Práctica Aprendizaje Automático I

Proyecto sobre aprendizaje no supervisado

Ignacio Franco y Carlos Menéndez

Índice

Índice 2
Tarea 1: Descripción del Dataset 3
Accidentes por meses 5
Accidentes por distrito 6
Accidentes por momento del día 6
Accidentes según condiciones meteorológicas 7
Lesividad del accidente según condiciones meteorológicas 7
Lesividad del accidente según tipo de persona 10
Lesividad del accidente según tipo de vehículo 11
Tarea 2.1 14
Tarea 2.2 19
Tarea 2.3 20
Tarea 2.4 23
Tarea 2.5 23
Tarea 2.6 24
Tarea 3.1 28
Tarea 3.2 28
Dendograma 28
Elbow Method 29
GAP 30
Conclusión 30
Tarea 3.3 31
Dunn 31
Davies-Bouldin 31
Calinski-Harabasz 32
Silhouette 33
Tarea 3.4 33
Tarea 3.4 37
Tarea 4.1 38
Tarea 4.2 38
Determinar eps 38
Determinar min_samples 39
Tarea 4.3 40
Tarea 4.4 42
Tarea 4.5 42
Tarea 5.1 44
Comparación con otros algoritmos 46

Tarea 1: Descripción del Dataset

Para el desarrollo de la práctica, contamos con unos datos sobre los accidentes de tráfico de la Comunidad de Madrid en 2024 hasta el 11 de octubre de ese mismo año. Contamos con 19 columnas de información las cuales vamos a tratar para poder utilizarlas en nuestros modelos de aprendizaje no supervisado.

Durante el preproceso, vamos a priorizar el uso de one hot encoding. Es el método que mejores resultados nos va a dar para algoritmos de clustering ya que se calculan mejor las distancias.

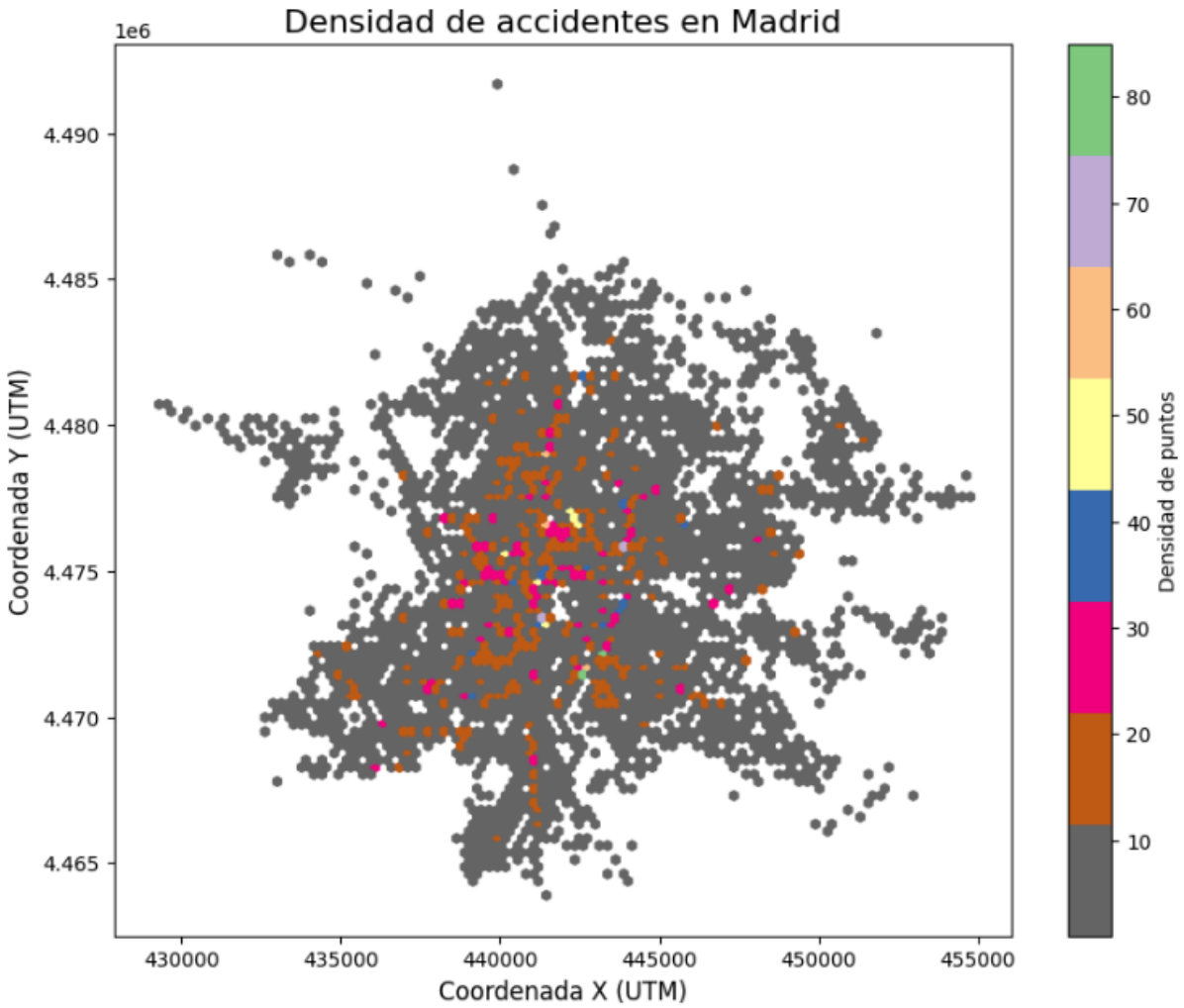
El dataset está formado por las siguientes columnas:

- num_expediente: Está formado por el año del accidente y el identificador. Vemos cómo puede aparecer el mismo número varias veces, esto se debe a que cada persona involucrada en un accidente tiene una fila distinta del dataset.
- Fecha: Fecha del accidente en formato dd/mm/aaaa. Tendremos que pasarla a datetime seguramente.
- hora: La hora exacta del accidente.
- localización: aquí tenemos la ubicación del accidente, puede ser 1 calle o dos (cuando son dos, se refiere a un cruce o intersección).
- Número: el número de la calle para tener más detalle sobre el accidente
- cod_distrito: el código del distrito, cada distrito tiene uno distinto.
- distrito: aquí tenemos el nombre del distrito, para entrenar modelos lo haremos con su respectivo código.
- tipo_accidente: en esta columna se clasifican los accidentes en función de cómo se hayan producido.
- estado_meteorológico: tenemos la descripción meteorológica en el momento del accidente, vemos como tenemos bastantes valores Nan, así que más adelante veremos qué hacemos con ellos
- tipo_vehiculo: aquí obtenemos el tipo de vehículo implicado en el accidente con un string.
- tipo_persona: el tipo de persona implicada en el accidente. También está representada como string así que tendremos que tratarla
- rango_edad:
- sexo: Es de tipo object pero solo hay dos valores distintos así que pasarlo a numérico no debería ser un problema.
- cod_lesividad: la codificación de la columna lesividad. Vemos que tiene muchos valores faltantes y son los mismos que en la columna lesividad, por lo que después observaremos qué es lo que está ocurriendo ahí.
- Lesividad: Strings que clasifican la gravedad de un accidente en función de las lesiones de un pasajero.
- coordenada_x_utm: columna numérica con las coordenadas x de los lugares donde se han producido accidentes.
- coordenada_y_utm: columna numérica con las coordenadas y de los lugares donde se han producido accidentes.

