8	Pocos	Poca
1	4	Pocos	Poca
2	6	Pocos	Poca
3	2	Muy pocos	Poca
4	6	Medio	Media
5	8	Bastantes	Media
6	8	Bastantes	Mucha
7	4	Pocos	Mucha
8	2	Muy pocos	Media - Mucha

Interpretación de la segmentación

El estudio refleja que la distancia al centro, el número de huéspedes y los dormitorios disponibles juegan un papel clave en cómo se forman los clústeres de las propiedades. Según los algoritmos y sus resultados, podemos inferir lo siguiente:

1. Propiedades céntricas (distancia baja):
    - Grupos pequeños (2-4 huéspedes) dominan la demanda en estas áreas.
    - Los alojamientos con 1-2 dormitorios son los más solicitados.
    - Estrategia: Promocionar propiedades compactas en el centro para parejas o pequeños grupos. Resaltar servicios clave como comodidad y accesibilidad.
2. Propiedades en zonas intermedias:
    - Se identifican preferencias de grupos medianos (4-6 huéspedes) con 2-3 dormitorios.
    - Estrategia: Posicionar propiedades de tamaño intermedio como opciones familiares con buena conexión al centro y precios competitivos.
3. Propiedades alejadas (distancia alta):
    - Los resultados muestran que estas propiedades son atractivas para grupos grandes (6-8 huéspedes).
    - Predomina la preferencia por espacios amplios con 3 o más dormitorios.
    - Estrategia: Promocionar casas grandes y resaltar ventajas como privacidad, espacio exterior y tarifas atractivas para familias y grupos.

Algoritmos como K-Means y GMM destacan que el número de huéspedes aumenta conforme las propiedades están más alejadas del centro. El clustering por Agglomerative y DBSCAN refuerza la relación: mientras más céntrico, menor es el tamaño de los grupos y de los alojamientos demandados.

Caso de estudio 2

Análisis de alojamientos en el Albaicín: clasificación por calidad, tipo y categoría (mención)

En este caso de estudio nos centramos en analizar los alojamientos turísticos de Granada en la región del Albaicín, destino muy popular entre los visitantes de Granada. El objetivo es identificar patrones y relaciones entre la calidad de los alojamientos (medida en estrellas), su categoría (por ejemplo, básica, preferente, plus) y el tipo de establecimiento (apartamento u hotel, o ninguno).

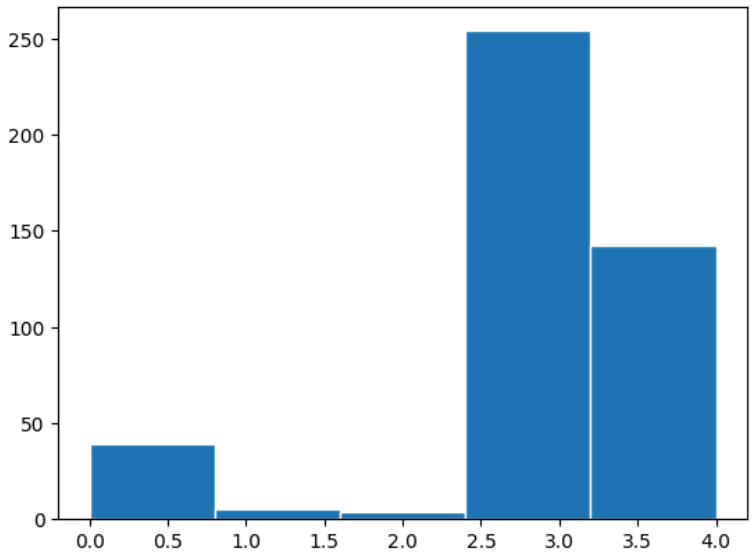
Este análisis busca determinar si la calidad de los alojamientos en el Albaicín depende significativamente del tipo de alojamiento o su categoría. Por ejemplo, podríamos descubrir si los alojamientos de categoría "plus" realmente ofrecen una calidad superior en comparación con los de categoría básica, o si el tipo de establecimiento (apartamento y hotel) tiene una mayor influencia en la valoración general.

El conjunto de datos seleccionado para este análisis contiene 444 registros, todos ellos correspondientes a alojamientos en el Albaicín. Puesto que hemos tenido que pasar algunas variables nominales a numéricas, la conversión ha sido la siguiente:

Tipos de alojamiento nominal: ['apartamento', NaN, 'hotel']
Tipos de alojamiento numérico: [0 -1 1]

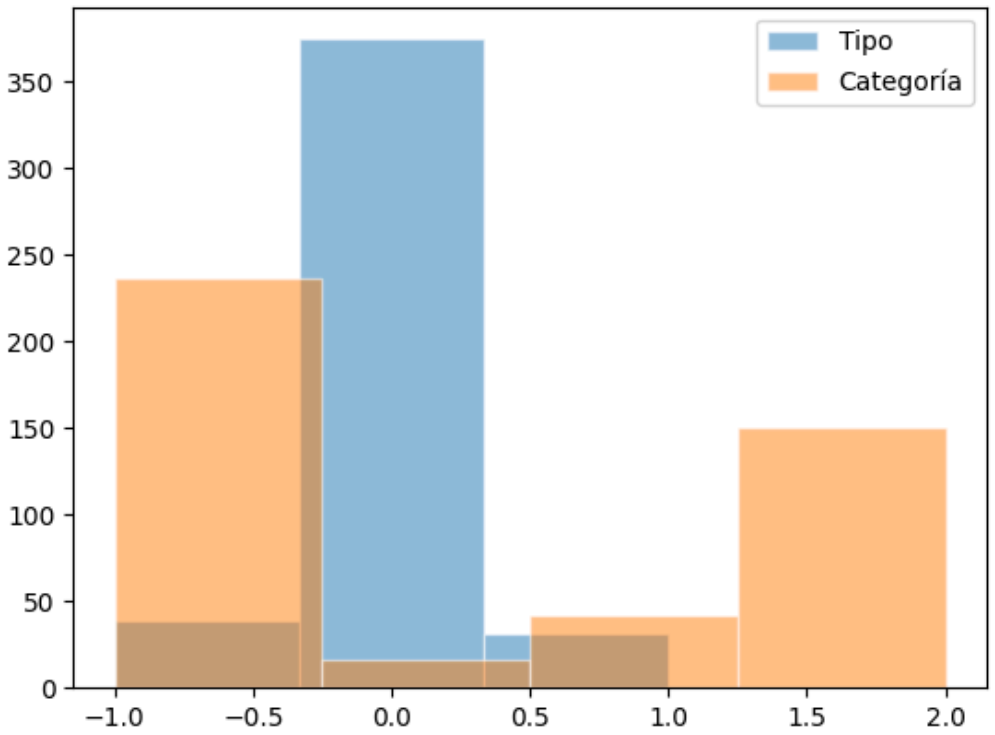
Categorías de alojamiento nominal: [NaN, 'plus', 'preferente', 'destacado']
Categorías de alojamiento numérico: [-1 1 2 0]

Observemos qué cantidad de estrellas tienen estos alojamientos del Albaicín para poder interpretar mejor la diferencia de los clusters:



Con este histograma identificamos un detalle muy importante: prácticamente no hay alojamientos con 2 ó 3 estrellas en calidad. La mayoría de alojamientos tienen 4 ó 5 estrellas y, casi 50 alojamientos tienen 1 estrella. Aún así, vemos que es más complicado obtener 5 estrellas que 4, ya que hay usuarios más exquisitos para dar la puntuación más alta de calidad. Veamos los histogramas para las otras características, configurados con el siguiente código:

```
unique1, counts = np.unique(datos2_cluster['tipo'], return_counts=True)
unique2, counts = np.unique(datos2_cluster['categoria'],
return_counts=True)
plt.hist(datos2_cluster['tipo'],alpha=0.5,edgecolor = "white",
bins=len(unique1), label='Tipo')
plt.hist(datos2_cluster['categoria'],alpha=0.5,edgecolor = "white",
bins=len(unique2), label='Categoría')
plt.legend(loc='upper right')
plt.show()
```



Las barras azules representan el tipo de alojamiento (ninguno -1, apartamento 0 y hotel 1). Vemos que la mayoría de alojamientos son apartamentos y hay igual número de hoteles que de alojamientos sin tipo. En cuanto a la categoría o mención (ninguna -1, destacado 0, plus 1 y preferente 2), casi la mitad de alojamientos no tienen mención y apenas unos pocos alojamientos pertenecen a ‘destacado’. Hay bastantes con categoría ‘preferente’ y unos 50 de categoría ‘plus’.

Para segmentar los alojamientos turísticos del Albaicín, se han empleado cinco algoritmos de clustering que ofrecen diferentes perspectivas sobre los datos. Esto nos permite comparar enfoques y seleccionar el más adecuado según la estructura de los datos y el objetivo del análisis:

- **K-Means**: Es uno de los métodos más utilizados debido a su rapidez y facilidad de implementación. Aunque asume que los clústeres tienen formas esféricas y homogéneas, su desempeño es adecuado para particionar conjuntos de datos bien definidos con tamaños de clústeres similares.
- **Agglomerative Clustering**: Este algoritmo jerárquico organiza los datos en una estructura basada en su similitud, facilitando la interpretación a través de dendrogramas. Es especialmente útil para identificar patrones jerárquicos complejos, aunque puede ser menos eficiente en conjuntos de datos extensos.
- **Spectral Clustering**: Aprovecha las propiedades espectrales de los grafos para segmentar los datos, lo que lo hace ideal para estructuras no lineales o con clústeres de formas irregulares. No obstante, su efectividad depende de la construcción adecuada del grafo de similitud.
- **BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)**: Diseñado para manejar grandes conjuntos de datos, este algoritmo utiliza un enfoque incremental que permite trabajar con memoria limitada. Es eficiente en términos computacionales, pero requiere un ajuste cuidadoso de parámetros como el umbral para obtener resultados precisos.
- **Gaussian Mixture Models (GMM)**: Este enfoque probabilístico nos ayuda a identificar clústeres no necesariamente esféricos. Es especialmente útil para datos con superposiciones, pero puede ser más costoso computacionalmente.

Aquí dejamos una tabla comparativo con los resultados obtenidos al aplicar estos algoritmos al subconjunto seleccionado:

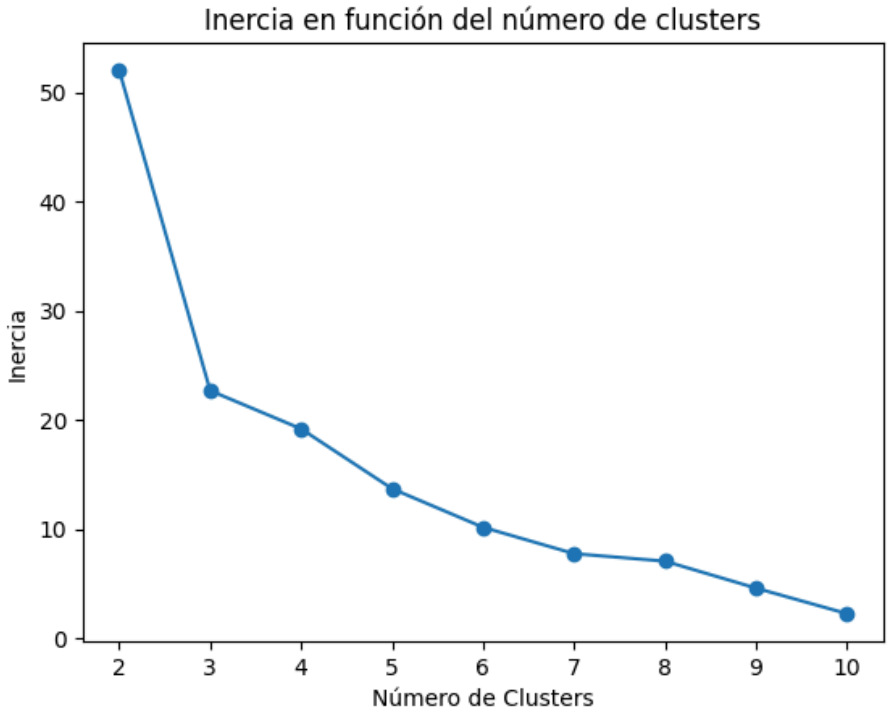
Algoritmo	Número clusters	Coeficiente Silhouette	Índice Calinski-Harabasz	Tiempo de ejecución (s)
k-Means	3/7	0.753/ 0.827	1185.076/ 1296.312	0.005808115005/ 0.0110557079315 1855
Agglomerative Clustering	2/4	0.629/ 0.676	684.049/956.672	0.008869647979 73632/ 0.016879558563 23242
Spectral Clustering	3	0.753	1185.076	0.1167244911193 8477
BIRCH	4	0.760	915.637	0.051466703414 91699
GMM	4	0.599	799.996	0.1116228103637 6953

Vemos que los algoritmos de K-Means y Spectral Clustering ofrecen la mejor calidad de clustering según el coeficiente de Silhouette y el índice de Calinski-Harabasz. Por otro lado, BIRCH también es bastante eficaz con respecto a Silhouette. El tiempo de ejecución del Agglomerative Clustering es

reducido, pero sus índices de calidad no son demasiado altos en comparación con el resto, exceptuando a GMM que tiene el peor rendimiento.

K-Means

Este algoritmo lo he usado para 3 clusters, que es el número óptimo según el método del codo, también implementado en el anterior caso de uso. El script usado ha sido el mismo, pero con el nuevo conjunto de datos. Podemos ver la gráfica de la inercia en función al número de clusters:



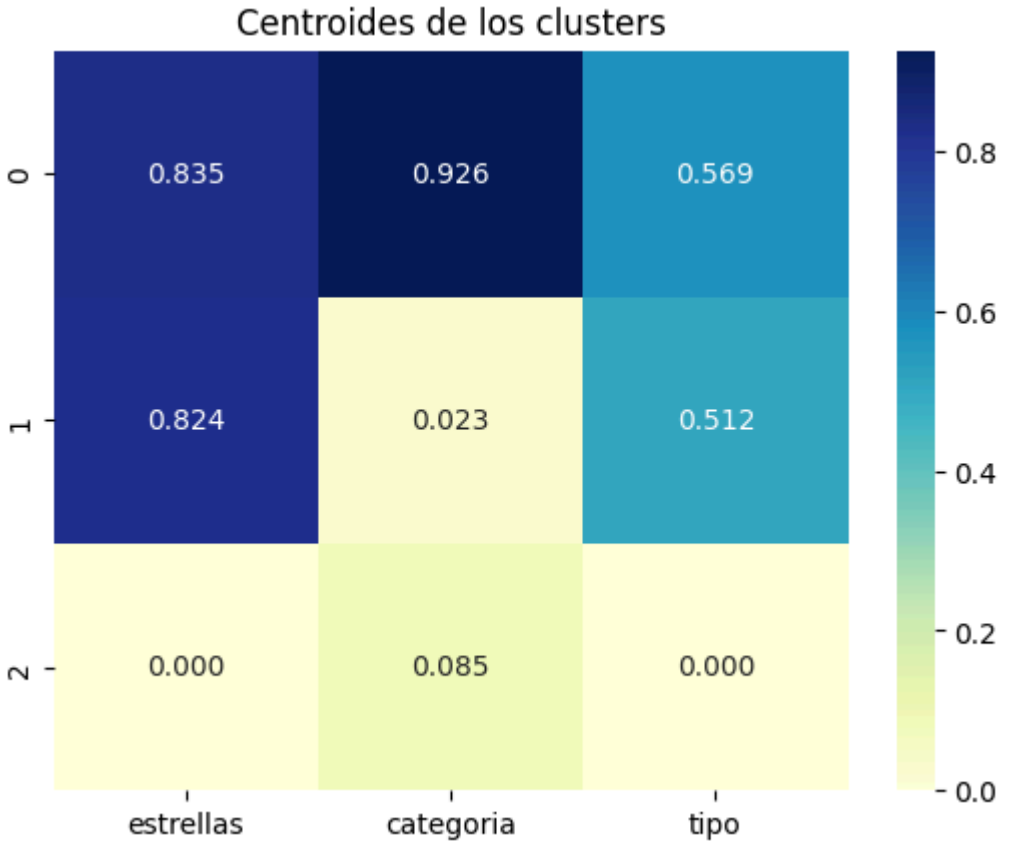
En la gráfica aparece claramente un codo en k=3, aunque probando más valores para el número de clusters he encontrado que para k=7 los coeficientes mejoran considerablemente.

Número de clusters = 3

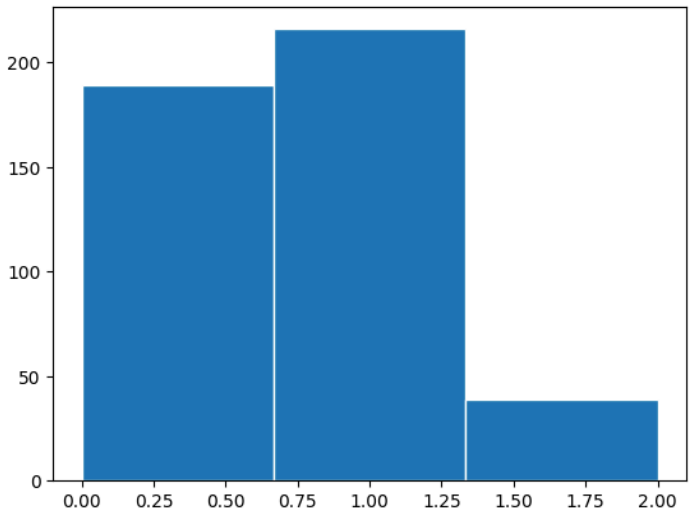
Al ejecutar el algoritmo con este número de clusters y las tres variables que queremos estudiar, obtenemos las siguientes métricas:

Calinski-Harabasz Index: 1185.076, Silhouette: 0.753, tiempo ejecución: 0.005808115005493164

Con estos valores interpretamos que los puntos están bien agrupados (ya que el índice de Silhouette es cercano a 1) y, además, los clusters están bien separados entre ellos (ya que el coeficiente de Calinski-Harabasz es alto). Por otro lado, para identificar los clusters desarrollados por el algoritmo, podemos visualizar un heatmap de centroides, así mostraremos los valores normalizados de los centroides de los clusters en función de las variables de calidad en estrellas, tipo de alojamiento y categoría.

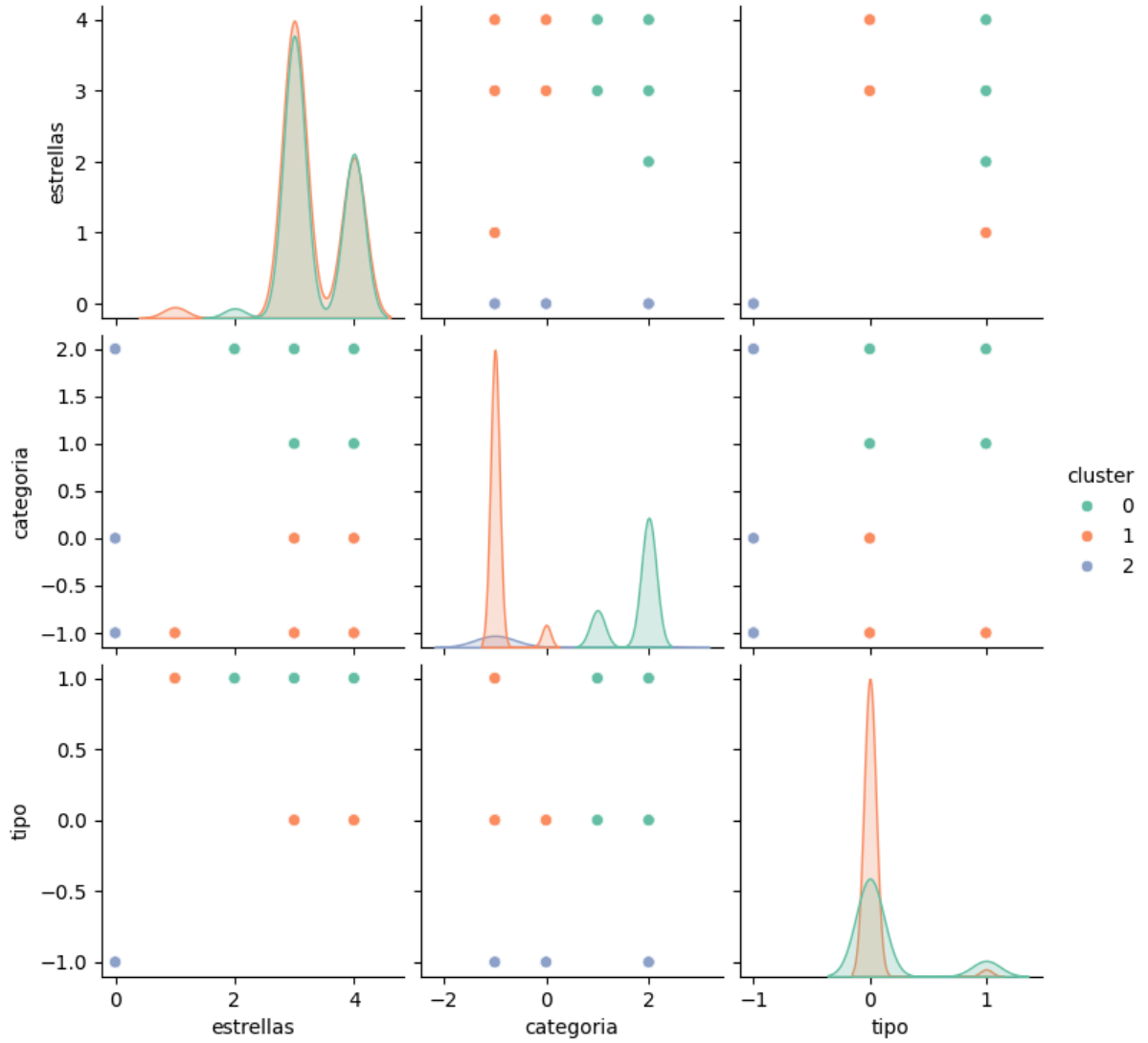


Vemos que el cluster 0 representa las reservas de alojamientos en el Albaicín donde el número de estrellas en calidad es alto, la categoría es un número alto (pertenecen a alguna categoría) y el tipo de alojamiento es tanto apartamento como hotel. Es decir, el cluster 0 contiene los alojamientos con mayor calidad y una categoría superior. Por otro lado, el cluster 1 contiene aquellos alojamientos con un número de estrellas alto, pero su categoría es ninguna o, como mucho, 'destacado'. Finalmente, el cluster 2 contiene alojamientos con missing values, con 1 estrella y de tipo nulo. Veamos en un histograma cuantos alojamientos pertenecen a cada cluster.



La mayoría de los datos se encuentran en los dos primeros clusters que, recordemos, tiene el número de estrellas en calidad muy alto. Es decir, que independientemente de la mención que tenga el alojamiento, la calidad es muy parecida. Aunque con una pequeña mejora en alojamiento ‘plus’ y ‘preferente’, que en ‘destacado’.

Veamos también la distinción de los clusters mediante un gráfico de dispersión (scatter matrix).



Como esperábamos, tenemos 3 clusters diferenciados por estrellas, categoría (mención) y tipo. Los clusters se diferencian principalmente por la categoría: si es una categoría baja (nula o ‘destacado’), entonces el alojamiento pertenece al cluster 1; si es una categoría alta (‘plus’ o ‘preferente’) entonces pertenece al cluster 0. El cluster 2 contiene alojamientos sin tipo y 0 estrellas en calidad, además la categoría es cualquiera excepto ‘plus’. Podemos rellenar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (Mención)	Tipo de alojamiento
0	4-5	‘plus’ o ‘preferente’	Apartamento/ hotel
1	4-5	ninguna o ‘destacado’	Apartamento/ hotel

2	0	todas excepto ‘plus’	ninguno

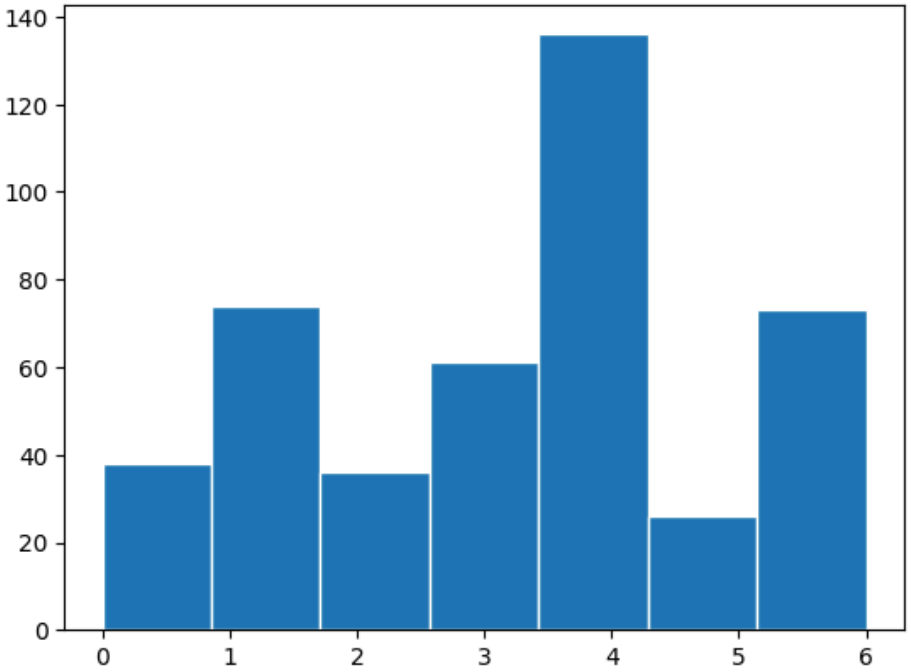
Esta clasificación puede resultar de gran ayuda a los turistas, ya que vemos que la calidad de los apartamentos y hoteles es prácticamente la misma. Sin embargo, los alojamientos que no tienen ningún tipo asignado tienden a pertenecer al cluster 2, que tiene la peor calidad. Por otro lado, también deberíamos guiarnos por la categoría (mención) del apartamento para obtener la mejor calidad, puesto que aquellos con mención ‘plus’ no pertenecen al cluster 0 y, por tanto, nos asegura muy alta calidad.

Número de clusters = 7

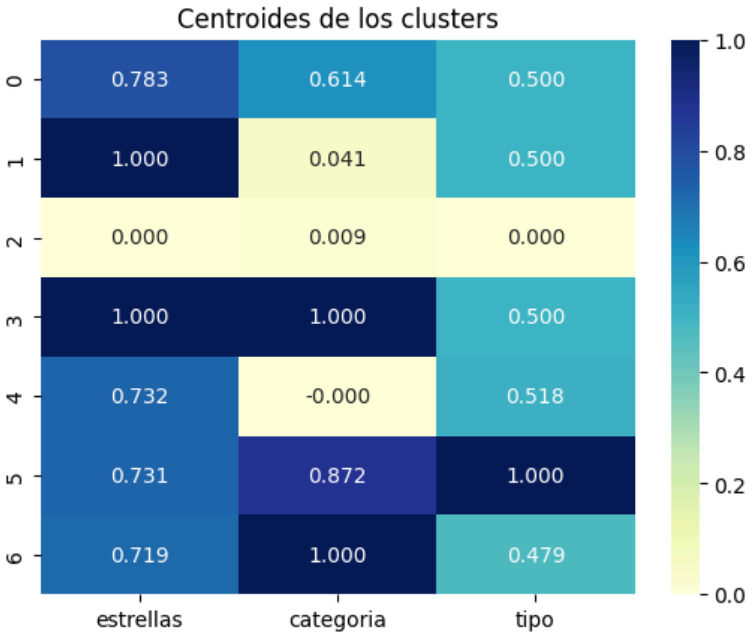
Al ejecutar el algoritmo de k-Means para este número de clusters los valores de las medidas de rendimiento suben a los siguientes:

Calinski-Harabasz Index: 1296.312, Silhouette: 0.827, tiempo ejecución: 0.011055707931518555

Los valores son un poco mejores que en el apartado anterior. Veamos cuántos alojamientos pertenecen a cada clusters mediante un histograma, como en el apartado anterior:

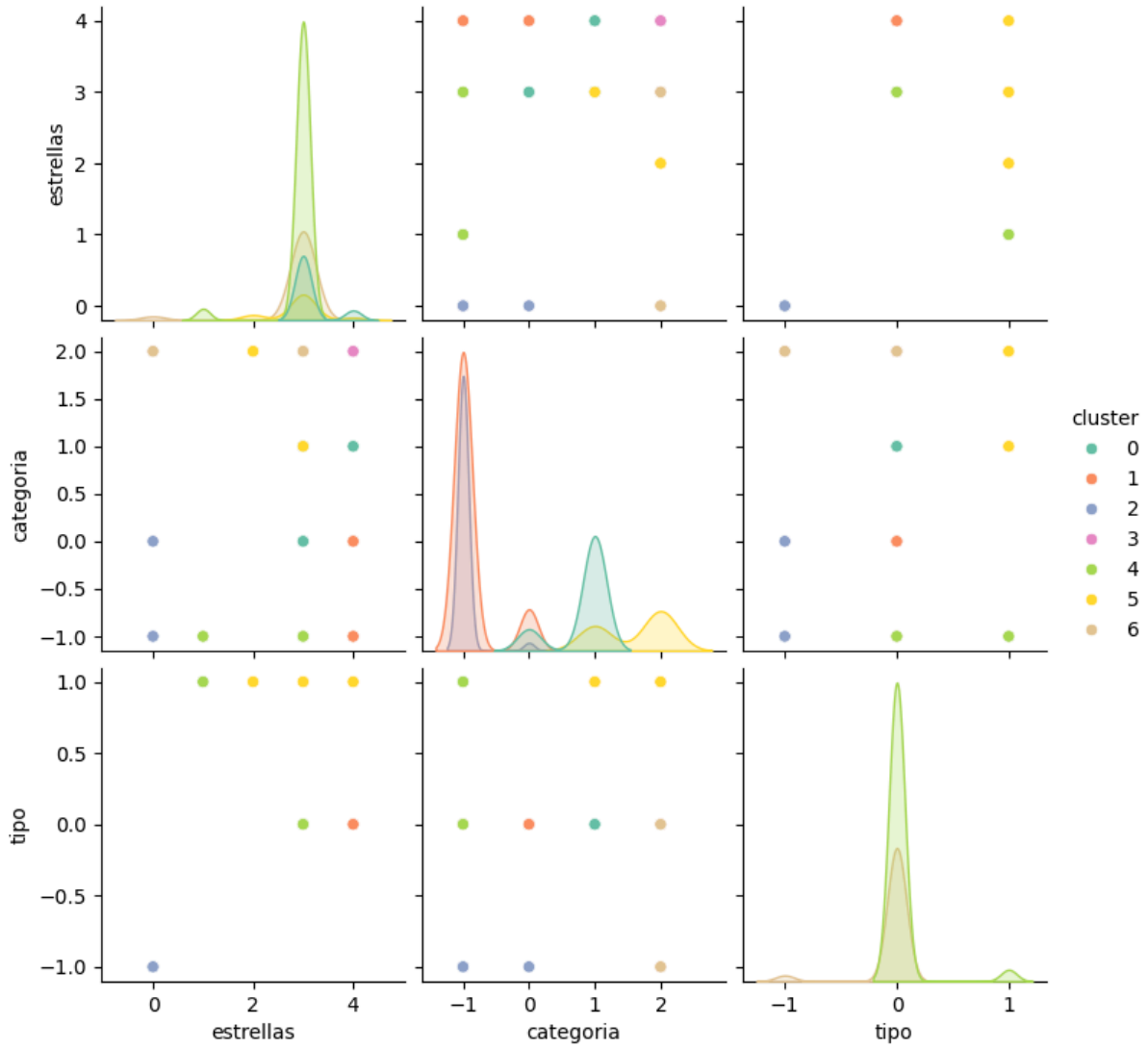


Tenemos que el cluster con más alojamientos es el 4 y el 5 es el que menos tiene. El resto están bastante repartidos por el resto de clusters. Con un heatmap vamos a estudiar qué alojamientos pertenecen a cada cluster.



Tenemos siete clusters divididos por número de estrellas en calidad, categoría o mención del alojamiento y su tipo. Hay dos clusters que destacan sobre el resto: el cluster 2 contiene a los alojamientos con 1 estrella y sin tipo ni categoría asignados; el cluster 3, por el contrario, tiene 5 estrellas en todos los alojamientos y categoría 'preferente' y está contenido por apartamentos. Otro cluster que contiene todo apartamentos de 5 estrellas es el cluster 1, pero la categoría es ninguna en su mayoría. El cluster 4 también tiene alojamientos sin categoría y contenido por apartamentos (aunque algún hotel) y de calidad alta, aunque más cercano a 4 estrellas que a 5. El cluster 5 tiene todo hoteles y categoría alta, sin embargo eso no nos asegura una calidad de 5 estrellas, puesto que se acerca más a 4 estrellas. Entre el cluster 0 y 6 la diferencia radica en la categoría: si es 'preferente' pertenece probablemente al cluster 6, y si es 'plus' o 'destacado' pertenece al cluster 0.

Para respaldar nuestra clasificación de clusters, creamos un scatter matrix:



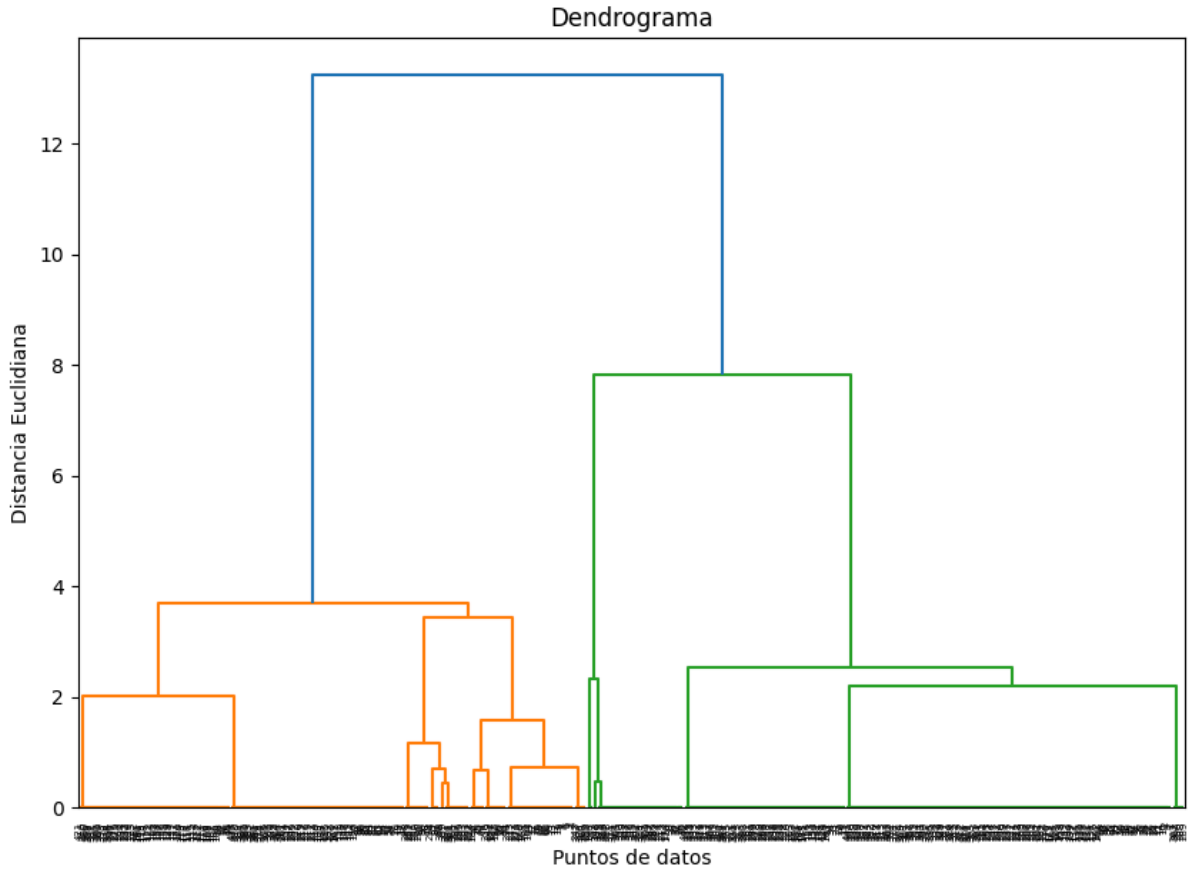
Y con esto podemos rellenar nuestra tabla de clusters.

Cluster	Calidad (en estrellas)	Categoría (mención)	Tipo
0	4-5	Destacado/plus	apartamento
1	5	ninguna/destacado	apartamento
2	1	ninguna/destacado	ninguno
3	5	preferente	apartamento
4	2-4	ninguna	apartamento/hotel
5	3-5	plus/preferente	hotel
6	4	preferente	ninguno/apartamento

Con esta tabla podemos destacar varias observaciones con respecto a los alojamientos en Booking en la localidad del Albaicín. Por un lado tenemos que si el alojamiento no tiene un tipo asignado, casi te asegura que su calidad va a ser la más baja (una estrella). Por otro lado, la diferencia entre un hotel con calidad alta y media tiene una gran relación con la categoría (mención) que tenga asignada: si no tiene ninguna categoría seguramente tendrá menos calidad que si tiene categoría de plus o preferente.

Agglomerative Clustering

Este tipo de clustering es jerárquico y, por tanto, deberíamos dibujar el dendograma de los datos para poder elegir el parámetro del número de clusters.



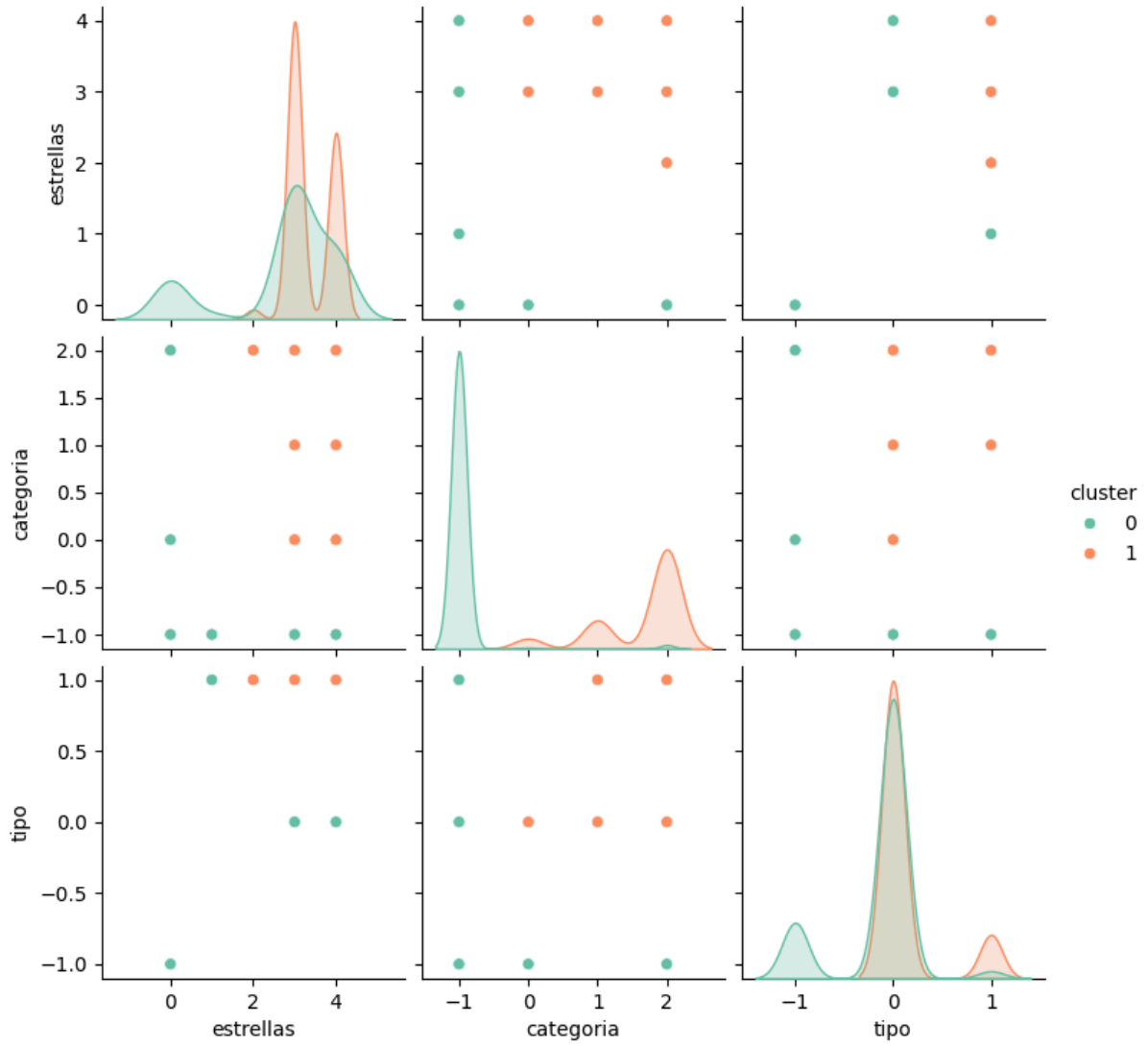
Vemos que la diferencia más grande entre rectas horizontales está entre la primera azul y la primera verde, aunque una diferencia muy parecida hay entre la primera verde y la primera naranja. Así que, estudiemos el número de clusters correspondientes a ambas diferencias: k=2 y k=4.