- positiva_alcohol: columna categórica que indica si la persona implicada en el accidente ha dado positivo en alcohol o no. Esta columna seguramente tengamos que convertirla en booleana en un futuro.
- positiva_droga: columna numérica que indica si la persona dio positivo en alcohol o no. Hay tantos valores nulos porque representan a los que no dieron positivo por lo que el tratamiento de datos faltantes en esta columna lo podremos completar con un simple fillna().

Procesamos el dataframe columna por columna priorizando utilizar One hot encoding en todo momento (como ya hemos mencionado anteriormente) y orientando el preproceso a sacar relaciones entre nuestros datos.

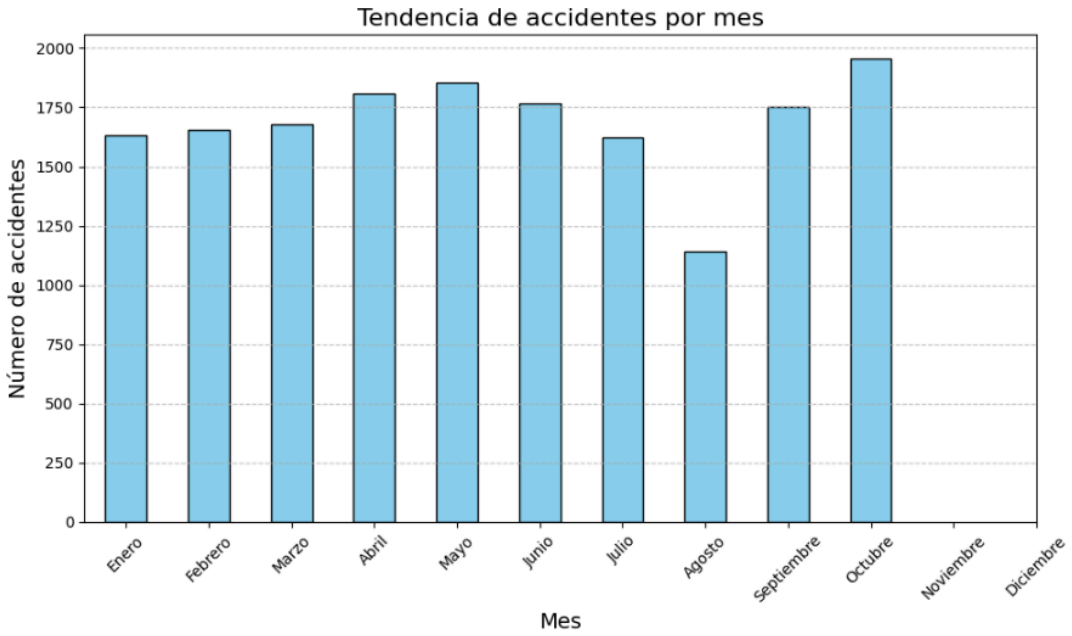
Además, hemos agrupado el nuestros datos



Hemos creado este mapa, donde cada punto está representado por las coordenadas de un accidente para ver un poco por donde están mayormente distribuidos. Como es normal y asumible, por el centro se acumulan más accidentes, sin embargo, podemos observar cómo se concentran más accidentes por la zona sur del centro que por la zona norte.

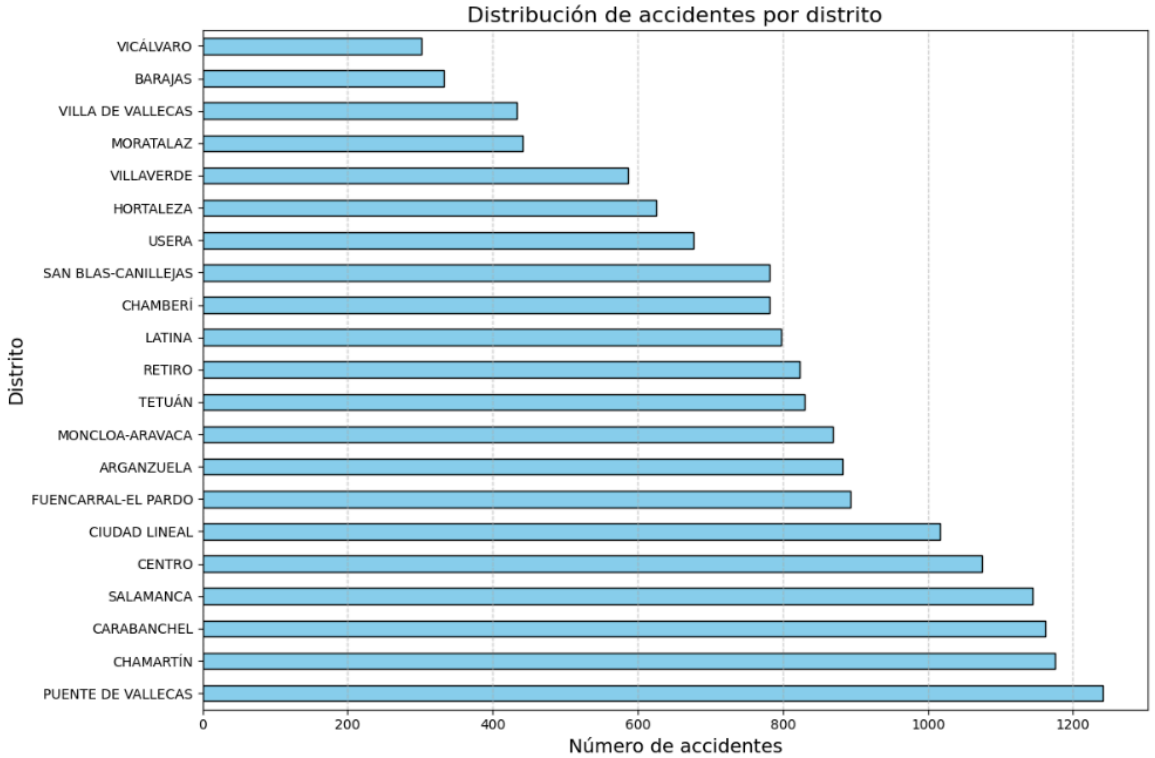
Encontramos un par de puntos verdes en el sur (los puntos verdes representan una gran concentración de accidentes) que a priori parecen ser los lugares donde más accidentes se concentran.

Accidentes por meses



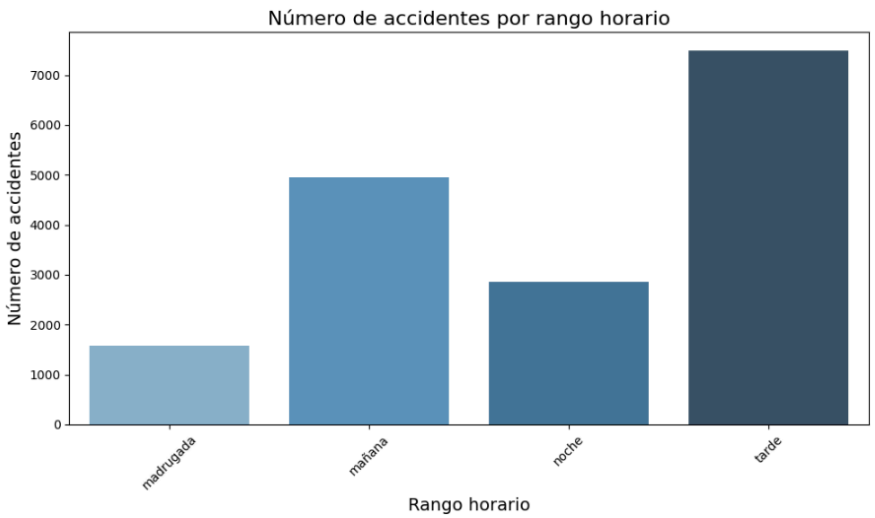
Estudiamos también la tendencia de accidentes por meses. Vemos como en agosto se reducen drásticamente. Esto se debe a que mucha gente de Madrid se va de vacaciones y abandona la comunidad. Por eso en septiembre, cuando la gente vuelve de vacaciones, se produce un aumento en los accidentes volviendo a la tendencia de meses anteriores.