Número de clusters = 2

Los valores de las medidas de rendimiento para éste número de clusters son los siguientes:

Calinski-Harabasz Index: 684.049, Silhouette: 0.629, tiempo ejecución: 0.008869647979736328

Son valores considerablemente más bajos que los del algoritmo K-Means, probablemente debido a la falta de clusters para clasificar los datos. Veamos los datos pertenecientes a ambos clusters mediante una scatter matrix.



Vemos que el cluster 0 contiene alojamientos de cualquier cantidad de estrellas, incluyendo 1 estrella. Además, contiene en su mayoría alojamientos sin categoría y que son en su mayoría apartamentos, aunque también contiene a todos los alojamientos sin tipo asignado. Por otro lado, el cluster 1 contiene apartamentos y hoteles con 3, 4 ó 5 estrellas y que pertenecen, al menos, a alguna categoría. De esta forma podemos rellenar la tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Tipo
0	1-5	ninguna	ninguna/apartamentos
1	3-5	destacado/plus/preferente	apartamento/hotel

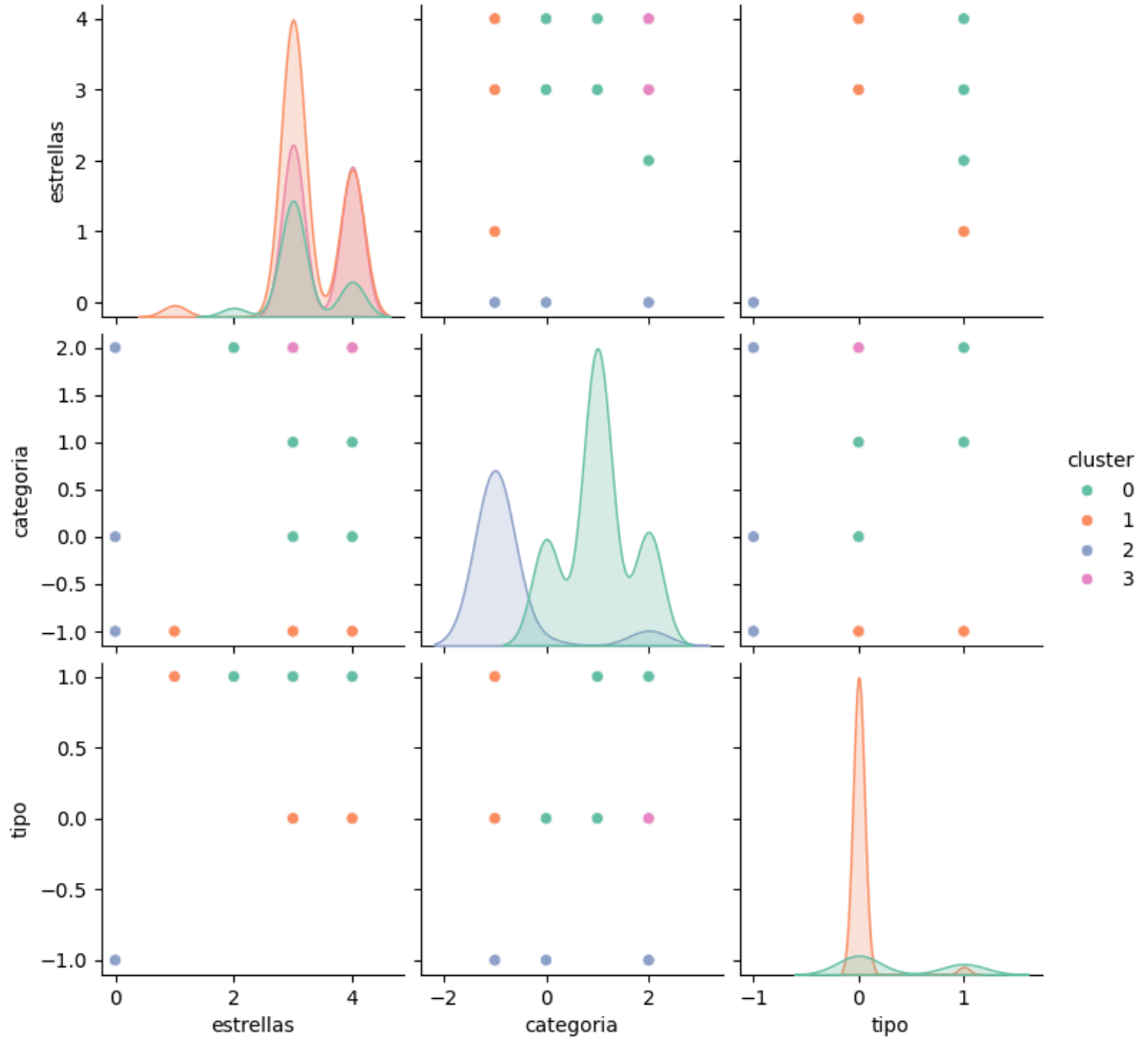
Esto quiere decir que si el alojamiento tiene alguna mención seguramente pertenece al cluster 1 que tiene una calidad mínima de 3 estrellas; si es hotel también pertenece a este cluster. Por el contrario, si el alojamiento no tiene tipo asignado, seguramente pertenece al cluster 0 que no te asegura ninguna calidad mínima ni máxima.

Número de clusters = 4

En este caso, los valores de rendimiento del clustering son:

Calinski-Harabasz Index: 956.672, Silhouette: 0.676, tiempo ejecución: 0.016879558563232422

Ligeramente mejores que en el caso anterior, pero considerablemente peores que en cualquier caso de K-Means. Veamos los datos pertenecientes a cada cluster con un scatter matrix:



Efectivamente encontramos cuatro cluster. El cluster 0 se diferencia del resto por contener alojamientos con alguna categoría asignada y algún tipo asignado y, en consecuencia, calidad de 3 estrellas como mínimo. El cluster 1 contiene mayoritariamente apartamentos de cualquier número de estrellas y sin categoría asignada. El cluster 2 contiene alojamientos de 1 estrella que en su mayoría no tienen una categoría asignada ni un tipo asignado. Por último, el cluster 3 contiene apartamentos de 4-5 estrellas que son de categoría ‘preferente’. Podemos generar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Tipo
0	3-5	cualquiera	apartamento/hotel
1	2-5	ninguna	apartamento/hotel
2	1	todas menos ‘plus’	ninguno
3	4-5	ninguna	apartamento

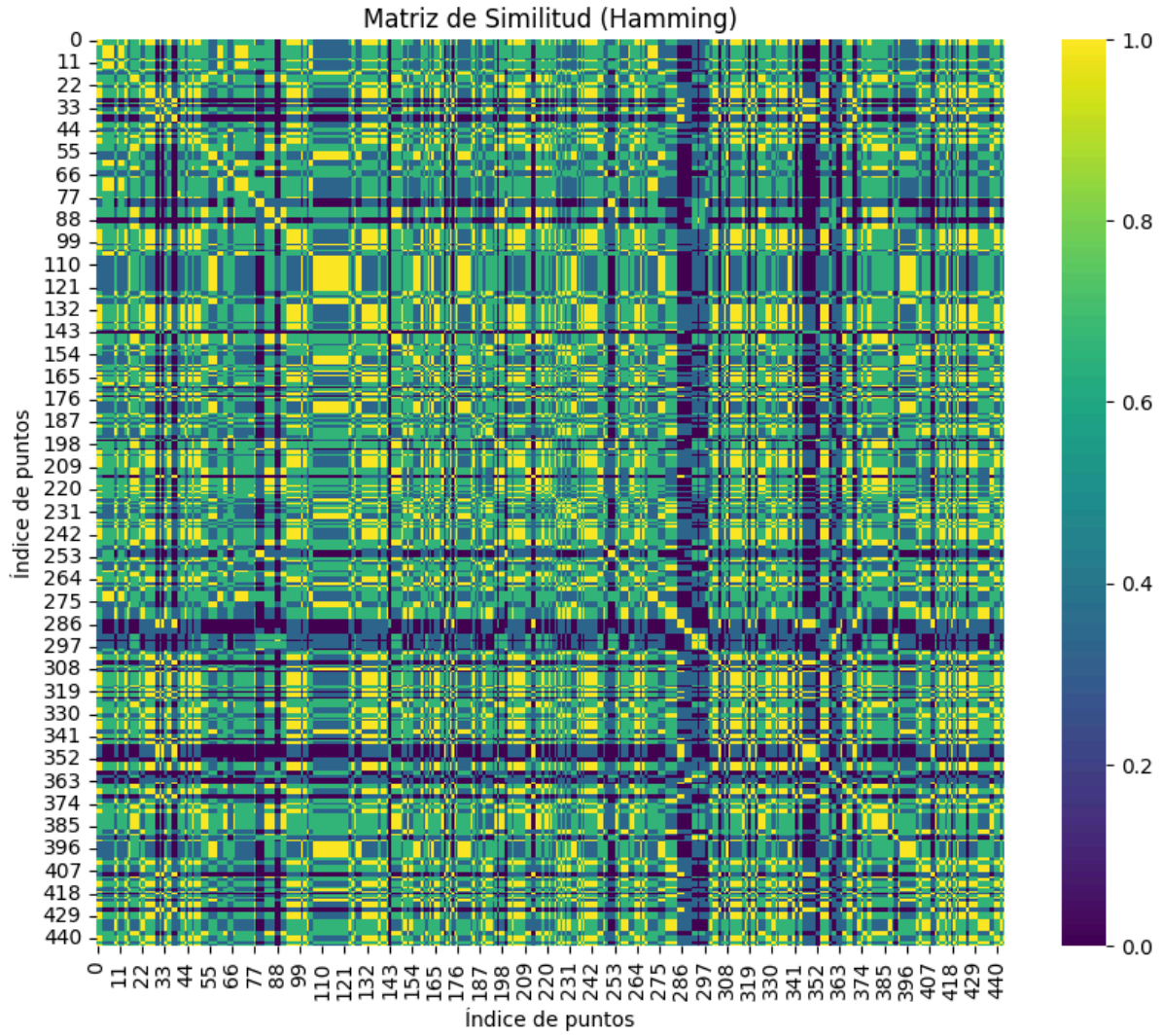
Con esta tabla podemos concluir con aspectos muy parecidos a los del apartamento anterior. Si el tipo es ninguno, entonces la calidad probablemente sea de una estrella. Además, no se encuentran alojamientos con categoría ‘plus’ que tengan una estrella. Por otro lado, si el alojamiento tiene alguna categoría asignada, entonces podemos asegurarnos una calidad mínima de 3 estrellas. Los otros dos clusters que aparecen distintos a los del apartado de k=2 no contienen una información demasiado valiosa.

Spectral Clustering

Spectral Clustering es un algoritmo de agrupamiento que utiliza las propiedades espectrales de un grafo construido a partir de los datos. En lugar de agrupar directamente los puntos en su espacio original, Spectral Clustering representa los datos en un espacio de menor dimensión basado en los eigenvectors de una matriz de similitud (como el Laplaciano del grafo). Este enfoque es especialmente útil para detectar estructuras complejas o no lineales en los datos, que podrían ser difíciles de identificar con algoritmos como K-Means.

Cuando los datos son discretos, como en nuestro caso, métricas de similitud como la distancia Hamming son adecuadas. La distancia Hamming mide la diferencia entre dos cadenas de bits o valores categóricos, contando las posiciones donde los valores son diferentes. Esto la hace ideal para datos categóricos o binarios, permitiéndonos capturar las similitudes en los patrones de las variables discretas y construir una matriz de similitud que Spectral Clustering pueda procesar. Este enfoque garantiza que las características de los datos se reflejen correctamente en el análisis del clustering. Para visualizarla he usado el siguiente script:

```
from sklearn.metrics.pairwise import pairwise_distances

# Crear matriz de similitud basada en distancia de Hamming
similarity_matrix = 1 - pairwise_distances(X2_normal, metric='hamming')
# Valores entre 0 y 1

plt.figure(figsize=(10, 8))
sns.heatmap(similarity_matrix, cmap='viridis', square=True)
plt.title("Matriz de Similitud (Hamming)")
plt.xlabel("Índice de puntos")
plt.ylabel("Índice de puntos")
plt.show()
```



Vemos que en varias secciones de la diagonal principal, aparecen bloques de celdas con valores altos (amarillos), loque indica que hay subconjuntos de datos que son muy similares entre sí, posiblemente perteneciendo a un mismo grupo o cluster. Sin embargo, hay muchos valores dispersos de similitud baja (colores oscuros) y valores medianos (tonos verdes), lo que sugiere que hay datos menos relacionados o que son puntos equidistantes de varios clusters. Vemos que hay áreas claras de mayor similitud distribuidas a lo largo de la diagonal, esto quiere decir que los datos podrían agruparse en 3, teniendo en cuenta los límites entre las zonas menos homogéneas (colores oscuros).

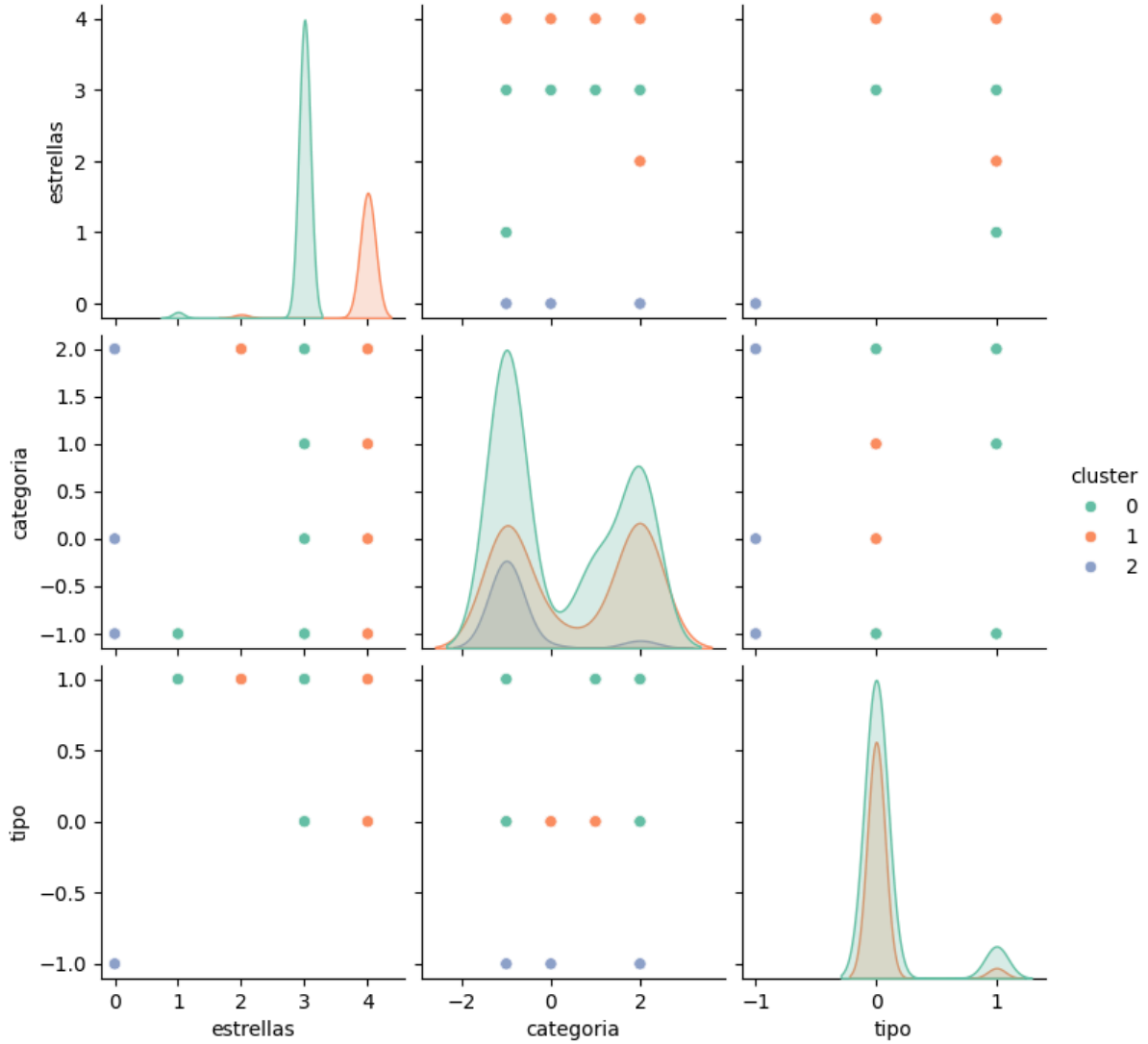
Con este número de clusters podemos ejecutar el algoritmo de la siguiente forma:

```
spectral = SpectralClustering(n_clusters=3, affinity='precomputed',
random_state=42)
labels = spectral.fit_predict(similarity_matrix)
```

Y evaluamos según las medidas de rendimiento obteniendo los siguientes valores:

Calinski-Harabasz Index: 1185.076, Silhouette: 0.753, tiempo ejecución: 0.11672449111938477

Lo que indica que la segmentación es bastante buena, con un índice de Silhoutte bastante alto, confirmando buena cohesión. Para visualizar los puntos pertenecientes a los diferentes clusters podemos usar una scatter matrix.



Vemos que el cluster 0 está compuesto por apartamentos y hoteles que tienen 4 estrellas (y los de 2 estrellas) y de cualquier categoría. Por otro lado, el cluster 1 contiene apartamentos y hoteles de 5 estrellas (y los de 3 estrellas) y de cualquier categoría. Finalmente, el cluster 2 contiene alojamientos sin tipo que tienen 1 estrella y de cualquier categoría exceptuando ‘plus’. Con esto podemos rellenar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Tipo
0	2 y 4	cualquiera	apartamento/ hotel
1	3 y 5	cualquiera	apartamento/ hotel
2	1	cualquiera excepto ‘plus’	ninguna

Vemos que la clasificación no es tan clara como en los anteriores algoritmos, ya que la diferencia entre los clusters 0 y 1 no está demasiado definida. El cluster 2, por el contrario, sí que está definido de igual forma que en el resto de algoritmos.

BIRCH

BIRCH (balanced Iterative Reducing and Clustering using Hierarchies) es un algoritmo de clustering que puede agrupar grandes conjuntos de datos generando primero un resumen pequeños y compacto del gran conjunto de datos que retiene la mayor cantidad de información posible. BIRCH resume grandes conjuntos de datos en regiones más pequeñas y densas llamadas entradas de Característica de agrupamiento (CF). Por otro lado, un árbol CF es un árbol donde cada nodo de hoja contiene un subgrupo. Cada entrada en un árbol de CF contiene un puntero a un nodo hijo y una entrada CF compuesta por la suma de entradas de CF en los nodos hijos.

Ahora podemos entender los parámetros de BIRCH:

- threshold: el máximo número de puntos de datos que un subgrupo en el nodo hoja puede contener.
- branching_factor: especifica el máximo número de subgrupos en cada nodo.
- n_clusters: el número de clusters que devuelve el algoritmo. Si no se especifica, el último paso del clustering no se ejecuta y se devuelven los clusters intermedios.

Para elegir los algoritmos he implementado un script que evalúa el rendimiento del algoritmo con distintos valores para los parámetros de BIRCH y elige cuál es el mejor:

```
from itertools import product

# Rango de parámetros a evaluar
threshold_values = np.arange(0.05, 0.5, 0.05)  # Valores de threshold
branching_factors = [10, 20, 30, 50]  # Valores de
branching_factor
n_clusters_values = [2, 3, 4, 5, 6]  # Valores de n_clusters

# Variables para almacenar los mejores resultados
best_score = -1
best_params = None

# Iterar sobre todas las combinaciones de parámetros
for threshold, branching_factor, n_clusters in
product(threshold_values, branching_factors, n_clusters_values):
   # Crear modelo BIRCH
   birch_model = Birch(threshold=threshold,
branching_factor=branching_factor, n_clusters=n_clusters)
   birch_model.fit(X2_normal)

   # Predecir etiquetas
   labels = birch_model.predict(X2_normal)

   # Evitar cálculos si solo hay un clúster
   if len(np.unique(labels)) <= 1:
       continue
```

```
    # Calcular la puntuación Silhouette
    score = metrics.silhouette_score(X2_normal, labels)

    # Guardar los mejores parámetros
    if score > best_score:
        best_score = score
        best_params = {
            "threshold": threshold,
            "branching_factor": branching_factor,
            "n_clusters": n_clusters,
        }

# Mostrar los mejores parámetros y el puntaje correspondiente
print("Mejores parámetros:")
print(f"Threshold: {best_params['threshold']}")
print(f"Branching Factor: {best_params['branching_factor']}")
print(f"N Clusters: {best_params['n_clusters']}")
print(f"Mejor Silhouette Score: {best_score:.4f}")
```

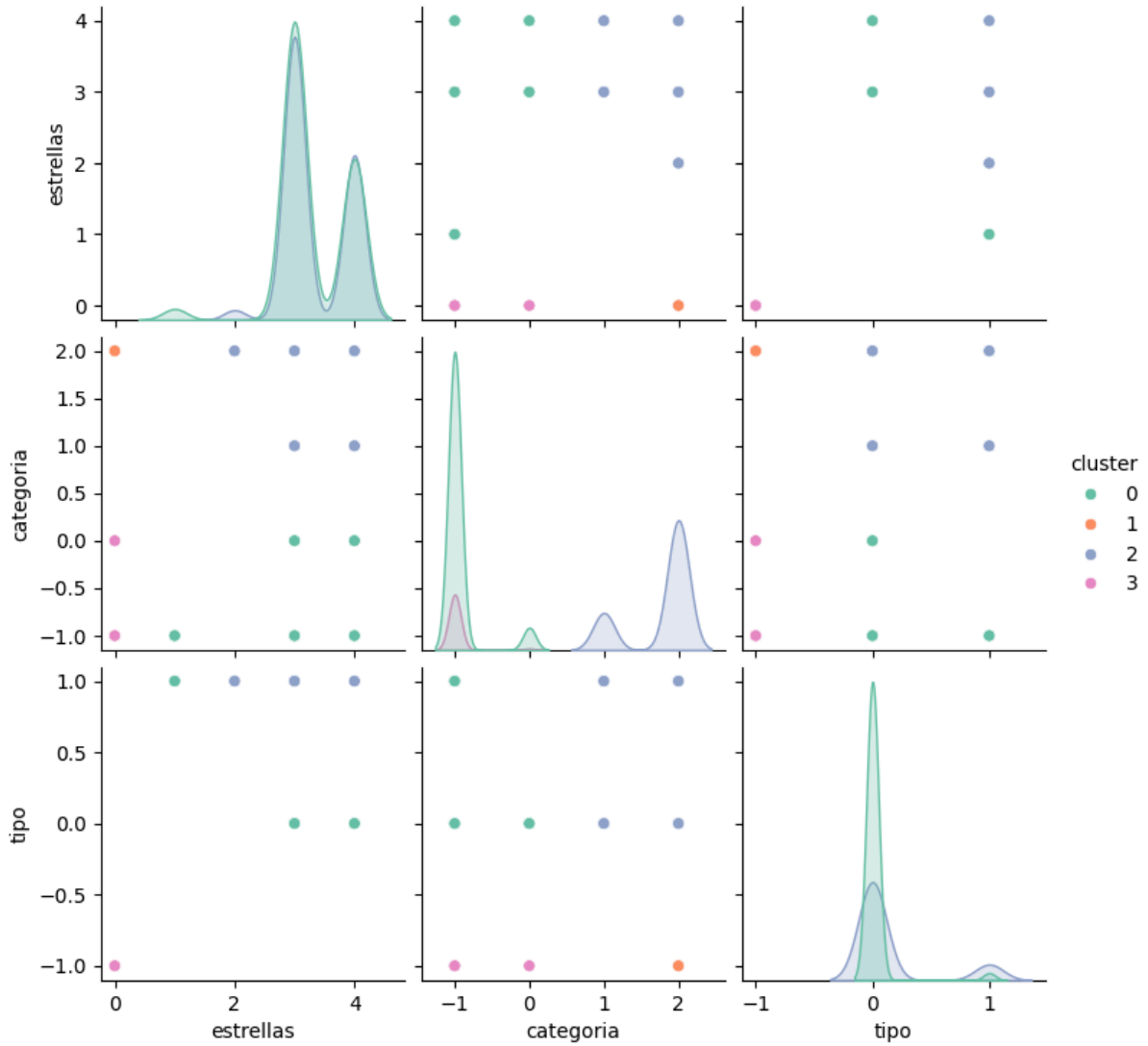
El resultado es el siguiente:
Mejores parámetros:

- Threshold: 0.15000000000000002
- Branching Factor: 10
- N Clusters: 4
- Mejor Silhouette Score: 0.7601

Lo que nos permite utilizar esos parámetros para usar en el algoritmo y calcular el resto de medidas:

Calinski-Harabasz Index: 915.637, Silhouette: 0.760, tiempo ejecución: 0.05146670341491699

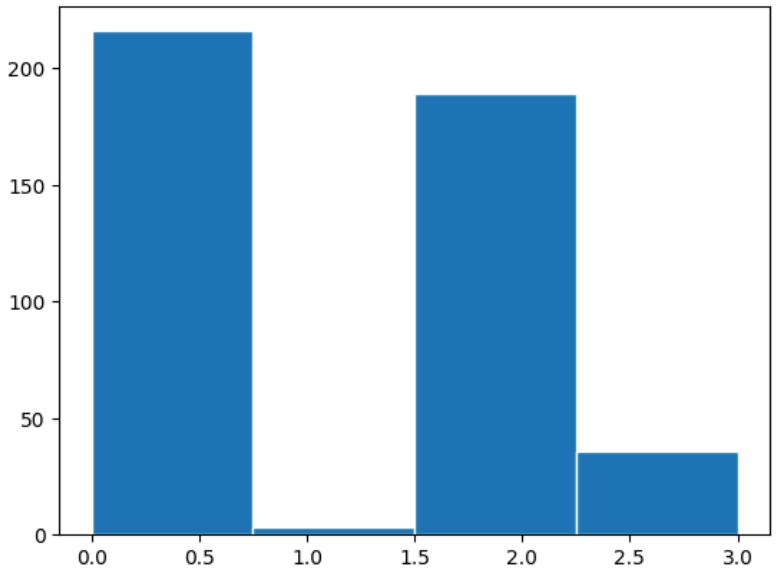
El coeficiente de Calinski-Harabasz no es bastante alto, pero el de Silhouette sí que lo es. Veamos mediante un scatter matrix cuáles son los datos pertenecientes a cada cluster.



El cluster 0 está compuesto por apartamentos y hoteles de 4-5 estrellas cuya categoría es nula o ‘destacado’, mientras que el cluster 2 se diferencia de éste por contener a los de categoría ‘plus’ o ‘preferente’. El cluster 1 contiene alojamientos sin tipo y de categoría ‘preferente’ que tienen 1 estrella. Finalmente, el cluster 3 está compuesto por alojamientos sin tipo pero de categoría ‘destacado’ o nula, que también tienen 1 estrella. Con esto, podemos rellenar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Tipo
0	4-5	nula o ‘destacado’	apartamento/hotel
1	1	‘preferente’	ninguno
2	4-5	nula o ‘destacado’	apartamento/hotel
3	1	nula o ‘destacado’	ninguno

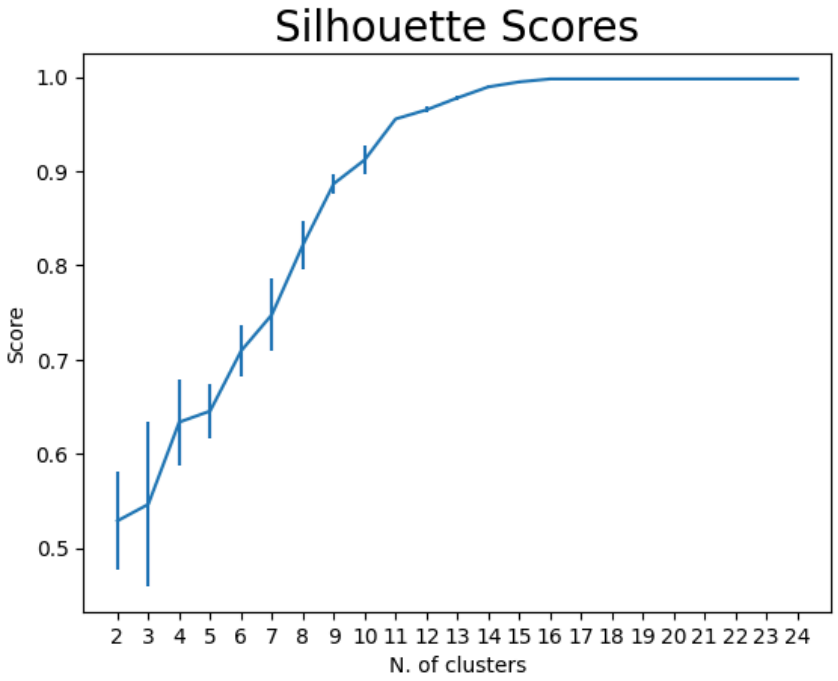
Veamos cuántos alojamientos hay en cada cluster mediante un histograma:



Vemos que el cluster 1 está prácticamente vacío, de hecho, contiene exactamente 3 alojamientos con estas características, por lo que no es muy representativo de los datos. El cluster 0 y 1 contiene la mayoría de los datos, lo que tiene sentido ya que la mayoría de los alojamientos tenían entre 4 y 5 estrellas. El último cluster contiene 36 alojamientos con dichas características, lo que representa ue la mayoría de alojamientos con 1 estrella en calidad no tienen un tipo asignado y su categoría es, como mucho, 'destacado'.

GMM

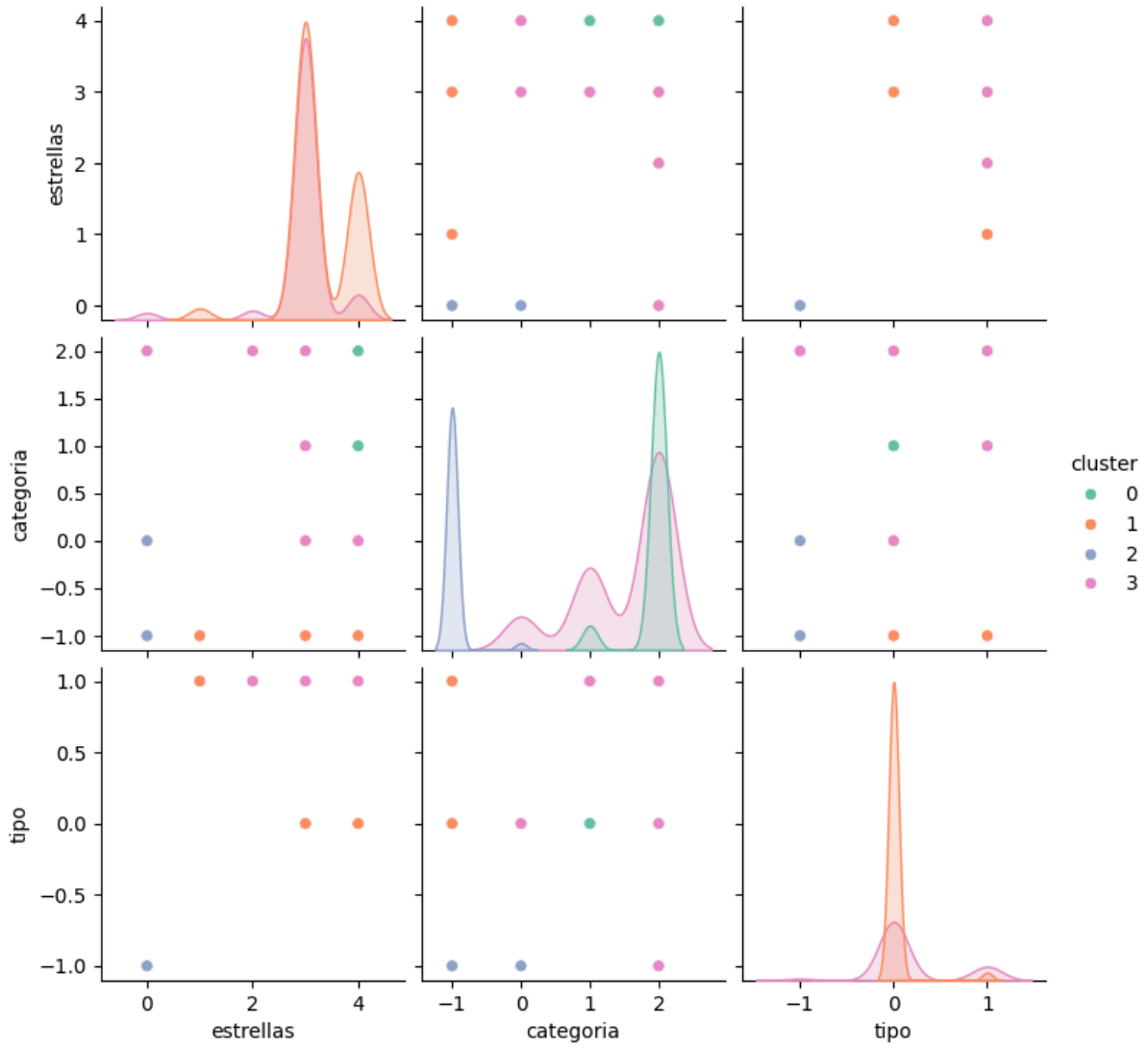
En este algoritmo, ya explicado en el caso de estudio anterior, usaremos la puntuación de la silueta para encontrar el número óptimo de clusters.



El gráfico tiene sentido ya que, como hay pocos valores de cada variable, podríamos incluir hasta 5*4*3=60 clusters que son todos los posibles valores de cada uno de los alojamientos y estaría perfectamente diferenciado. Sin embargo, para que la clusterización sea interpretable y mantenga la relación con los anteriores algoritmos, usaremos 4 clusters. Al ejecutar el algoritmo con este parámetro, los valores de las medidas de rendimiento son las siguientes:

Calinski-Harabasz Index: 799.996, Silhouette: 0.599, tiempo ejecución: 0.11162281036376953

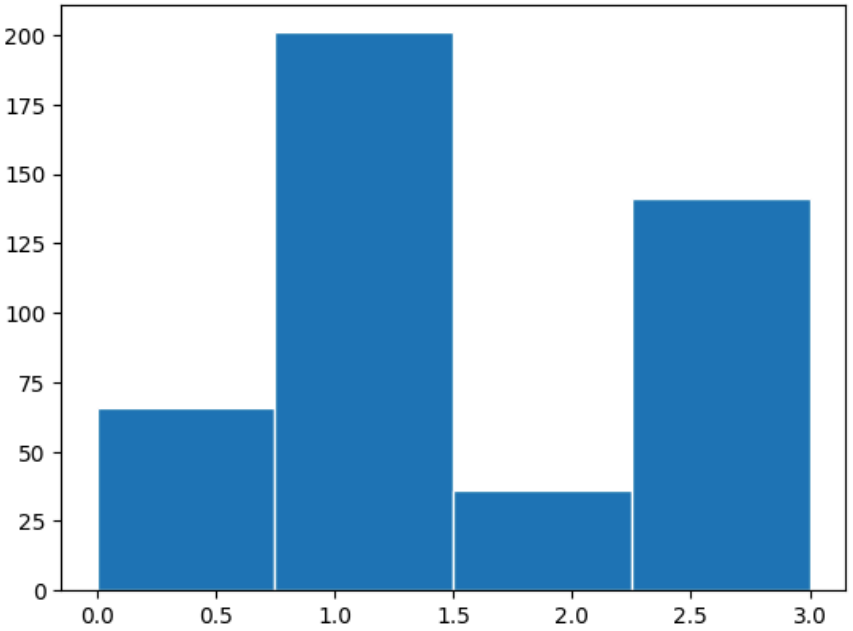
Son valores bajos comparados con los anteriores algoritmos. Veamos con el scatter matrix para reconocer los diferentes clusters.



Vemos los 4 clusters bien diferenciados. Por un lado, tenemos el cluster 0 con apartamentos de 5 estrellas y de categoría ‘preferente’ y ‘plus’. Por otro lado, tenemos el cluster 1 conteniendo apartamentos y hoteles de más de 2 estrellas y que tienen alguna categoría asignada. El cluster 2 son alojamientos sin tipo y de categoría nula o ‘destacado’ y con 1 estrella. Finalmente, el cluster 3 tiene alojamientos de todos los tipos, todos los posibles valores de calidad y de cualquier categoría, así que podemos suponer que son los outliers. Obtenemos entonces la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Tipo
0	5	‘preferente’ o ‘plus’	apartamento
1	2-5	‘preferente’/’plus’/’destacado’	apartamento/ hotel
2	1	ninguna	ninguno
3	cualquiera	cualquiera	cualquiera

Veamos la cantidad de elementos en cada cluster mediante un histograma.



Los clusters con más elementos son el 1 y el 3, tiene sentido puesto que son los más generales. Vemos que esta clasificación dada por el algoritmo no es especialmente reveladora, ya que no distingue ningún cluster demasiado sobre el resto y no se pueden sacar conclusiones valiosas.

Interpretación de la segmentación

Una vez hemos analizado los resultados de todos los algoritmos, podemos concluir con una recomendación para los turistas basándonos en ellos. Para elegir el mejor alojamiento, los turistas deberían centrarse en los algoritmos que muestran una segmentación más clara y confiable en cuanto a calidad, como K-Means con 3 clusters y Agglomerative Clustering.

Si buscamos un alojamiento de alta calidad (4 ó 5 estrellas), los mejores resultados provienen de los clusters 0 de K-Means (con 3 clusters), Agglomerative Clustering (con 4 clusters) y BIRCH. Basándonos en estos, deberíamos asegurarnos de que el alojamiento tenga una mención ‘plus’ o ‘preferente’, ya que éstos suelen estar asociados a una calidad más alta.

Por otro lado, si un alojamiento no tiene tipo asignado o categoría, es probable que pertenezca a clusters 2 ó 3 en los algoritmos de K-Means y Agglomerative Clustering, lo que indica una calidad baja, generalmente 1 estrella.

En cuanto a los algoritmos Spectral Clustering, GMM y BIRCH, éstos no ofrecen una segmentación tan clara ni útil para los turistas, ya que no aportan distinciones definitivas entre los clusters.

Caso de estudio 3

Análisis de Segmentación de Apartamentos Turísticos en Granada: Albaicín y Centro

En este caso de estudio, nos enfocamos en analizar y comparar los apartamentos turísticos situados en dos áreas emblemáticas de Granada: el Albaicín y el Centro. El objetivo es identificar patrones y segmentos de interés basándonos en variables clave como la calidad de los apartamentos (medida en estrellas), la categoría (como “plus” o “preferente”), el área de los mismos y el precio medio por noche en todas las búsquedas en las que aparece el apartamentos . Este análisis tiene como finalidad explorar cómo estas características influyen en la oferta turística y en la preferencia de los visitantes en cada ubicación.

```
apartamentos_albaicin = datos[datos.Location.str.contains('Albaicín') & datos.Type.str.contains('apartamento')]
apartamentos_centro = datos[datos.Location.str.contains('Centro de Granada') & datos.Type.str.contains('apartamento')]


# Selección de columnas y renombrado
columnas_renombradas = {"Price avg": "precio", "Rating": "valoracion",
                        "Ranking position avg": "ranking", "Total Beds": "dormitorios",
                                                                "Type":"tipo", "Quality":"estrellas","Guests":"huespedes",
                                        "Distance":"distancia", "Number of views":"visitas",
                                           "Review":"reseñas","Surface Area (m2)":'area',"Special":'categoria'}

datos3_cluster_alb = apartamentos_albaicin.rename(columns=columnas_renombradas)
datos3_cluster_cen = apartamentos_centro.rename(columns=columnas_renombradas)

# Combinar conjuntos
# por ahora la que mas -> precio, estrellas (categoria)
usadas = ['estrellas','categoria','area', 'precio']
# category to number in Special
datos3_cluster_alb['categoria'] = datos3_cluster_alb['categoria'].astype('category')
datos3_cluster_alb['categoria'] = datos3_cluster_alb['categoria'].cat.codes
datos3_cluster_alb = datos3_cluster_alb[usadas]
```

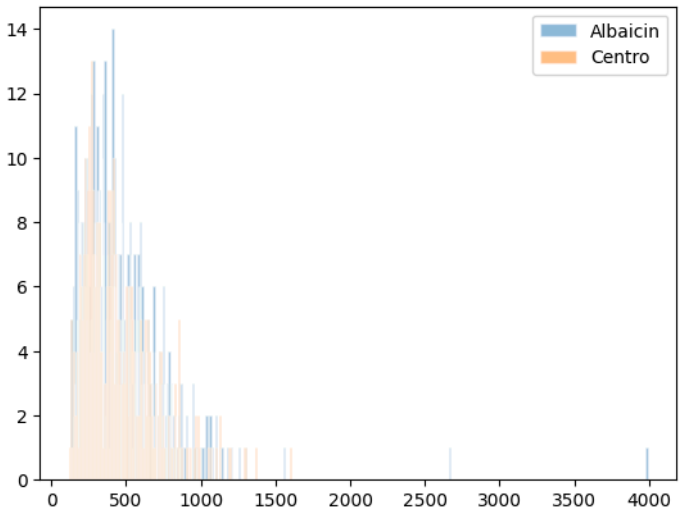
```
datos3_cluster_cen['categoria']                                    =
datos3_cluster_cen['categoria'].astype('category')
datos3_cluster_cen['categoria']                                    =
datos3_cluster_cen['categoria'].cat.codes
datos3_cluster_cen = datos3_cluster_cen[usadas]

# Normalización
X2_normal_alb = datos3_cluster_alb.apply(norm_to_zero_one)
X2_normal_cen = datos3_cluster_cen.apply(norm_to_zero_one)
```

Al segmentar y comparar los resultados para ambos subconjuntos de datos, esperamos descubrir tendencias específicas que puedan orientar tanto a los turistas en su toma de decisiones como a los propietarios en la optimización de su oferta. Por ejemplo, es posible que los apartamentos del Albaicín presenten una mayor concentración de opciones exclusivas y con precios elevados debido a su valor histórico, mientras que en el Centro predominen opciones más asequibles con mayor variabilidad en su calidad.