Accidentes por distrito



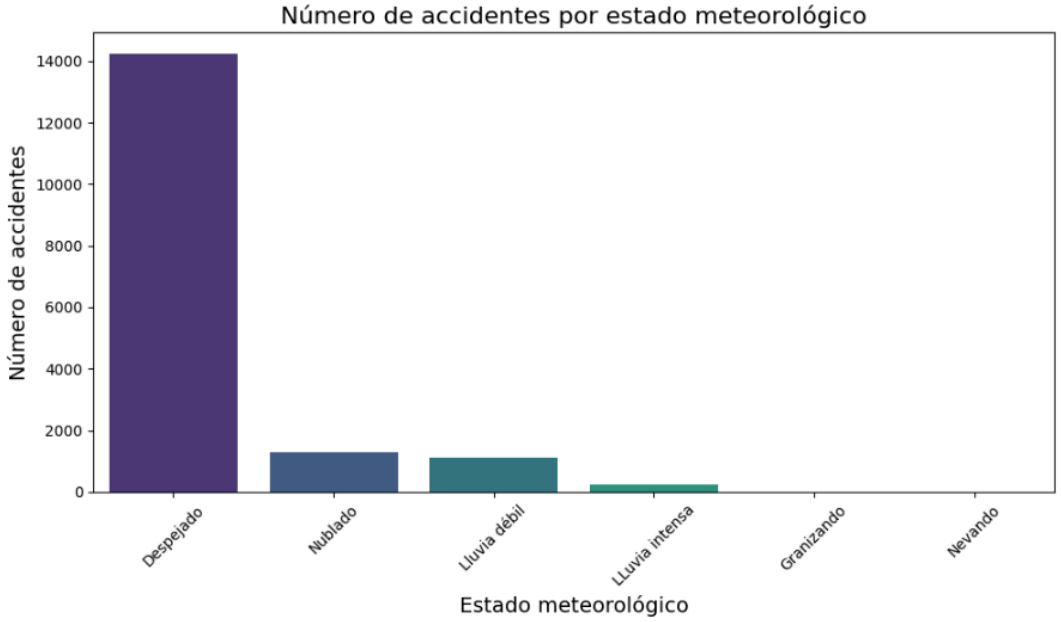
Al analizar cómo se han repartido los accidentes por distrito, vemos que Puente de Vallecas encabeza la clasificación (probablemente sea donde hemos visto los dos puntos verdes en el mapa de Madrid). Seguido puente de Vallecas encontramos varios distritos más céntricos y quizá tirando un pelín al norte como (Chamartín, Salamanca o el centro).

Accidentes por momento del día



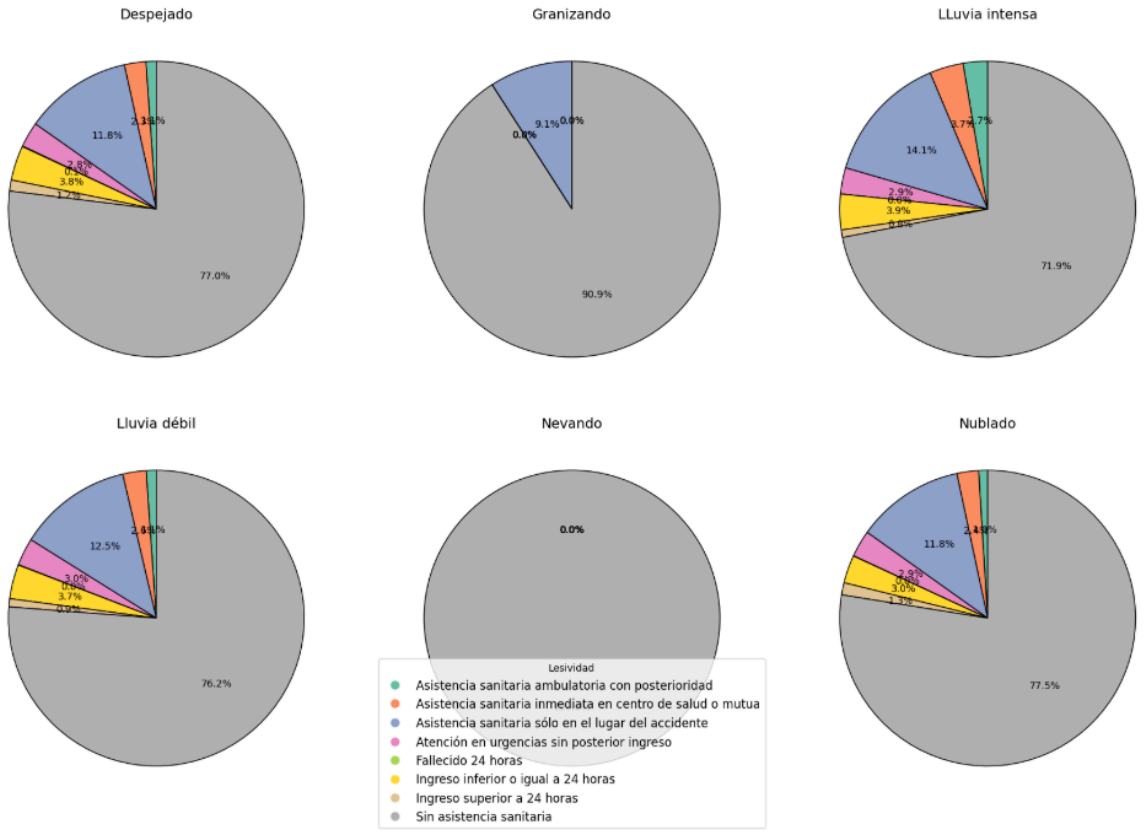
Con esta gráfica podemos observar cómo ha afectado el momento del día en los accidentes de 2024. Como es de esperar, la mayoría han sido por la tarde y por la mañana, ya que son las franjas horarias con mayor tráfico.

Accidentes según condiciones meteorológicas



En este gráfico vemos que los accidentes suelen producirse con el cielo despejado, cuando en Madrid gran parte del año hace un tiempo soleado lo cual no nos aporta gran información. Para ello vamos a relacionar la lesividad del accidente con el estado meteorológico para ver realmente cómo afectan las condiciones climáticas a los accidentes.

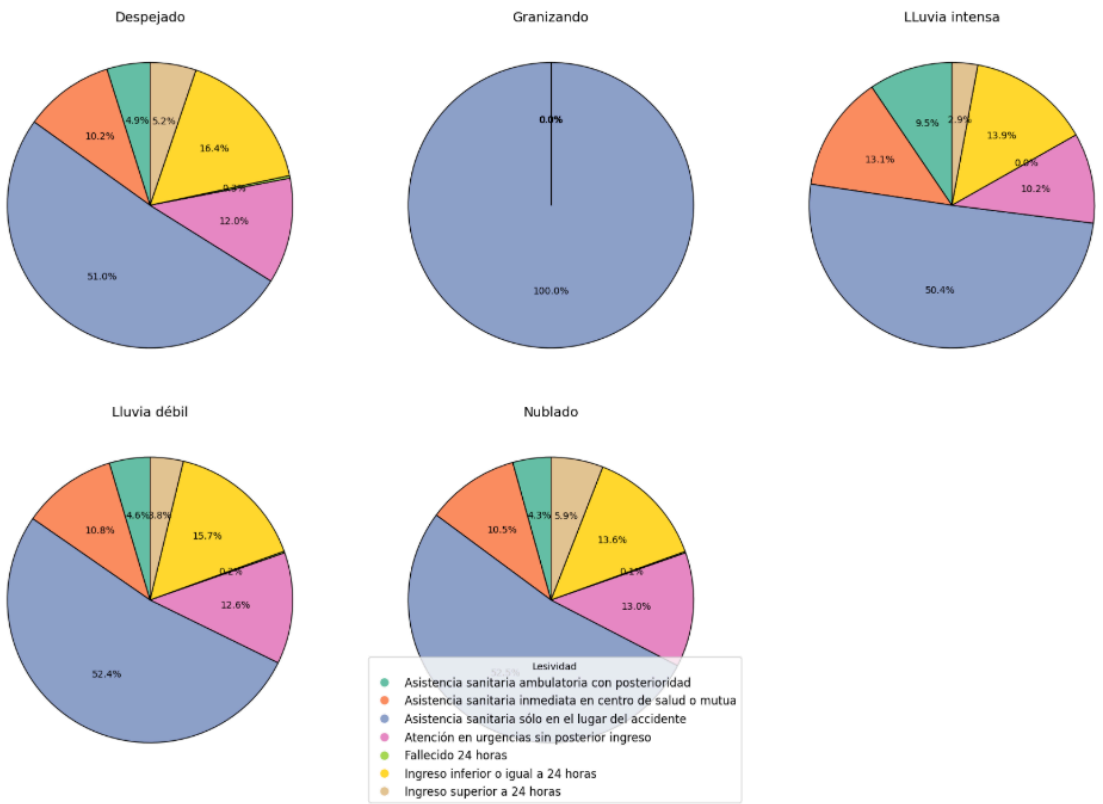
Lesividad del accidente según condiciones meteorológicas



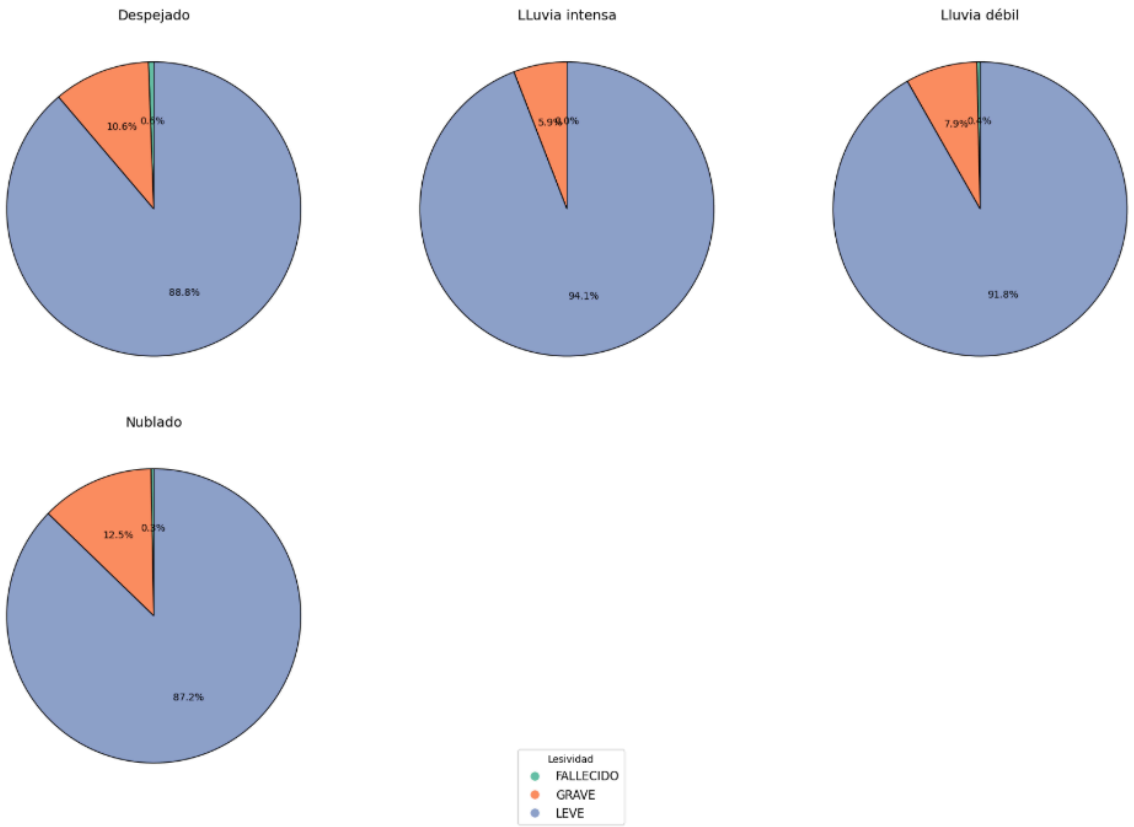
Vemos cómo cuando hay lluvia intensa hay más tendencia a que una persona implicada en el accidente necesite asistencia sanitaria (alrededor de un 28%) que cuando el cielo está despejado(alrededor de un 77%). Respecto a los datos cuando nieva o cuando está granizando no podemos dar muchas explicaciones ya que no tenemos suficientes ejemplos. Esto se puede deber a dos cosas:

- Que en la Comunidad de Madrid nieva y graniza muy poco y por eso faltan datos.
- Que cuando las condiciones meteorológicas son peores, los conductores son más prudentes y prestan más atención a la conducción.

A continuación, vamos a eliminar los datos donde no hay asistencia sanitaria para ver mejor la gravedad de los accidentes cuando ha sido necesario asistencia médica.

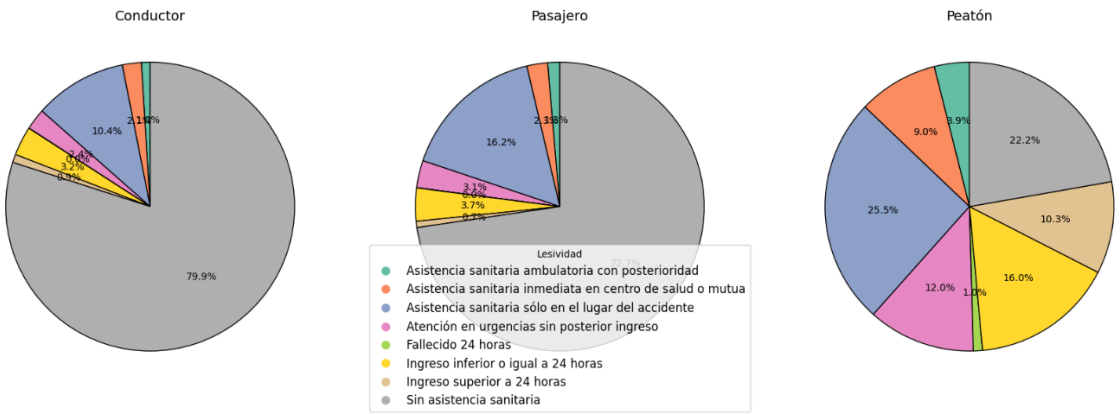


Vemos como en la mayoría de los casos la asistencia sanitaria se realiza en el lugar del accidente sin necesidad de acudir a ningún hospital o centro de salud cercano. Vamos a mapear la función como indica el pdf de las instrucciones de los datos, asignando las variables LEVE, GRAVE y FALLECIDO a cada grado de lesividad. Para observar mejor las proporciones, vamos a eliminar las filas donde la asistencia sanitaria es requerida sólo en el lugar del accidente.