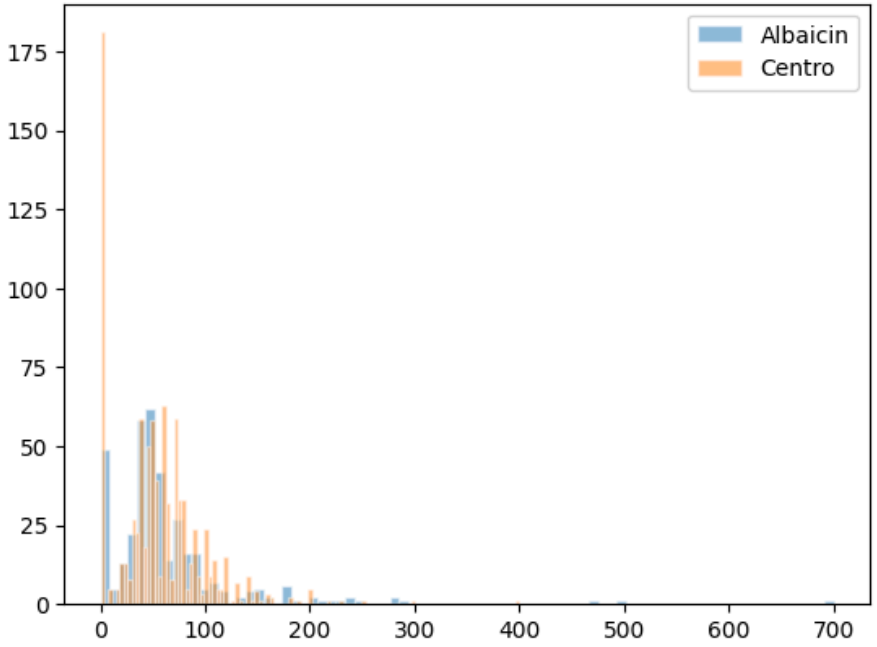
El conjunto de datos para el caso del Albaicín contiene 374 alojamientos, mientras que para el caso del Centro de Granada tenemos 904 alojamientos. Hay que tener en cuenta que la variable categoría es de tipo nominal y tenemos que pasarla a numérica antes de normalizar los datos:

> Categorías de alojamiento nominal: [NaN, 'plus', 'preferente', 'destacado']
> Categorías de alojamiento numérico: [-1 1 2 0]

Observemos qué diferencia hay entre el precio de los apartamentos del Albaicín y los del Centro mediante un histograma:



Observamos que el precio de los apartamentos, tanto en un barrio como en otro, oscila entre 100 y 1300 euros en general. Sin embargo, hay apartamentos, sobre todo del Albaicín que cuestan mucho más, llegando incluso a 4000 euros. Estos casos se podrían considerar casos atípicos. Veamos un histograma que muestre el área de los apartamentos:



Vemos que hay muchos apartamentos en el Centro, aunque también en el Albaicín, que tienen asociado un área de 0 m², lo que sugiere que no se alquila el apartamento entero, sino una única habitación. También vemos que la mayoría de apartamentos tienen como máximo 200m², excepto algunos pocos del Albaicín que llegan a 700m², lo que también podríamos considerar un caso atípico. Por último, veamos la cantidad de estrellas en cada apartamento de cada barrio:

Albaicín: Calidad (en estrellas): [3 4] Cantidad: [234 140]
Centro: Calidad (en estrellas): [0 3 4] Cantidad: [1 505 398]

Para interpretar los datos va a ser muy importante tener en cuenta estos resultados. Vemos que en Albaicín solo hay 2 posibles valores para el número de estrellas (3 y 4), bien representados ambos. Pero en el caso del Centro, solo hay un apartamento con 0 estrellas y, el resto están repartidos entre 3 y 4 estrellas.

Para llevar a cabo la segmentación, se emplearon cinco algoritmos de clustering diferentes, cada uno aportando una visión particular de los datos y permitiendo contrastar los resultados para determinar el método más adecuado en este caso:

- **K-Means**: Es un método rápido y sencillo que requiere predefinir el número de clústeres. Supone que los grupos tienen formas esféricas, lo que puede ser una limitación en datos con distribuciones más complejas. Sin embargo, su eficacia en situaciones bien definidas lo hace muy popular.
- **Agglomerative Clustering**: Este enfoque jerárquico organiza los datos en una estructura de árbol que facilita interpretar las relaciones entre clústeres. Es ideal para conjuntos de tamaño moderado, pero puede volverse lento al trabajar con datos muy grandes.
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: Este algoritmo es capaz de detectar clústeres de densidades variables y manejar eficazmente los puntos

atípicos. No obstante, su desempeño depende de parámetros como el radio de vecindad (*eps*) y el tamaño mínimo del clúster (*min_samples*), que requieren una configuración cuidadosa.

- **BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)**: Resulta especialmente útil para grandes volúmenes de datos, ya que primero crea un resumen jerárquico antes de aplicar la segmentación. Sin embargo, su sensibilidad al parámetro *threshold* puede influir significativamente en los resultados.
- **Gaussian Mixture Models (GMM)**: Este enfoque probabilístico permite identificar clústeres con formas más variadas que K-Means, lo que lo hace más flexible. Es especialmente efectivo cuando los datos presentan superposiciones, aunque puede ser más costoso en términos computacionales.

En las siguientes secciones se analizan los resultados obtenidos al aplicar estos algoritmos para segmentar los apartamentos turísticos del Albaicín y el Centro de Granada, destacando patrones relevantes y características distintivas de cada zona.

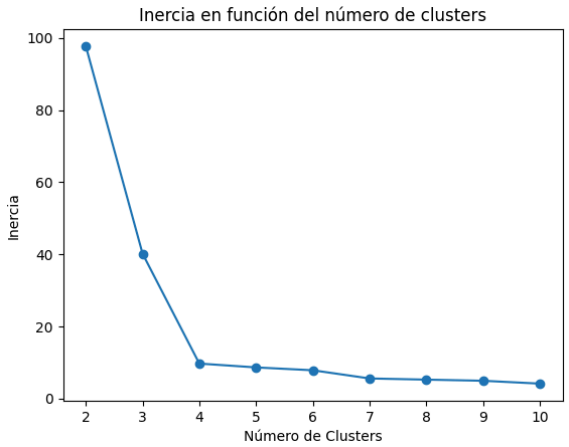
Algoritmo	Subconjunto	Número clusters	Coeficiente Silhouette	Índice Calinski-Harabasz	Tiempo de ejecución (s)
k-Means	Albaicín	4	0.807	2098.33	0.006044387817382812 5
	Centro	3	0.528	2407.825	0.007987737655639648
Agglomerative Clustering	Albaicín	4	0.777	1524.411	0.010474443435668945
	Centro	2	0.681	3136.001	0.021709203720092773
DBSCAN	Albaicín	7	0.728	945.202	0.007416009902954101 6
	Centro	8	0.462	1062.286	0.018457412719726562
BIRCH	Albaicín	3	0.607	402.922	0.015136003494262695
	Centro	2	0.703	3545.145	0.020900249481201172
GMM	Albaicín	4	0.793	1849.339	0.012037038803100586
	Centro	2	0.681	3136.001	0.014931440353393555

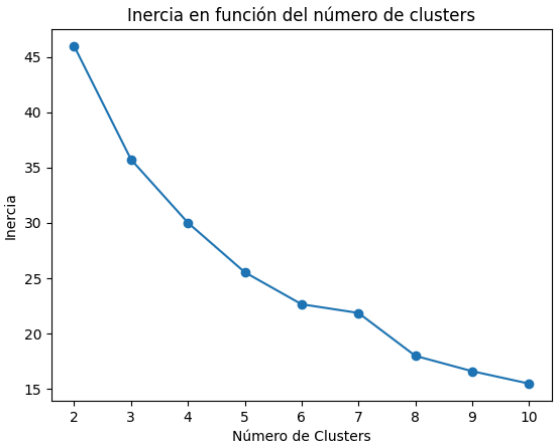
Para el Albaicín, el algoritmo k-Means ofrece el mayor coeficiente Silhouette (0.807), indicando una buena compactación y separación de los clústeres, además de ser el más rápido en ejecución (0.006 s). Sin embargo, el índice Calinski-Harabasz es más alto en Agglomerative Clustering y GMM, lo que

sugiere que estos algoritmos también forman clústeres bien definidos en términos de dispersión interna y separación. En el Centro, Agglomerative Clustering destaca con el mayor índice Calinski-Harabasz (3136.001), indicando clústeres robustos, mientras que BIRCH logra un buen equilibrio entre calidad (coeficiente Silhouette de 0.703) y rapidez (0.0209 s). DBSCAN, aunque eficaz para detectar clústeres más numerosos (7-8 clústeres), presenta coeficientes Silhouette más bajos, lo que refleja mayor superposición o ruido en los datos.

K-Means

Como ya hemos visto en los anteriores casos de estudio, para encontrar el número óptimo de clusters podemos usar el método del codo. Veamos las gráficas de la inercia en función al número de clusters en ambos casos:

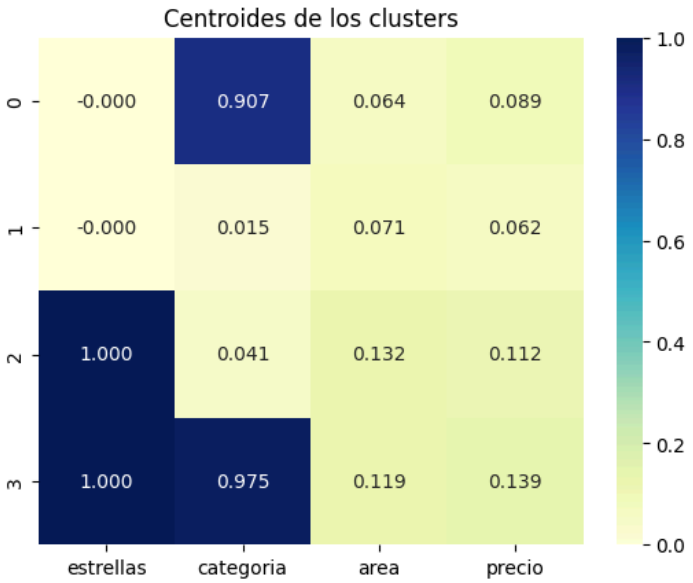




A la izquierda tenemos el gráfico para el conjunto de datos del Albaicín, donde claramente encontramos un codo en 4 clusters, que será el valor que utilicemos en el algoritmo. A la derecha tenemos el gráfico para el conjunto de datos del Centro de Granada, donde no se ve dónde está el codo y, probando varios valores, he concluido con que 3 clusters era un buen valor. Veamos cuáles son los valores de las medidas de rendimiento en ambos casos:

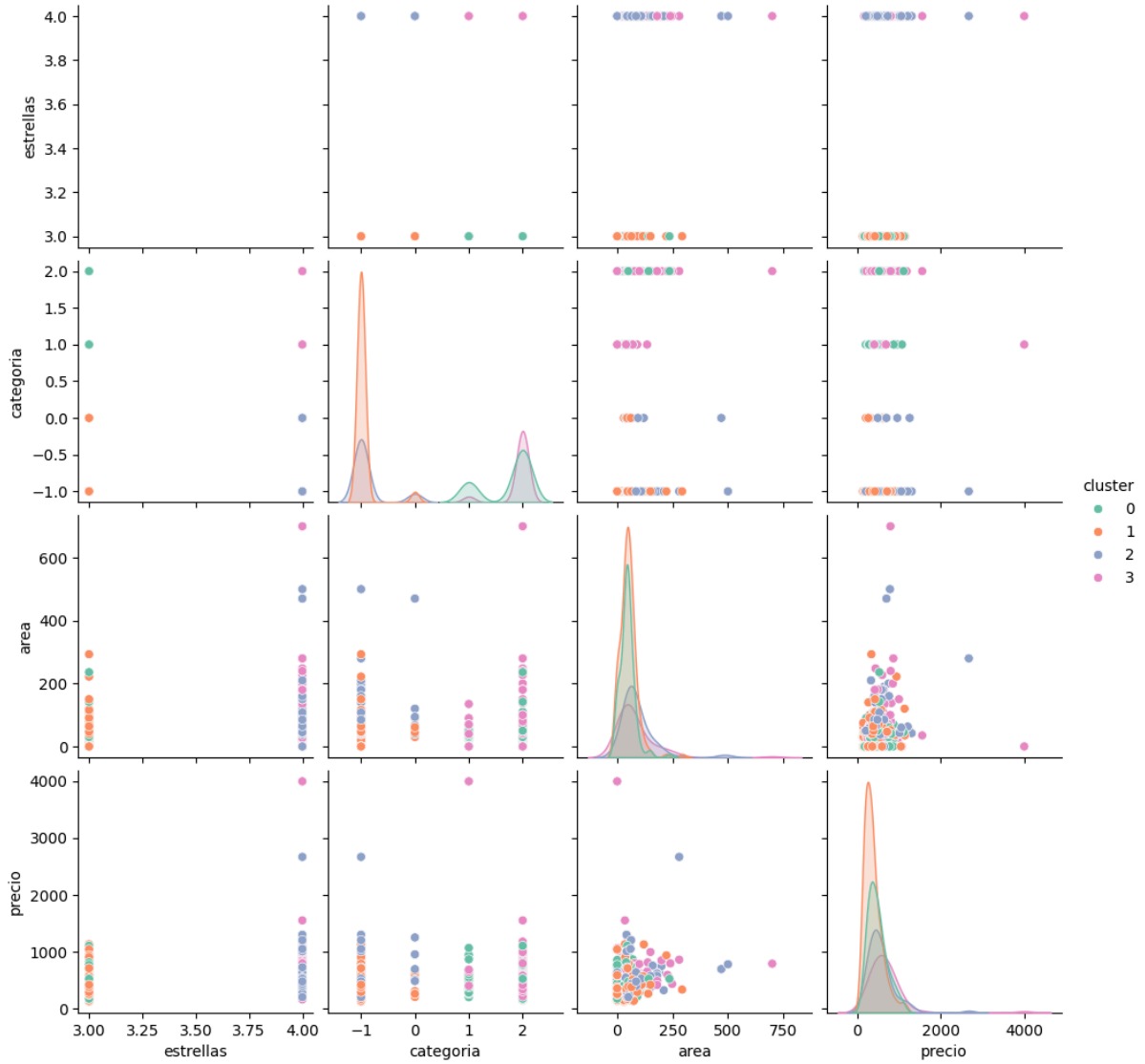
Albaicín:
Calinski-Harabasz Index: 2098.331, Silhouette: 0.807, tiempo ejecución: 0.0060443878173828125
Centro:
Calinski-Harabasz Index: 2407.825, Silhouette: 0.528, tiempo ejecución: 0.007987737655639648

Como era de esperar por lo indicado en el método del codo, el valor de Silhoutte para el caso del Albaicín es considerablemente mejor. Aun así, ambos valores son superiores a 0.5, lo que sugiere que el clustering es apropiado en ambos casos. Si nos centramos en el caso del Albaicín, obtenemos el siguiente heatmap:



El cluster 0 contiene apartamentos de 3 estrellas pero con una categoría asignada, seguramente ‘preferente’ o ‘plus’, tienen área más pequeña y un precio intermedio. El cluster 1 se diferencia del 0 por la categoría, ya que sus apartamentos no tienen ninguna categoría asignada. El cluster 2 contiene apartamentos de 4 estrellas con categoría nula o ‘destacado’, el área de estos apartamentos es la más elevada aunque su precio no es tan elevado como en el cluster 3. El cluster 3 contiene a los apartamentos de 4 estrellas que tienen categoría asignada ‘preferente’ o ‘plus’, el área es más pequeña que el anterior cluster, pero su precio es el más alto entre todos.

Analicemos también el scatter matrix antes de rellenar la tabla de los clusters:

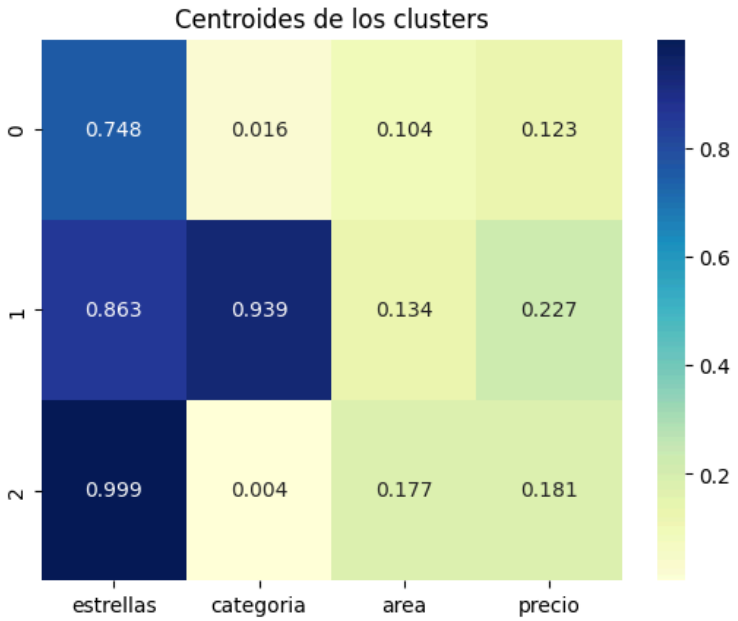


En esta matriz vemos la clara diferencia en estrellas entre los 4 clusters: los dos primeros tiene el número de estrellas más bajo (3) y los dos siguientes tienen el número de estrellas más alto (4). Por otro lado, los clusters 1 y 2 no tienen categorías asignadas o, como mucho, ‘destacado’, mientras que los clusters 0 y 3 tienen las categorías ‘plus’ y ‘preferente’. En cuanto al área, todos rondan el mismo área, aunque los apartamentos más grandes pertenecen al cluster 3 y los medianos al cluster 2. Finalmente, los apartamentos más caros pertenecen al cluster 3 y los intermedios al cluster 2.

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	3	‘plus’/’preferente’	< 250	< 1200
1	3	nula/‘destacado’	< 300	< 1200
2	4	nula/’destacado’	< 500	< 3000
3	4	‘plus’/’preferente’	< 800	< 4000

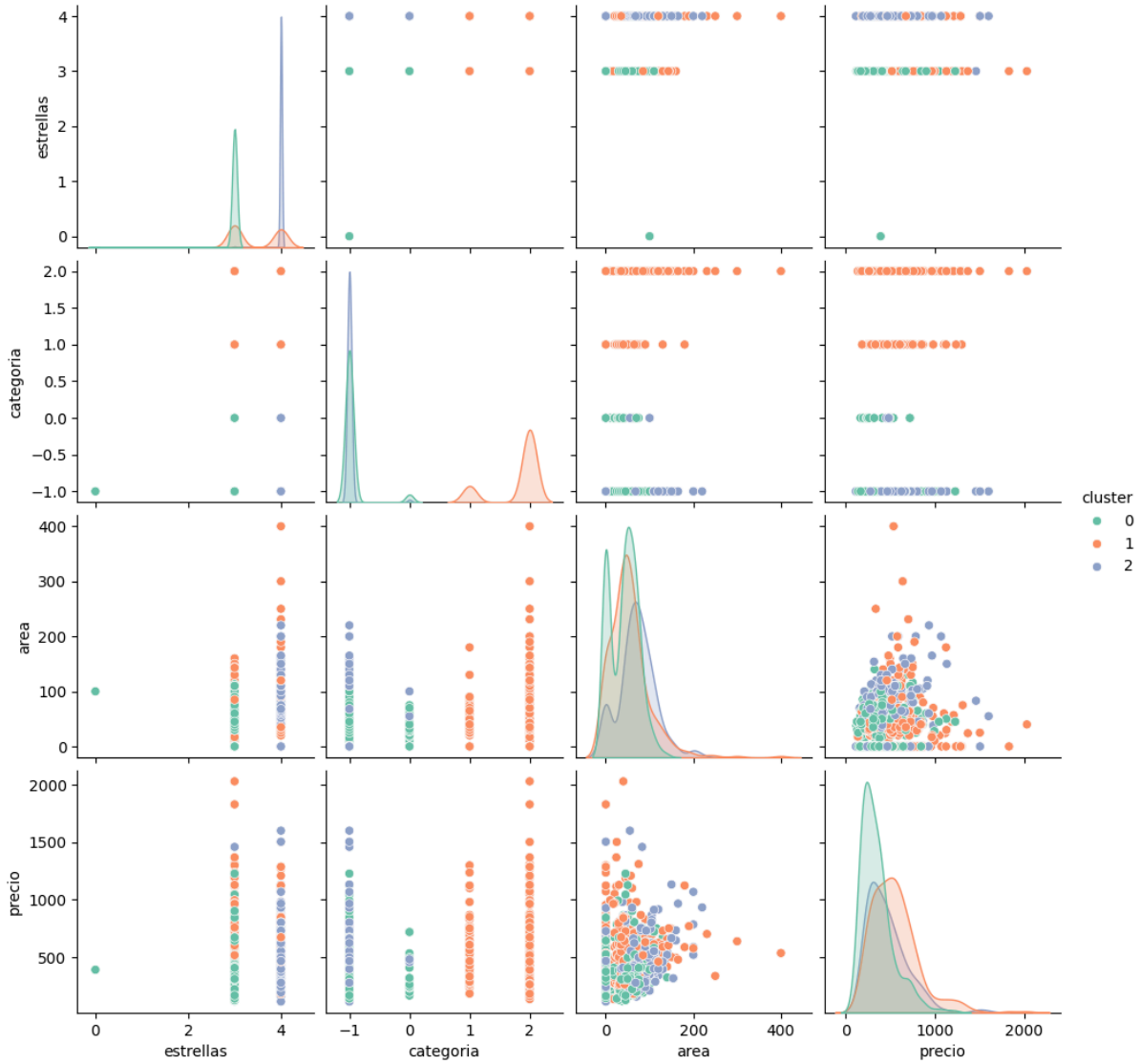
Tenemos que coincidir en que los apartamentos con menos calidad son los que cuestan menos y tienen menos superficie, sin embargo, la categoría no afecta en la calidad de los alojamientos.

Veamos el caso de los apartamentos del Centro de Granada mediante el heatmap:



Tenemos que el cluster 0 contiene a los alojamientos con menos estrellas en calidad y que no tienen categoría asignada, como mucho la de ‘destacado’. Además tienen un área más pequeña y un precio más bajo. Por otro lado, los alojamientos del cluster 2 tienen más estrellas de media y categoría ‘plus’ o ‘preferente’ que coincide con el precio más elevado de todos y un área mediana. Finalmente, el cluster 2 contiene apartamentos con 4 estrellas en general, sin categoría asignada y el área más grande de todas.

Analicemos el scatter matrix en este caso:

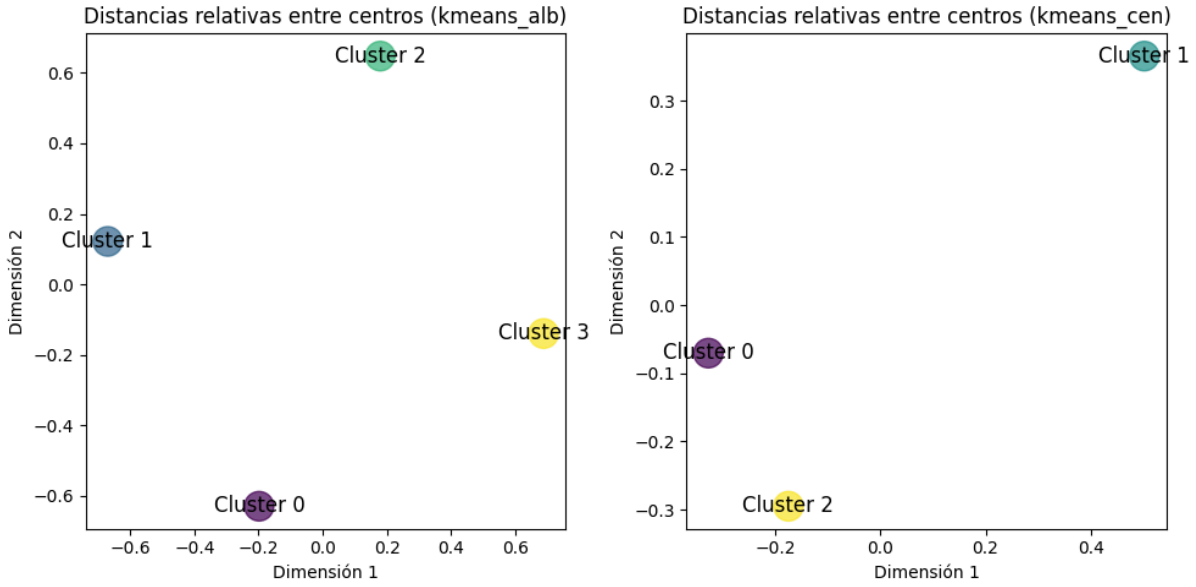


El cluster 0 es el que contiene al apartamento de 0 estrellas, que es único y contiene en su mayoría alojamientos de 3 estrellas. El cluster 1 se caracteriza por contener los apartamentos con categoría asignada, que coincide con los apartamentos que tienen mayor área y mayor precio. El cluster 2 contiene apartamentos de 4 estrellas pero que no tienen categoría. Podemos rellenar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	3	nula/'destacado'	< 100	< 1500
1	3-4	'plus'/'preferente'	< 400	< 2000
2	4	nula/'destacado'	< 200	< 1700

Vemos que, al igual que en el caso anterior, la categoría no afecta a la calidad de las estrellas. Pero sí que afecta el precio y el área, ya que los apartamentos más grandes y más caros pertenecen al cluster 2, que contiene también a los alojamientos de 4 estrellas.

Veamos las distancias relativas entre los centros de los clusters en ambos casos.

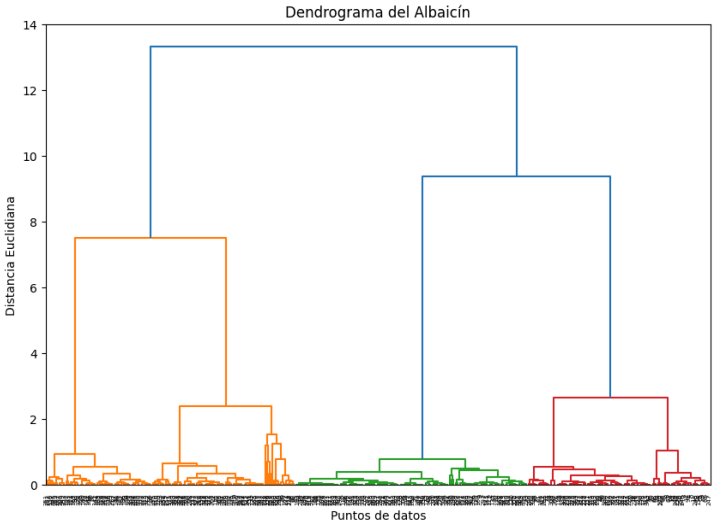


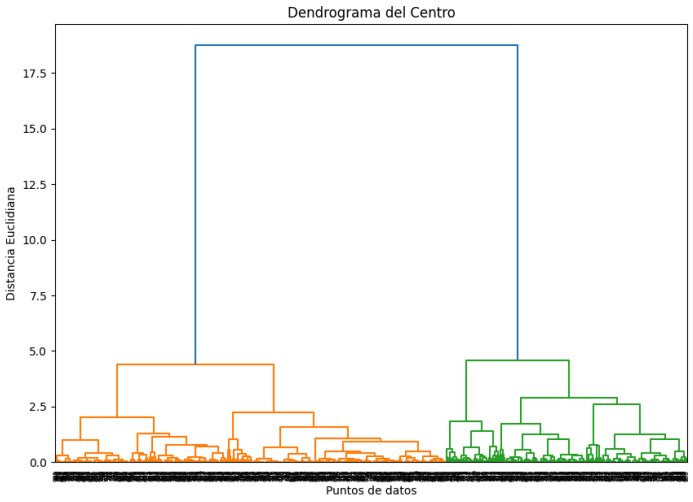
En el gráfico de la izquierda (Albaicín) los centros de clusters están equidistantes entre ellos, lo que refleja que son todos igual de diferentes entre ellos. Esto tiene sentido, puesto que la categoría y la calidad no coinciden en ninguno de ellos por igual.

En el gráfico de la derecha (Centro) los centros de los clusters están moderadamente dispersos, con el cluster 1 ubicado más lejos que los demás, lo que refleja que los apartamentos en este cluster tienen características significativamente diferentes de los otros, probablemente relacionados con su mayor tamaño en área y precios elevados, además de tener categoría asignada. Los cluster 0 y 1 están más cercanos en el espacio multidimensional, lo que indica que comparten similitudes, como los valores de categoría y los precios y áreas más bajos.

Agglomerative Clustering

Como ya sabemos de los anteriores casos de estudio, este tipo de algoritmo es jerárquico y necesita un dendrograma para poder determinar el número de clusters óptimo en cada caso.





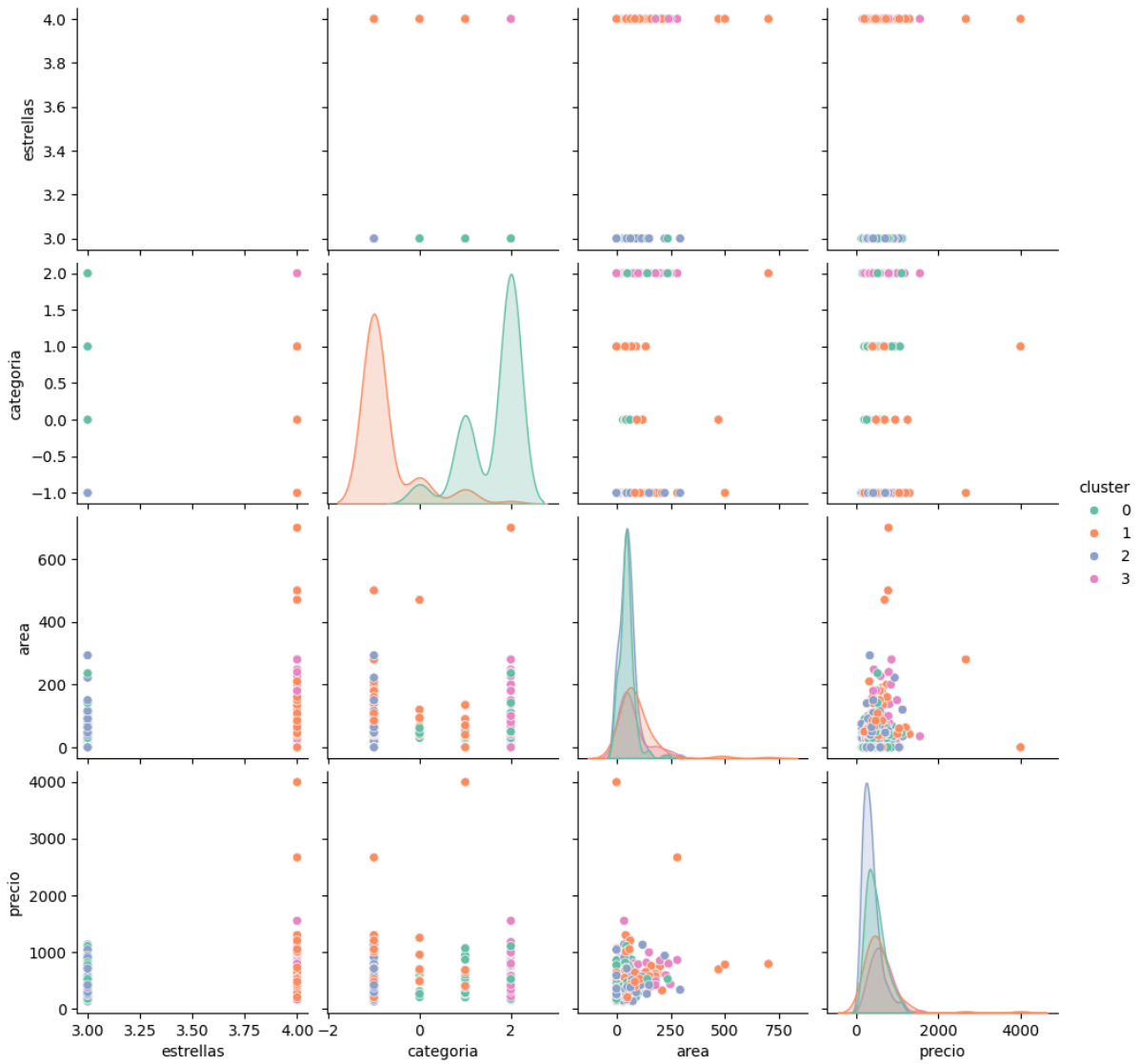
En el caso del Albaicín la diferencia entre las rectas horizontales más grande es entre la primera recta naranja y la primera recta roja, lo que indica que el número de clusters óptimo es 4, igual que en caso anterior. En el caso del Centro de Granada, la diferencia entre rectas más grande indica que podemos usar un valor de 2 clusters.

Los valores de las medidas de rendimiento en cada uno de los casos son:

Albaicín:
Calinski-Harabasz Index: 1524.411, Silhouette: 0.777, tiempo ejecución: 0.010474443435668945
Centro:
Calinski-Harabasz Index: 3136.001, Silhouette: 0.681, tiempo ejecución: 0.021709203720092773

Ambos son clustering buenos, aunque el del Albaicín es ligeramente mejor en cuanto a cohesión (Silhouette) y el del Centro es mejor en cuanto a varianza (Calinski-Harabasz).

Veamos la scatter matrix en el caso del Albaicín.

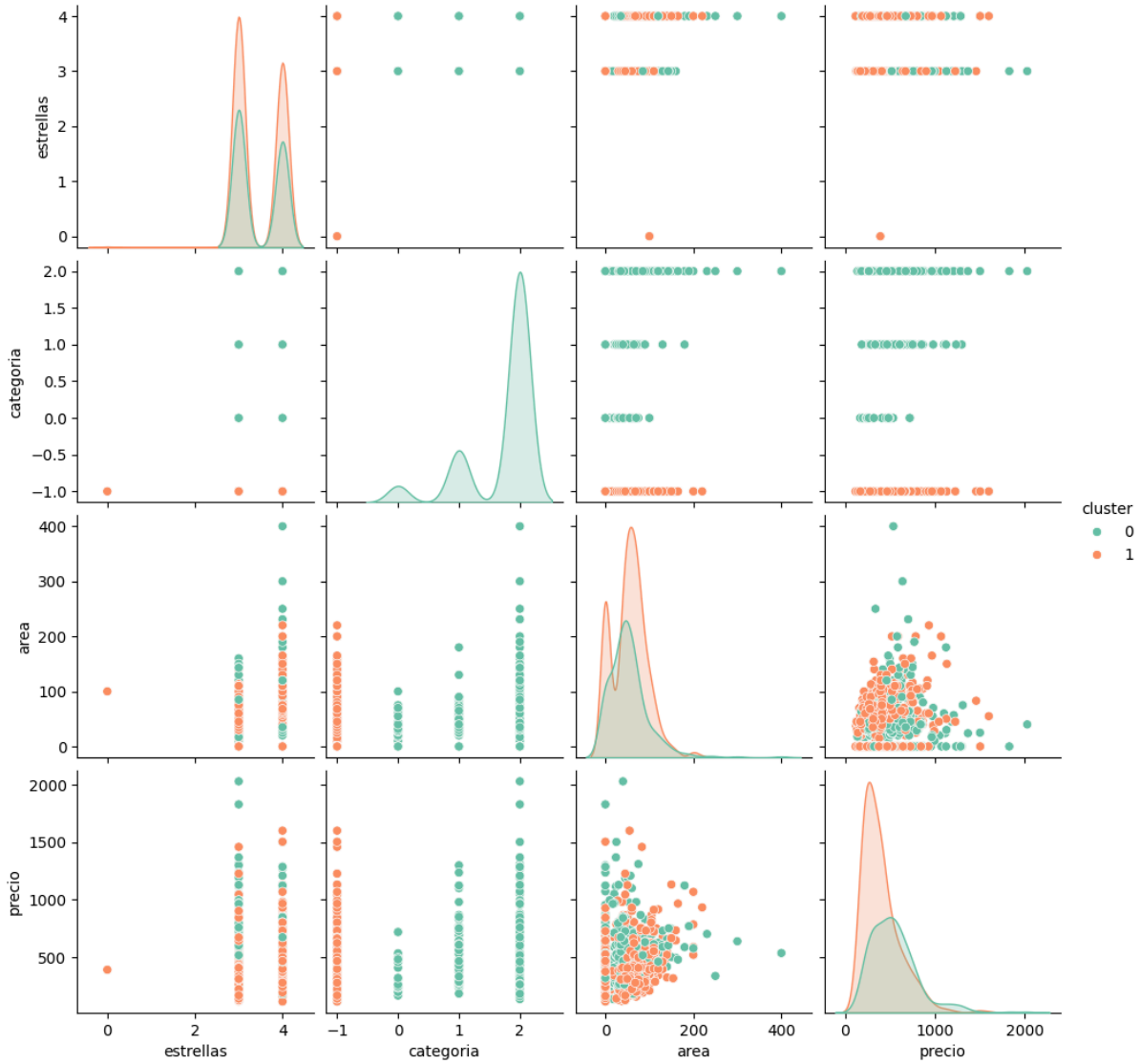


Tenemos que el cluster 0 tiene los apartamentos de 3 estrellas y con alguna categoría asociada, pero el área no es grande y el precio es bajo. El cluster 2 también contiene apartamentos de 3 estrellas, pero no tienen categoría asociada. Por otro lado, el cluster 1 contiene apartamentos de 4 estrellas de cualquier categoría, pero contiene los apartamentos más caros y más grandes, mientras que los apartamentos del cluster 3 tiene categoría 'preferente' pero tienen precio y área cercanos a la media.

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	3	alguna	< 300	< 1000
1	4	cualquiera	< 800	< 4000
2	3	nula	< 300	< 1000
3	4	'preferente'	< 300	< 1500

El cluster más diferenciado de todos es el 1, puesto que contiene los apartamentos más caros y grandes y eso asegura una calidad de 4, independientemente de la categoría. Los clusters 0 y 2 se diferencian por la categoría, pero eso no los diferencia en calidad, lo que no nos resulta de demasiado interés. También existen alojamientos de 4 estrellas que tienen mención 'preferente' y que, además, son económicos y área intermedia, que se encuentran en el cluster 3.

Veamos la clasificación de clusters en el caso del Centro de Granada.



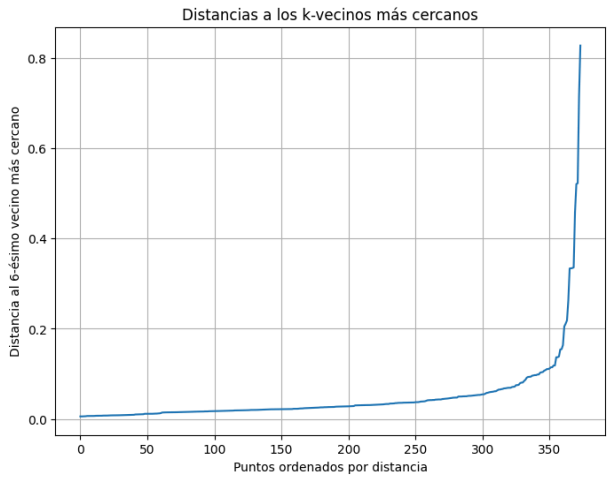
En este caso, solamente tenemos dos clusters diferenciados principalmente por la categoría. El cluster 0 tiene alguna categoría asignada, estos son los que tienen un área más grande y un precio más elevado. Por otro lado, el cluster 1 contiene apartamentos sin ninguna categoría asignada, esto coincide con apartamentos más baratos y más pequeños. En cuanto a calidad, las estrellas están repartidas por ambos clusters de igual forma. Podemos entonces rellenar la siguiente tabla:

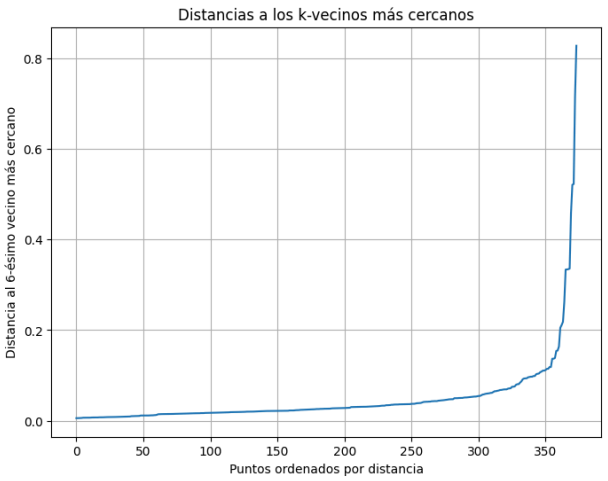
Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	cualquiera	alguna	< 200	< 1500
1	cualquiera	nula	< 400	< 2000

Como habíamos comentado, podemos diferenciar los apartamentos por área y precio, que suelen ir a la par, pero la calidad no marca una diferencia en este caso.