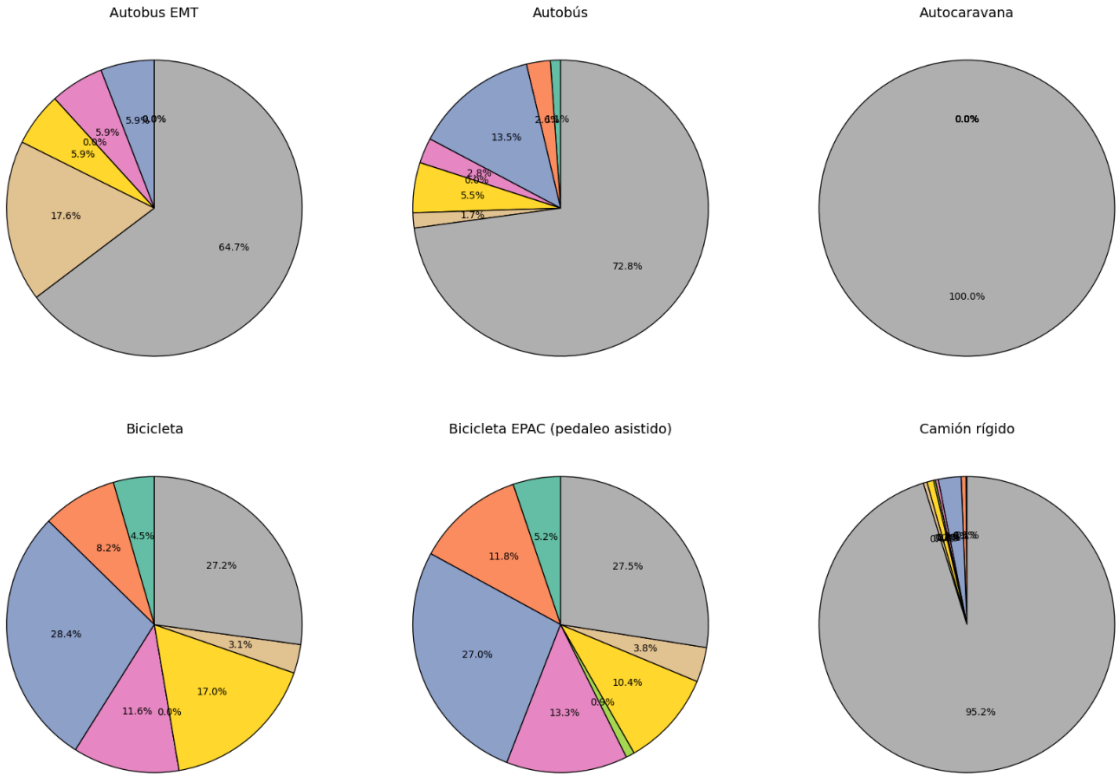
Aquí podemos observar cómo las personas implicadas en los accidentes tienen mayor tendencia de sufrir consecuencias graves cuando hay niebla. Sin embargo, cuando el cielo está despejado hay mayor riesgo de que el accidente sea grave que con lluvia intensa y lluvia débil.
Esto se debe seguramente a que cuando el tiempo es bueno, los conductores van más confiados y circulan a una velocidad mayor lo que hace que aumente el riesgo de accidentes más graves.
De hecho, cuando la lluvia es intensa, ha habido menos accidentes graves que cuando la lluvia es débil ya que con la lluvia intensa hay más prudencia al volante.

Lesividad del accidente según tipo de persona

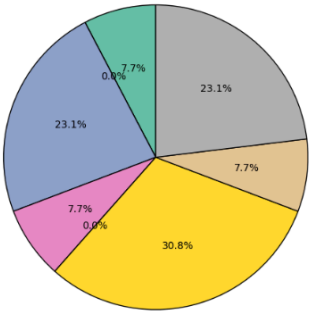


En este gráfico podemos ver cómo los peatones son los más perjudicados cuando ocurre algún accidente. El peatón es el elemento de la vía más vulnerable en caso de accidente y tan sólo en el 22,2% de los casos no ha sido necesario la atención sanitaria.

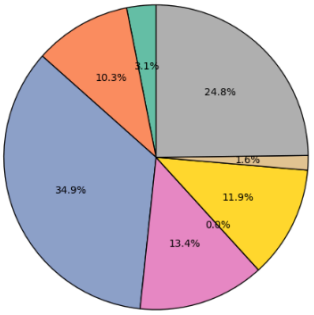
Lesividad del accidente según tipo de vehículo