DBSCAN

Este algoritmo, también implementado en el caso de estudio 1, requiere la especificación de dos parámetros ya explicados: épsilon y muestras mínimas. Para las muestras mínimas usaremos dos veces el número de dimensiones, es decir, 2*4=8. Pero para el épsilon usaremos el método de distancia de los k vecinos más cercanos (kNN), lo que nos resulta en estos dos gráficos (uno por cada subconjunto de datos):



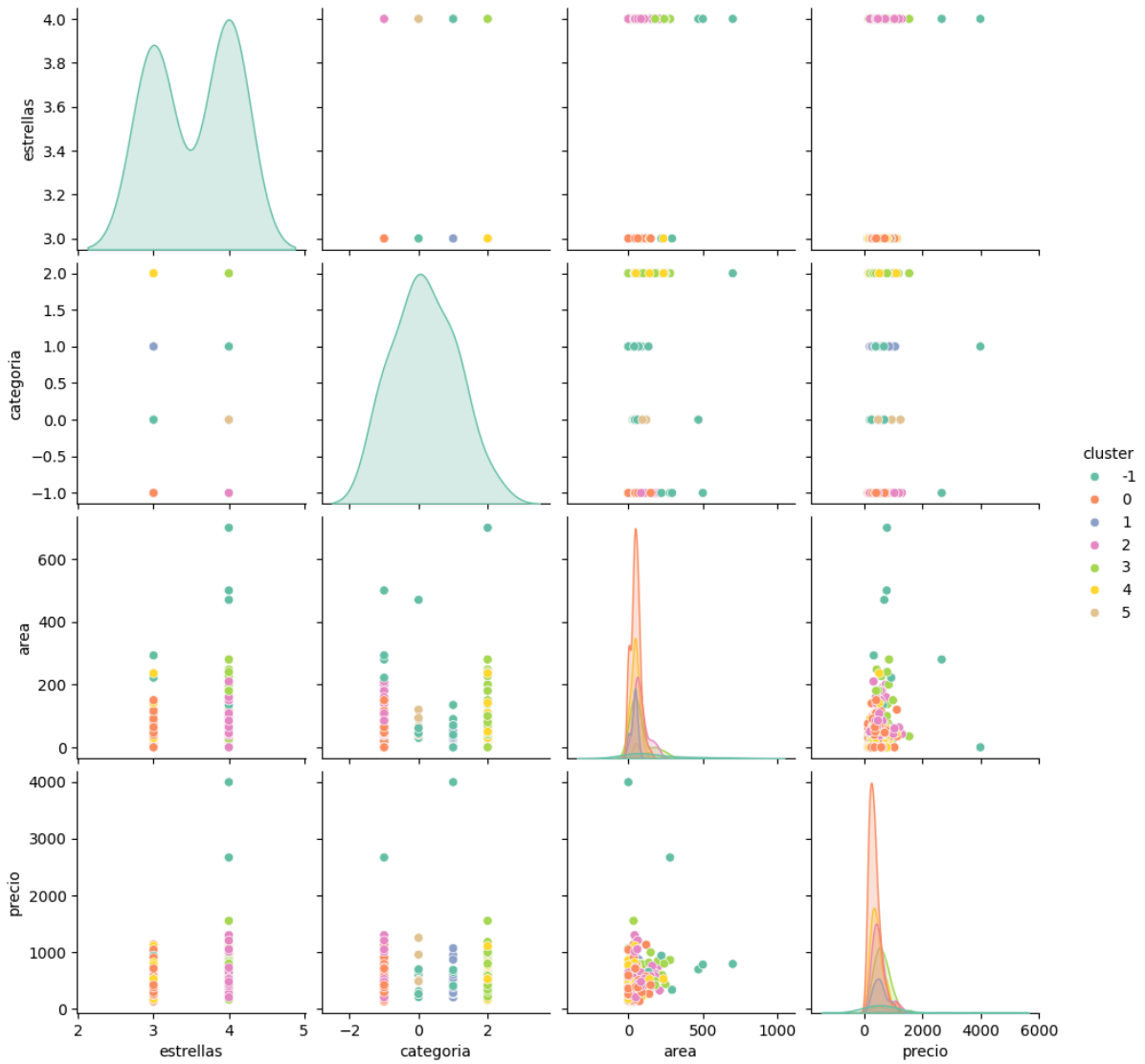


Los valores de las medidas de rendimiento obtenidos al ejecutar el algoritmo para ambos casos son:

Albaicín:
Calinski-Harabasz Index: 945.202, Silhouette: 0.728, tiempo ejecución: 0.0074160099029541016
Centro:
Calinski-Harabasz Index: 1062.286, Silhouette: 0.462, tiempo ejecución: 0.018457412719726562

En este caso sí que hay una gran diferencia entre los resultados proporcionados para los distintos subconjuntos. El clustering del Albaicín tendrá mucha más cohesión, en cambio en del Centro tendrá clusters que se solapen más o con más diferencia de datos dentro de los clusters.

En ambos casos el codo se encuentra entre 0.1 y 0.2, lo que nos lleva a elegir un épsilon de 0.15 en ambos casos. Veamos qué resultados nos da el algoritmo para el subconjunto del Albaicín:

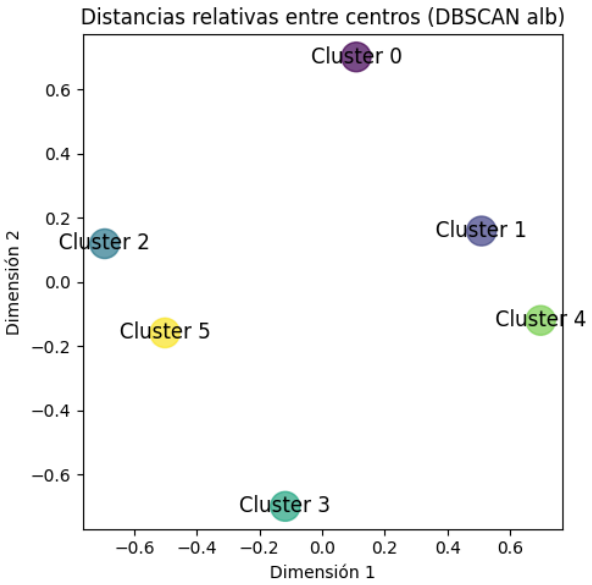


Obtenemos 7 clusters, de los cuales el cluster -1 contiene los outliers, que son los descritos en el epílogo, aquellos que son mucho más caros que el resto y mucho más grandes que el resto. Los otros clusters se diferencian por la cantidad de estrellas y la categoría en la que se encuentran. El cluster 2 contiene apartamentos más pequeños de 4 estrellas y de categoría nula. El cluster 3 tiene apartamentos también de 4 estrellas, pero son un poco más grandes y coinciden con los apartamentos de categoría 'preferente'. El cluster 4 también tiene la categoría 'preferente' pero con una calidad 3 estrellas. De esta forma, podemos rellenar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
-1	cualquiera	cualquiera	< 800	< 4000
0	3	nula	< 300	< 1500
1	3	'plus'	< 300	< 1500
2	4	nula	< 300	< 1500

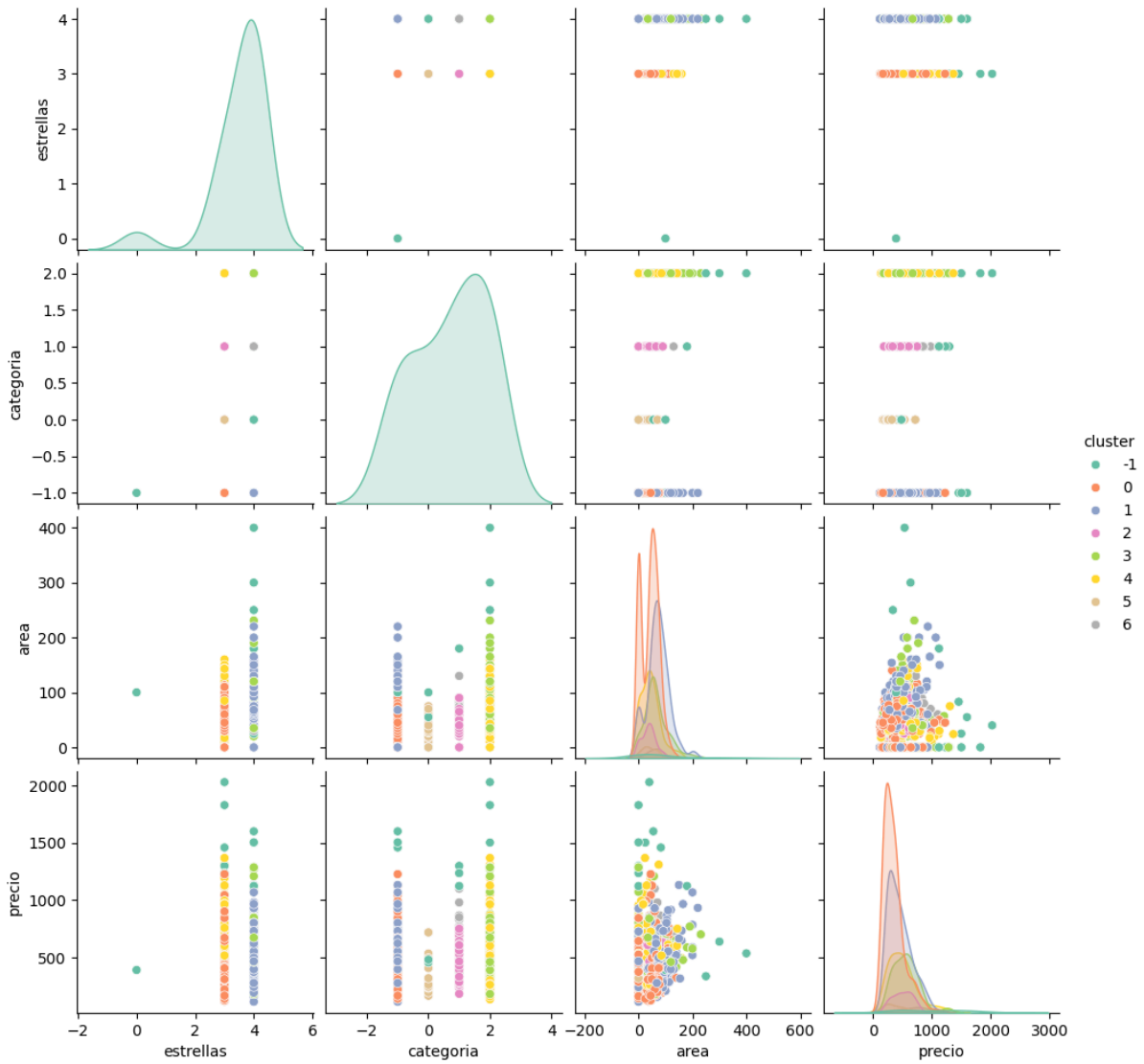
3	4	‘preferente’	< 300	< 1500
4	3	‘preferente’	< 300	< 1500
5	4	‘destacado’	< 300	< 1500

En cuanto a área y precio el algoritmo no hace ninguna distinción más allá de detectar los outliers. Como habíamos comentado, los clusters se distinguen por categoría y calidad. Podemos verlo representado también con las distancias relativas entre los centros:



En el caso del Albaicín, los clusters 0 y 3 son los más alejados, ya que no coinciden ni en categoría ni en calidad. En cambio, los clusters 1 y 4 coinciden en calidad de 3 estrellas y tienen alguna categoría asignada parecida (‘plus’ y ‘preferente’), por eso se encuentran tan cerca. Los clusters 2 y 5 tienen ambos 4 estrellas en calidad y categorías muy cercanas (nula y ‘destacado’).

Veamos el clustering realizado para el subconjunto del Centro de Granada.

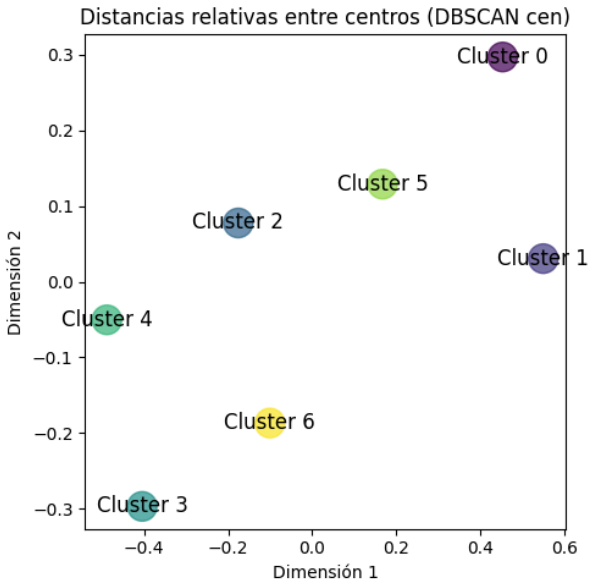


Los clusters creados por el algoritmo en el caso del Centro de Granada son 8, siendo uno de ellos el cluster -1 que contiene los outliers. El resto de clusters, igual que en el caso anterior, se diferencian por estrellas y categoría, aunque también afectan de alguna manera al área y el precio de los alojamientos. Veámoslo en la tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
-1	cualquiera	cualquiera	< 400	< 2000
0	3	nula	< 100	< 1500
1	4	nula	< 250	< 1500
2	3	‘plus’	< 100	< 1500
3	4	‘preferente’	< 250	< 1500
4	3	‘preferente’	< 200	< 1500

5	3	‘destacado’	< 100	< 1500
6	4	‘plus’	< 150	< 1500

Aquí vemos que hay una diferencia de clusters por calidad y categoría, pero que además sí que afecta en algo el área. Tenemos que los clusters con calidad de 3 estrellas son generalmente más pequeños, aunque no se puede ver una diferencia en el precio. Veamos las distancias relativas de lso centros de los clusters en este subconjunto de datos:



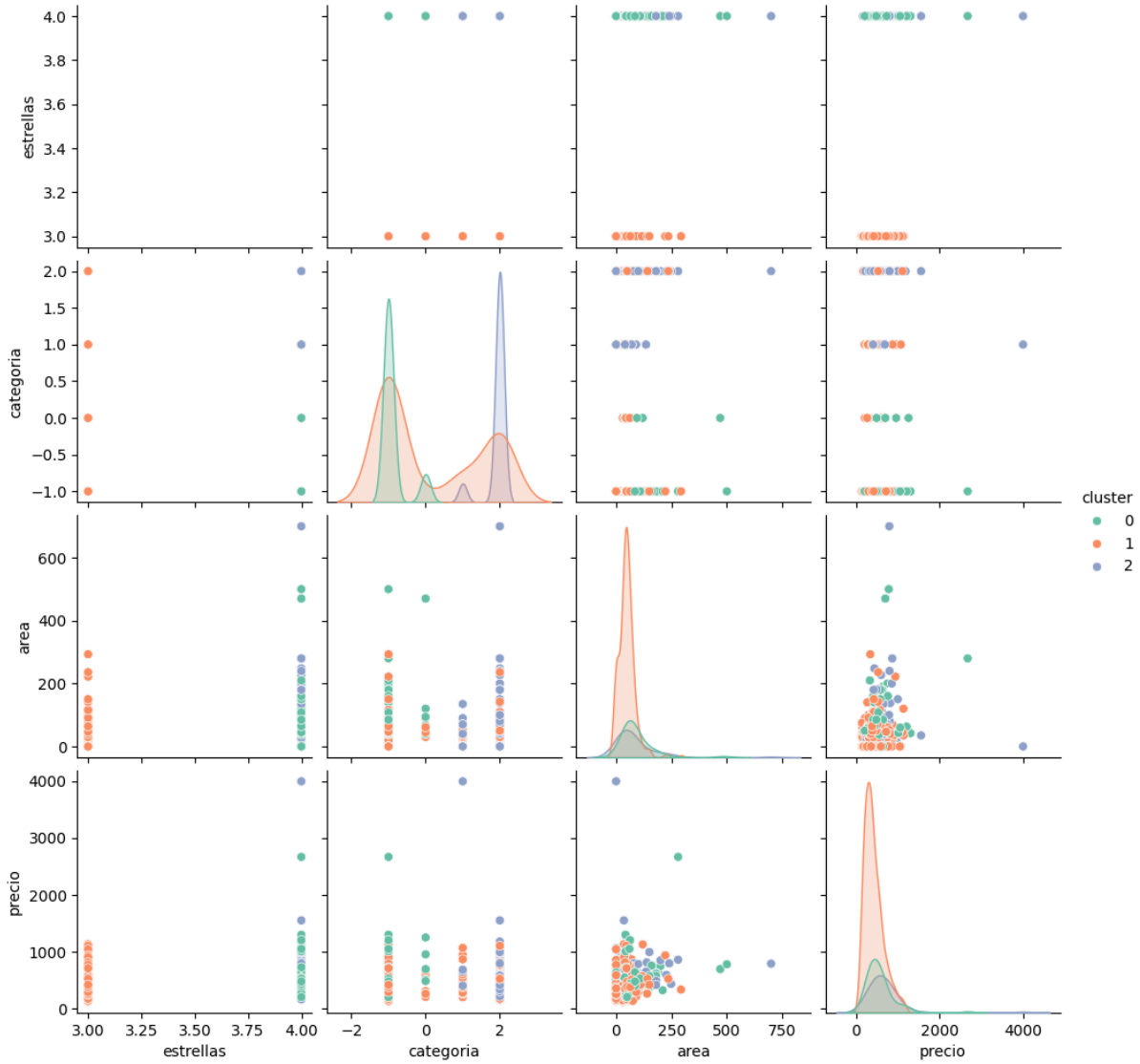
Los clusters 0 y 3 son los más lejanos entre ellos debido a que no coinciden en calidad y, además, tienen las categorías más alejadas (nula y ‘preferente’). Los clusters del medio se distinguen por calidad y categoría, cuanto más alejadas, más lejana tienen la categoría.

BIRCH

Este algoritmo también ha sido implementado en el segundo caso de estudio. Podríamos usar la puntuación de Silhouette para elegir los parámetros de este algoritmo, pero lo que haremos será usar el número de clusters que también hemos usado en Agglomerative Clustering. Con estos valores para el número de clusters obtenemos resultados bastante altos en cuanto a medidas de rendimiento, que son las siguientes:

Albaicín:
Calinski-Harabasz Index: 402.922, Silhouette: 0.607, tiempo ejecución: 0.015136003494262695
Centro:
Calinski-Harabasz Index: 3545.145, Silhouette: 0.703, tiempo ejecución: 0.020900249481201172

A diferencia de en el resto de algoritmos, los resultados del Centro son mejores que los resultados del Albaicín, exceptuando el tiempo de ejecución que es ligeramente superior. Veamos el scatter matrix de los apartamentos del Albaicín:



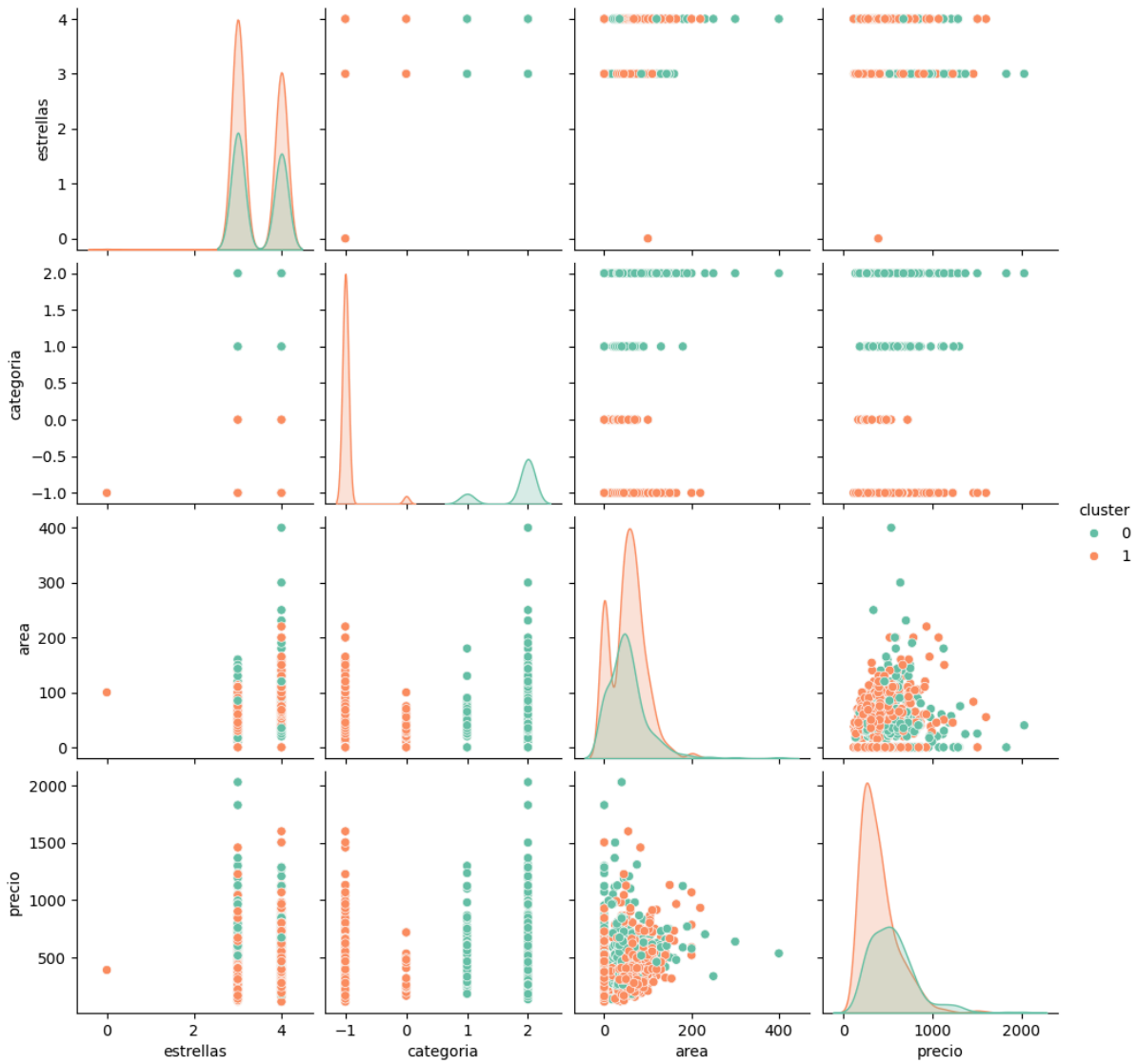
Obtenemos 3 clusters diferenciados por categorías y calidad en estrellas. El cluster 1 contiene todos

los apartamentos de 3 estrellas, independientemente de la categoría, el precio y el área. Por otro lado, los clusters 0 y 2 se diferencian por la categoría: el cluster 0 tiene categorías ‘destacado’ y nula, mientras que el cluster 2 tiene categorías ‘preferente’ y ‘plus’. Rellenamos la tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	4	nula/‘destacado’	< 600	< 3000
1	3	cualquiera	< 300	< 1100
2	4	‘plus’/ ’preferente’	< 800	< 40000

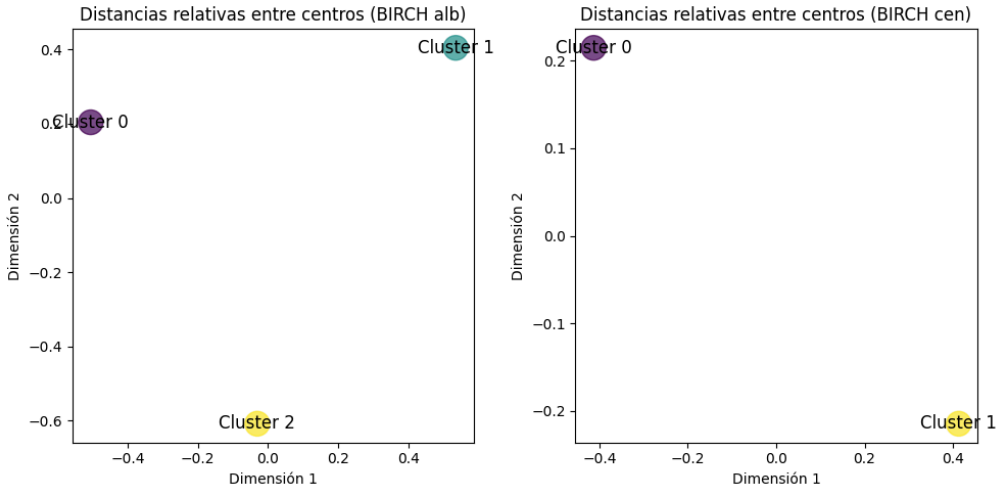
Vemos que hay una gran diferencia entre el cluster 1 y los otros dos clusters en todas las características.

Veamos los diferentes clusters en el caso del Centro de Granada usando un scatter matrix:



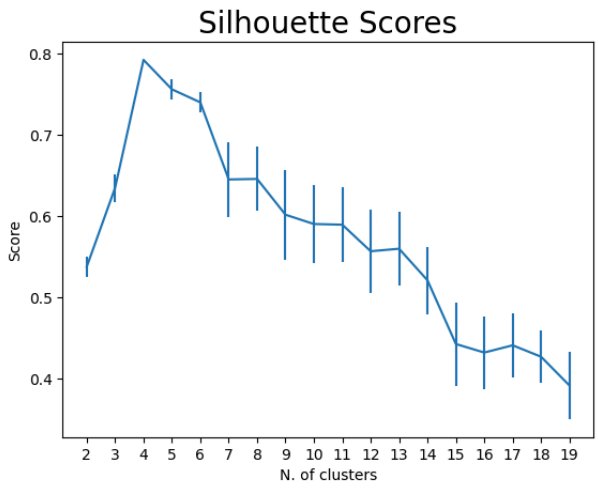
El cluster 0 contiene apartamentos con categorías de ‘plus’ y ‘preferente’, que suponen aquellos apartamentos con mayor precio y un poco menos de área. El cluster 1 tiene los apartamentos de categoría nula o ‘destacado’, pero estos puede que sean más grandes en área y con un precio más bajo.

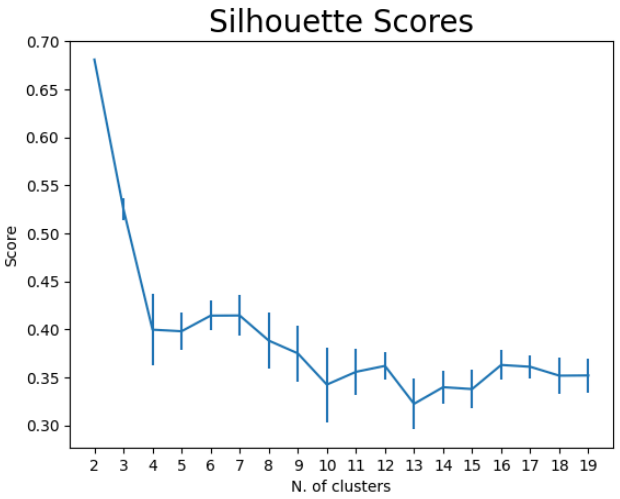
Las distancias entre los centros de los clusters son las siguientes:



GMM

Este algoritmo ha sido también usado en los otros casos de estudio y, como en dichos casos, podemos elegir el valor del número de clusters usando el coeficiente de Silhouette. tenemos las siguientes gráficas:

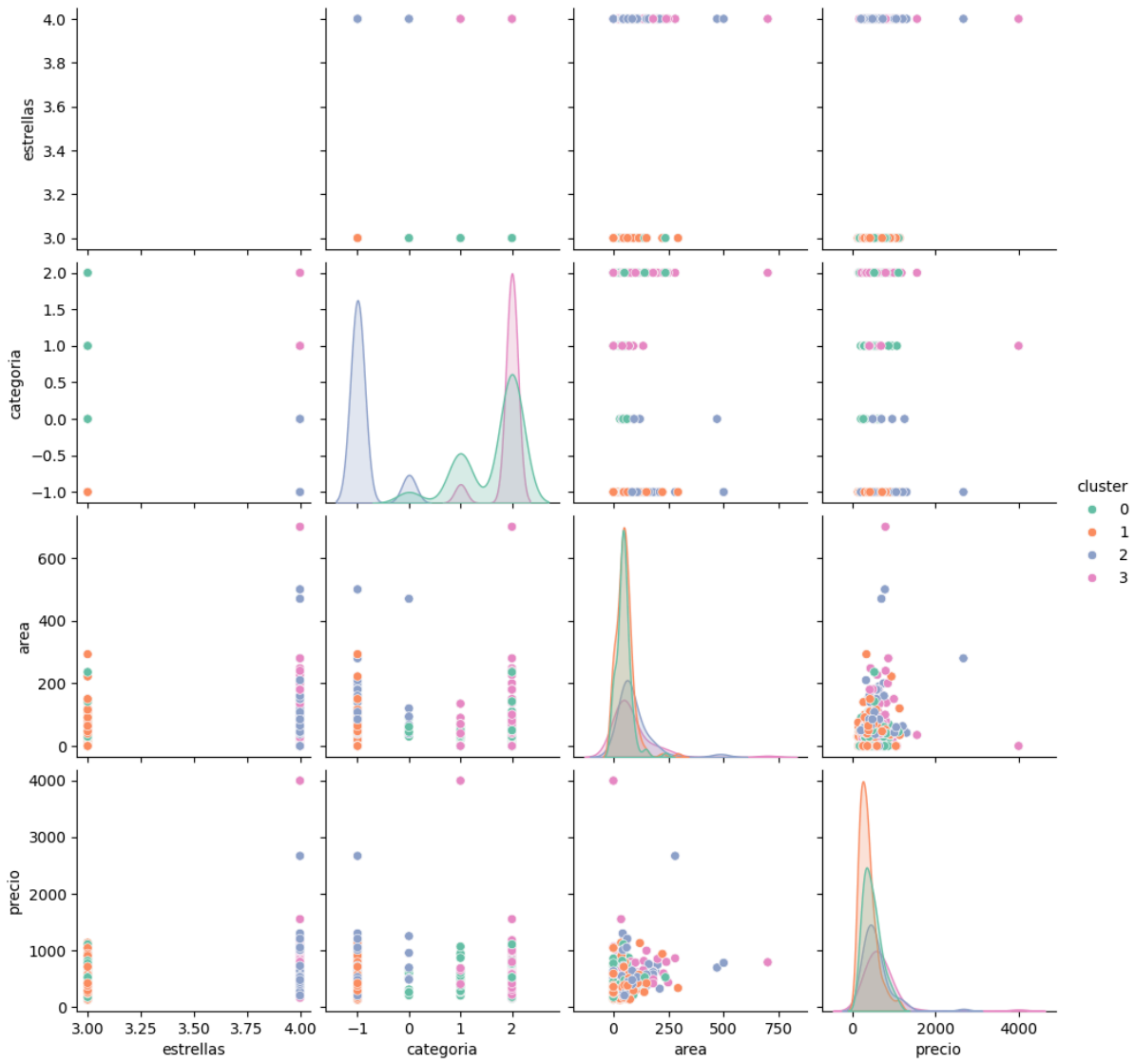




En el caso del Albaicín vemos un número de clusters claro, son 4 clusters lo que se recomienda en otros algoritmos y de igual forma en este algoritmo. En el caso del centro obtenemos el mejor coeficiente con 2 clusters, como también se ha hecho en los otros algoritmos. Veamos las medidas de rendimiento conseguidas por cada subconjunto:

Albaicín:
Calinski-Harabasz Index: 1849.339, Silhouette: 0.793, tiempo ejecución: 0.012037038803100586
Centro:
Calinski-Harabasz Index: 3136.001, Silhouette: 0.681, tiempo ejecución: 0.014931440353393555

Analicemos los distintos clusters en cada caso. Empezamos por el caso de los apartamentos del Albaicín:

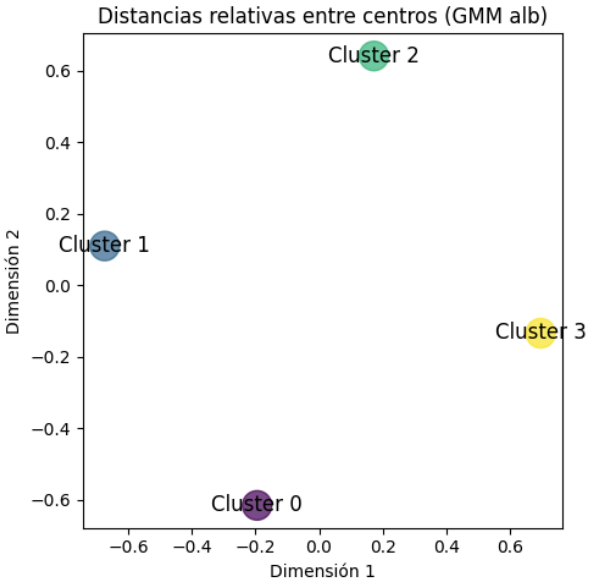


El cluster 0 contiene apartamentos de 3 estrellas que tienen alguna categoría asignada. Si la categoría es ‘destacado’ tienen un área pequeña, pero si la categoría es ‘preferente’ el área es más grande. El cluster 1 contiene apartamentos de 3 estrellas pero con categoría nula, pero no hay un dato relevante en cuanto a área y precio. Los clusters 2 y 3 tienen 4 estrellas en calidad, pero se distinguen por categoría: el cluster 2 tiene categorías bajas y el cluster 3 tiene categorías altas. Podemos rellenar la siguiente tabla:

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	3	alguna	< 300	< 1500
1	3	nula	< 300	< 1500
2	4	nula/ ‘destacado’	< 600	< 3000
3	4	‘plus’/ ’preferente’	< 800	< 4000

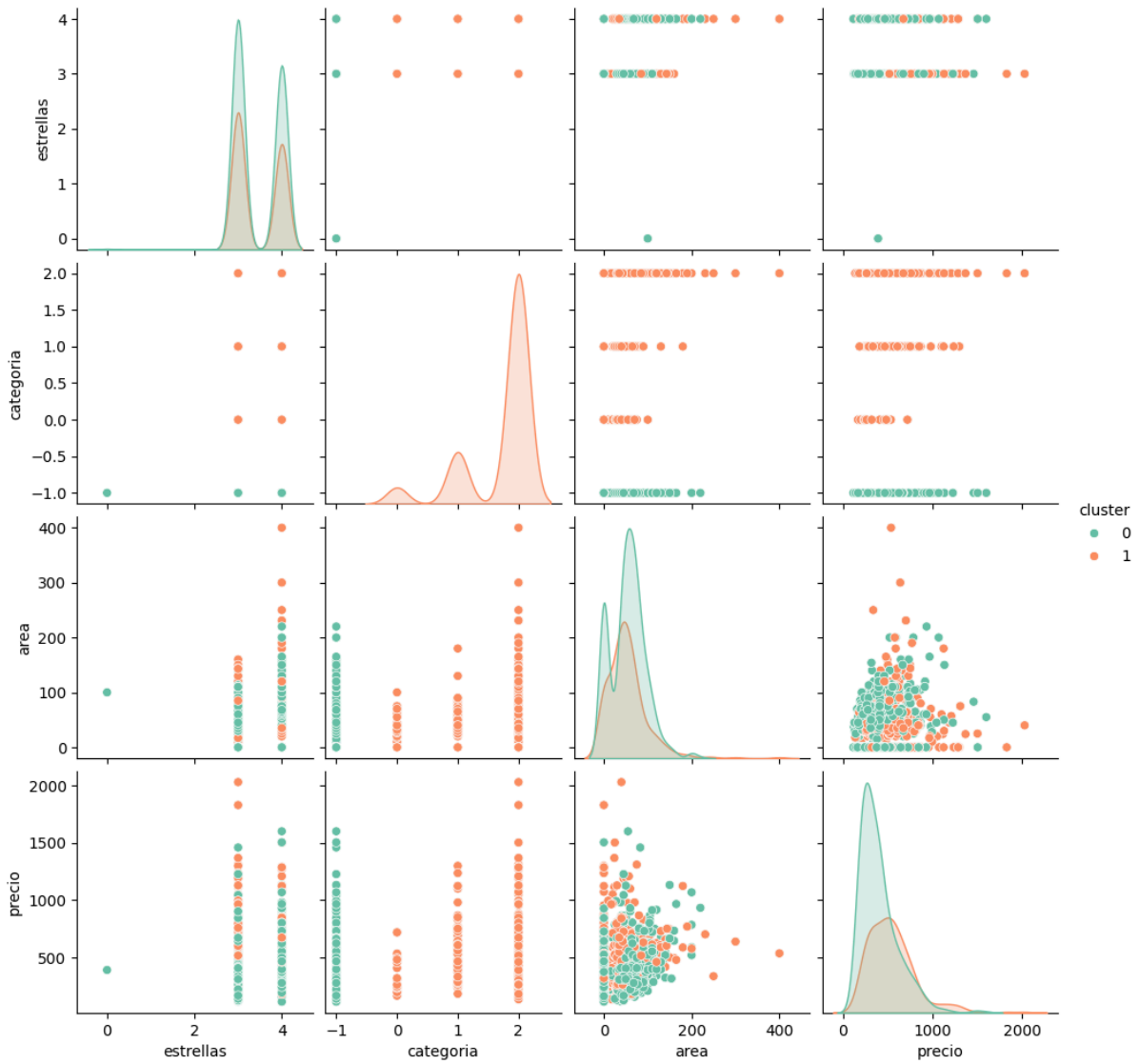
Tenemos que el área y la calidad sí que están relacionadas, puesto que los apartamentos grandes coinciden con los de mayor calidad y de la misma forma ocurre con el precio. Un precio más alto asegura una calidad mayor. La categoría no influye en ningún aspecto.

Esto se puede ver claramente en la distancia relativa de los centros de los clusters:



Los clusters son equidistantes porque se distinguen los unos de los otros por el número de estrellas o si la categoría es baja o alta, lo que los hace bien distinguidos.

Veamos el caso del Centro de Granada:



Aquí sólo tenemos 2 clusters bien diferenciados por categorías. El cluster 0 contiene los apartamentos con categoría nula, que no influye de manera clara en el área del alojamiento o en su precio. Por otro lado, el cluster 1 contiene apartamentos con alguna categoría asignada, que además son aquellos que tienen un área más grande y un precio más elevado.

Cluster	Calidad (en estrellas)	Categoría (mención)	Área (en m²)	Precio (en euros)
0	cualquiera	nula	< 250	< 1500
1	3-4	alguna	< 400	< 2000

Interpretación de la segmentación

El análisis de segmentación permite identificar patrones significativos entre los apartamentos turísticos del Albaicín y el Centro de Granada, considerando las variables de calidad, categoría, área y precio. En el Albaicín, los apartamentos tienden a diferenciarse en mayor medida por su calidad y categoría, con segmentos específicos como apartamentos de 4 estrellas con categorías altas asociados a precios más elevados y áreas reducidas. Por otro lado, en el Centro, la segmentación se centra más en la categoría como elemento distintivo, donde los apartamentos con categoría “plus” o “preferente” tienden a tener mayor área y precio. Los algoritmos utilizados, como k-Means y Agglomerative Clustering, muestran una separación más clara en el Albaicín, como lo indica su coeficiente Silhouette superior, mientras que en el Centro, la variabilidad dentro de los clústeres es mayor, reflejando una oferta más heterogénea. El uso de DBSCAN y BIRCH revela la presencia de outliers y agrupamientos más finos en el Albaicín, mientras que en el Centro, estos algoritmos confirman una distinción más marcada por categoría y área. En resumen, el Albaicín parece tener una oferta más segmentada según calidad y categoría, mientras que el Centro refleja una relación más directa entre categoría, área y precio.

Bibliografía

Bedre, Renesh. “DBSCAN clustering algorithm in Python (with example dataset).” *DBSCAN clustering algorithm in Python (with example dataset)*, 2 June 2024, https://www.reneshbedre.com/blog/dbscan-python.html. Accessed 3 December 2024.

Hugo. “How to Choose Optimal Hyperparameters for DBSCAN.” *stataiml*, https://stataiml.com/posts/how_to_set_dbscan_paramter/.

“Implementing Agglomerative Clustering using Sklearn.” *geeksforgeeks*, https://www.geeksforgeeks.org/implementing-agglomerative-clustering-using-sklearn/.

Jain, Sandeep. “Affinity Propagation.” *GeeksforGeeks*, 22 May 2024, https://www.geeksforgeeks.org/affinity-propagation/. Accessed 6 December 2024.

Jain, Sandeep. “ML | OPTICS Clustering Explanation.” *GeeksforGeeks*, 2 May 2023, https://www.geeksforgeeks.org/ml-optics-clustering-explanation/. Accessed 6 December 2024.

Lavorini, Vincenzo. “Gaussian Mixture Model clustering: how to select the number of components (clusters).” *Medium*, https://towardsdatascience.com/gaussian-mixture-model-clusterization-how-to-select-the-number-of-components-clusters-553bef45f6e4.

Mahendru, Khyati. “How to Determine the Optimal K for K-Means?” *Medium*, https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb.

McInnes, Leland. “Comparing Python Clustering Algorithms.” *hdbscan*, https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html.

Triceratops. “How to get the optimal number of clusters using hierarchical cluster analysis automatically in python?” *stackoverflow*, https://stackoverflow.com/questions/50695226/how-to-get-the-optimal-number-of-clusters-using-hierarchical-cluster-analysis-au.