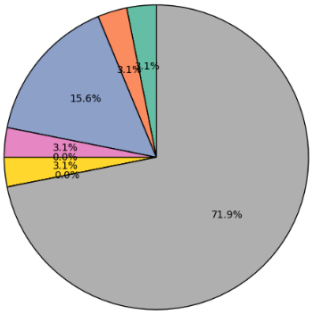
Ciclo



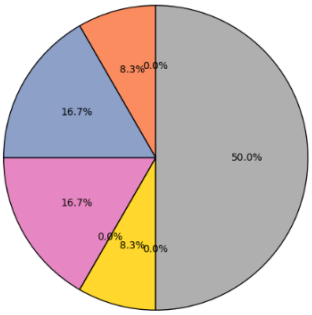
Ciclomotor



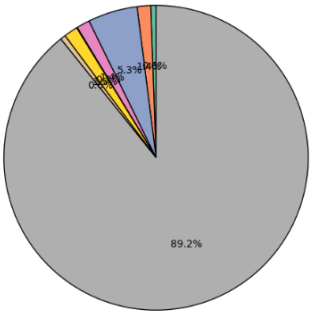
Cuadriciclo ligero



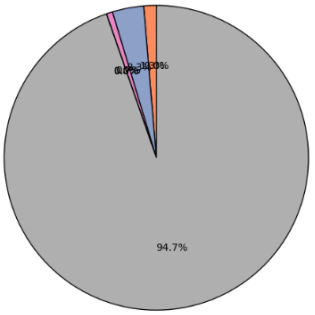
Cuadriciclo no ligero



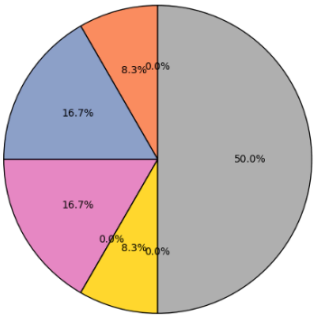
Furgoneta



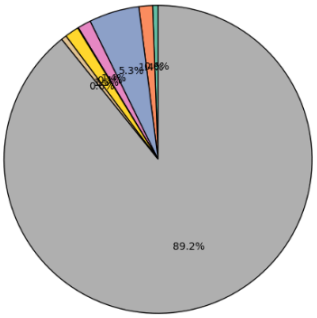
Maquinaria de obras



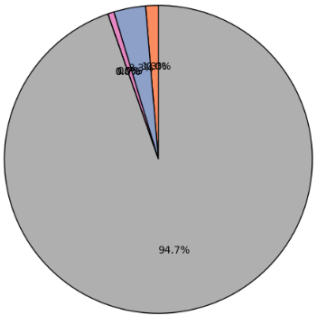
Cuadriciclo no ligero



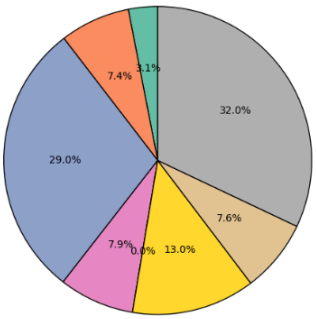
Furgoneta



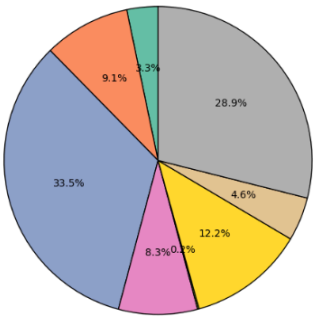
Maquinaria de obras



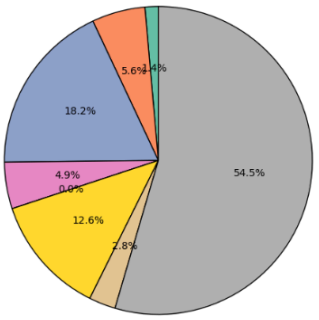
Motocicleta > 125cc

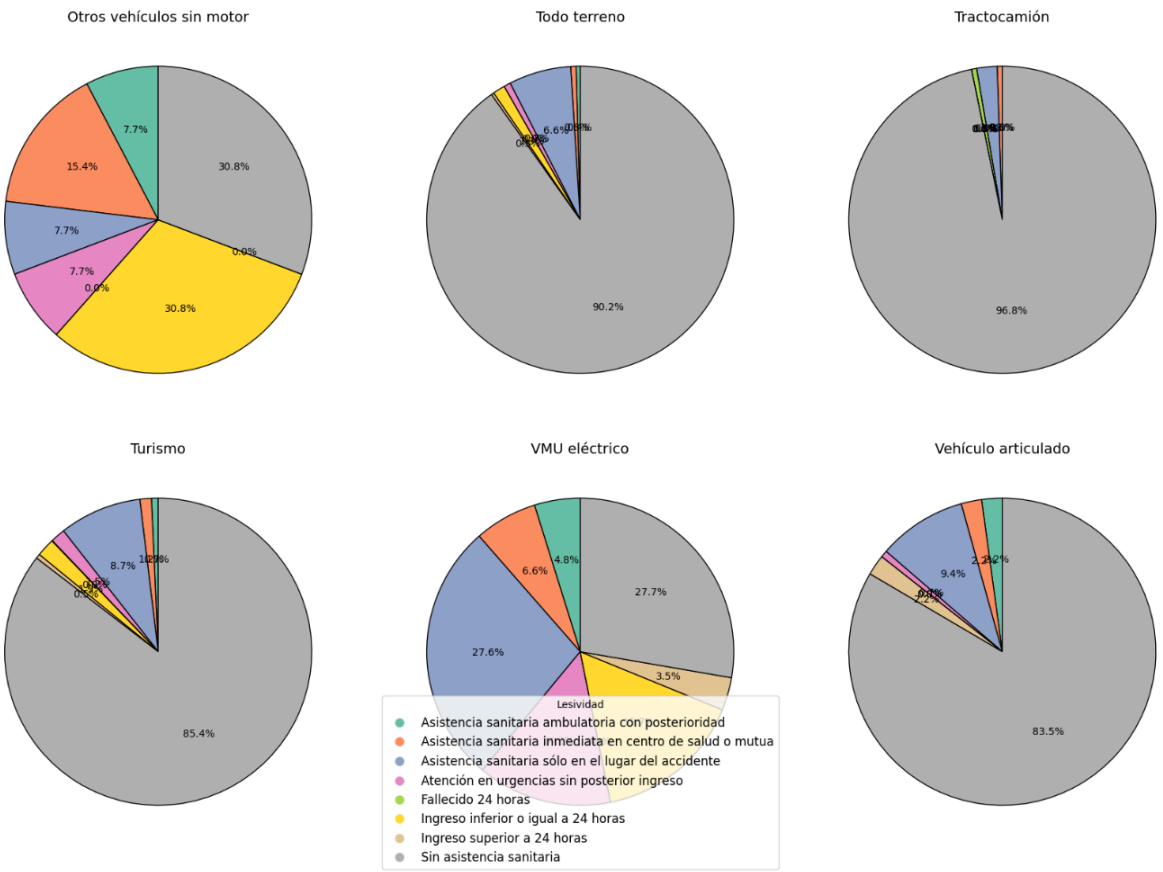


Motocicleta hasta 125cc



Otros vehículos con motor



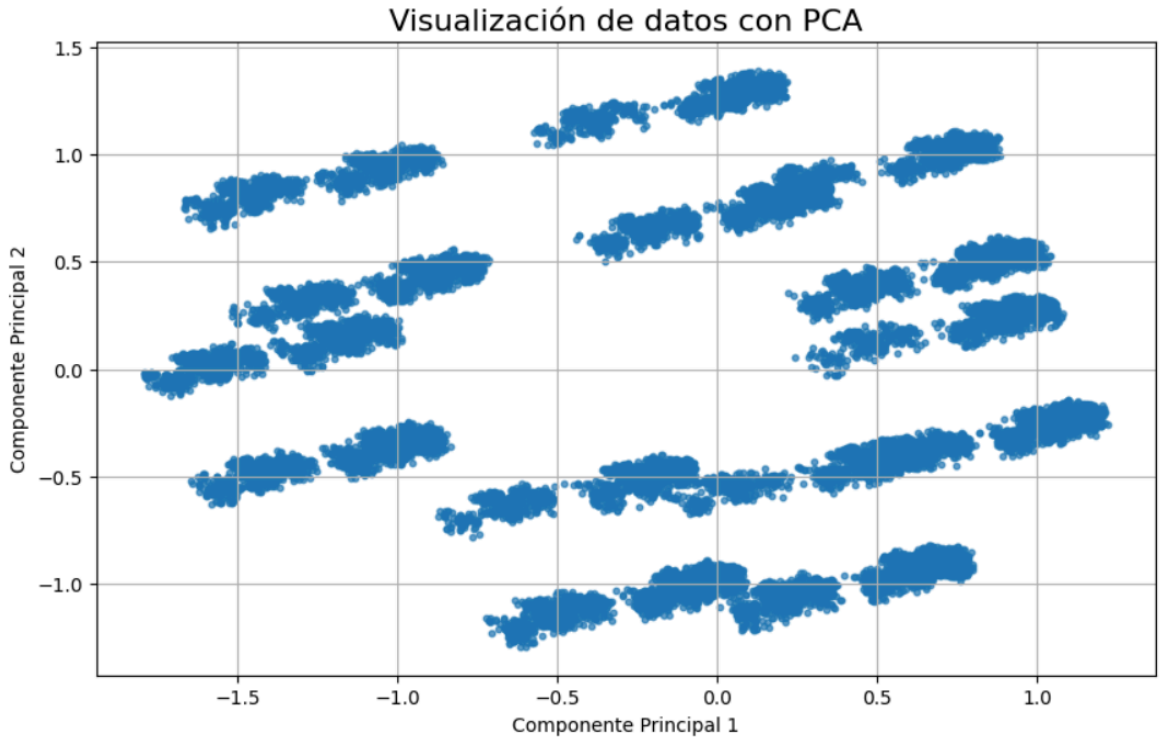


Analizando la gravedad de los accidentes en función del tipo de vehículo vemos como vehículos de dos ruedas como la bicicleta o las motos. Además, con vehículos de movilidad personal o cuadriciclos también hay un riesgo alto de necesitar asistencia sanitaria. Una de las observaciones que más nos ha llamado la atención es el gráfico del autobús de la EMT. El porcentaje de ingreso mayor a 24 horas (GRAVE) es altísimo en comparación con muchos otros vehículos. Esto seguramente se deba a que en este tipo de autobuses, los pasajeros muchas veces están de pie agarrados a una barra sin sujeción. Un choque puede causar graves consecuencias en alguien que no está atado al autobús. Por otro lado, vehículos como los camiones, furgonetas, maquinaria de obras,... incluso los turismos, son muy seguros y en la mayor parte de los casos no se necesita ningún tipo de asistencia sanitaria.

Una vez hemos estudiado todas estas visualizaciones, hacemos unas últimas operaciones de preproceso: eliminamos la columna número y transformamos las variables temporales para hacerlas cíclicas y así trabajan mejor los algoritmos de clustering.

Tarea 2.1

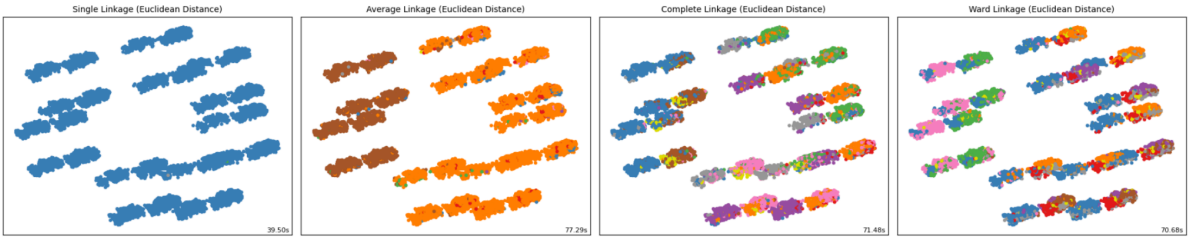
Hemos elegido los algoritmos de AglomerativeClustering y Birch para realizar este ejercicio. Para aplicar dichos algoritmos, antes de nada escalamos los datos y hacemos una primera visualización reduciendo el dataset a 2 dimensiones con ayuda de PCA.



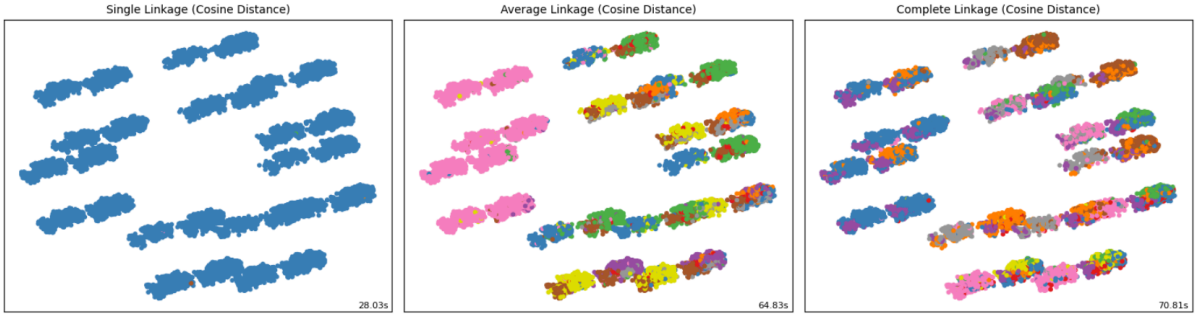
Obtenemos este gráfico que nos ayuda a ver la estructura de los datos con los que estamos trabajando.

Primero, vamos a aplicar AglomerativeClustering con cada tipo de distancia y método de enlace distinto. Para todos ellos vamos a utilizar 10 como valor de n_cluster para ver cómo se agrupan los datos en función de la distancia y de los métodos de enlace.

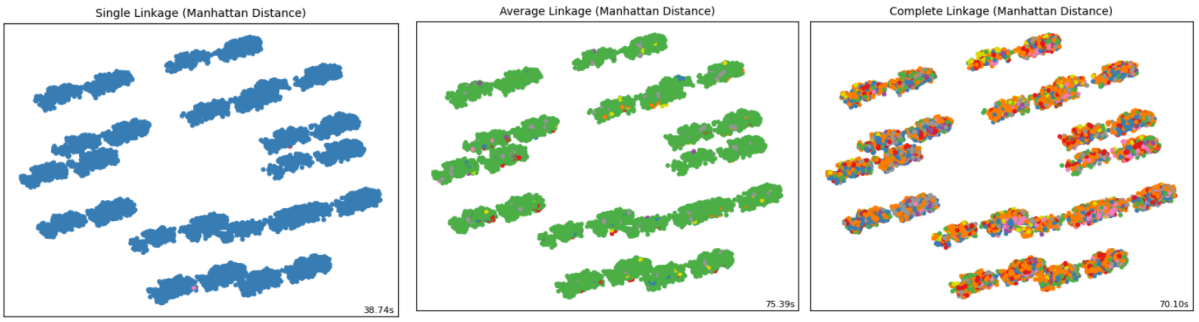
- Distancia euclídea:



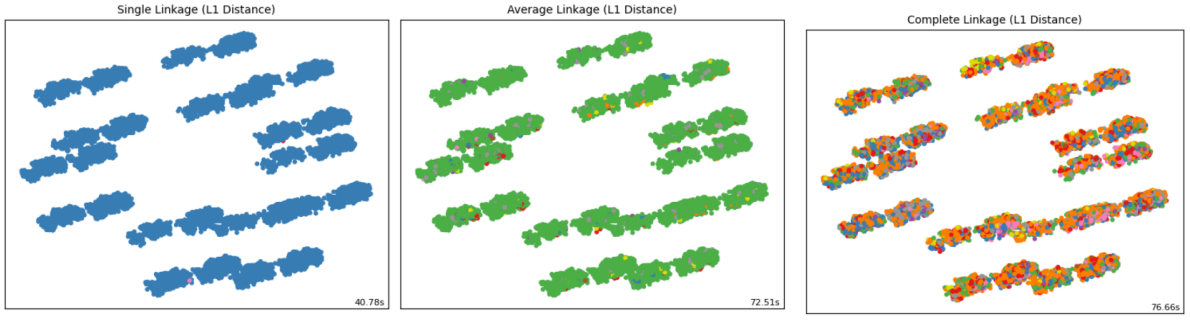
- Distancia coseno:



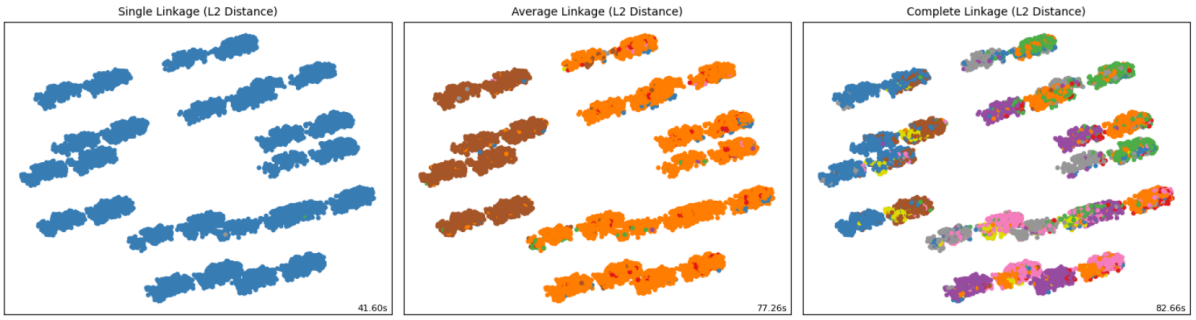
- Distancia Manhattan:



- Distancia l1:



- Distancia l2:



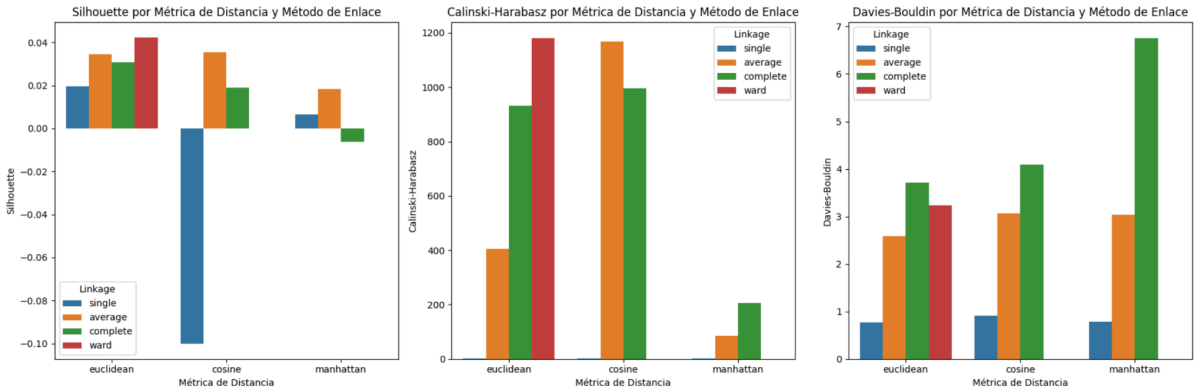
Si nos damos cuenta, el clustering realizado con la métrica L1 es muy parecido (por no decir casi idéntico) al clustering realizado con la distancia de Manhattan. Esto mismo ocurre con L2 y la distancia euclídea. Esto se debe a que la distancia L1 es, por definición, la suma de las diferencias absolutas entre las coordenadas de los puntos, lo que corresponde a la distancia Manhattan, mientras que la distancia L2 calcula la raíz cuadrada de la suma de los cuadrados de las diferencias, que es la distancia euclídea.

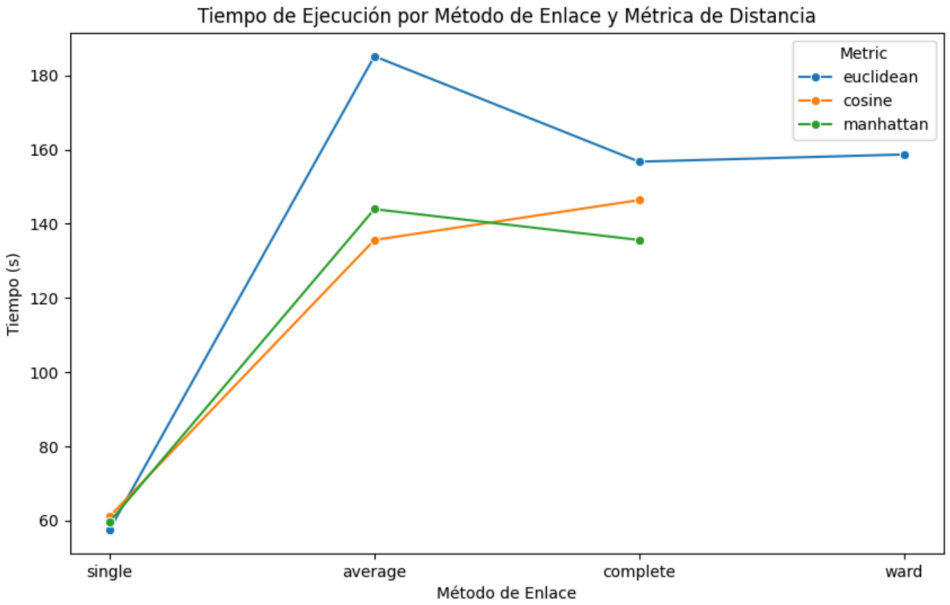
Ahora, vamos a evaluar con índices de calidad cada una de las distancias (a partir de ahora no contamos ni con l1 ni con l2 por lo que acabamos de explicar) y comparar resultados. Hemos

puesto como n_clusters el valor 10 para todas las distancias para ver el efecto que produce cada una de ellas. Para un análisis más profundo deberíamos probar con más clusters pero eso lo vamos a hacer más adelante.

	Metric	Linkage	Time (s)	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	euclidean	single	57.600439	0.019704	1.689552	0.770186
1	euclidean	average	185.153580	0.034547	406.263820	2.592987
2	euclidean	complete	156.732489	0.030678	931.832858	3.718638
3	euclidean	ward	158.671265	0.042336	1180.652864	3.231827
4	cosine	single	61.124115	-0.100048	1.233233	0.912685
5	cosine	average	135.616617	0.035299	1166.794972	3.058628
6	cosine	complete	146.386522	0.018791	995.431124	4.086613
7	manhattan	single	59.681377	0.006624	1.642110	0.781190
8	manhattan	average	143.942468	0.018226	86.437892	3.040511
9	manhattan	complete	135.629528	-0.006208	205.924290	6.755255

Obtenemos estos resultados los cuales vamos a graficar para verlos mejor. Tenemos una gráfica con los resultados de los índices y otra con los tiempos de ejecución



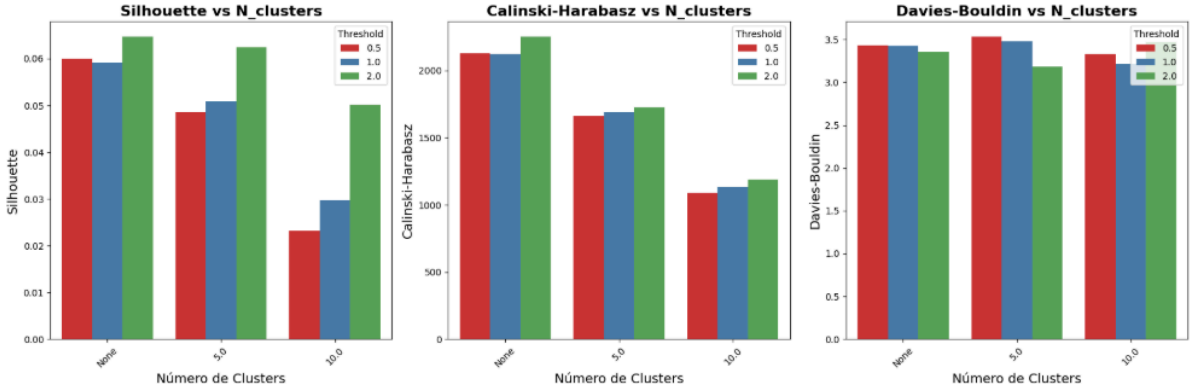


Distancia Euclídea:
Basándonos en Silhouette, produce los mejores resultados cuando agrupamos nuestros datos en 10 clusters. Los mejores métodos de enlace son ward (método específico de esta distancia) y average mientras que el método single tiene un rendimiento muy inferior al resto. Por último, hay que decir que es la distancia con mayor tiempo de ejecución.

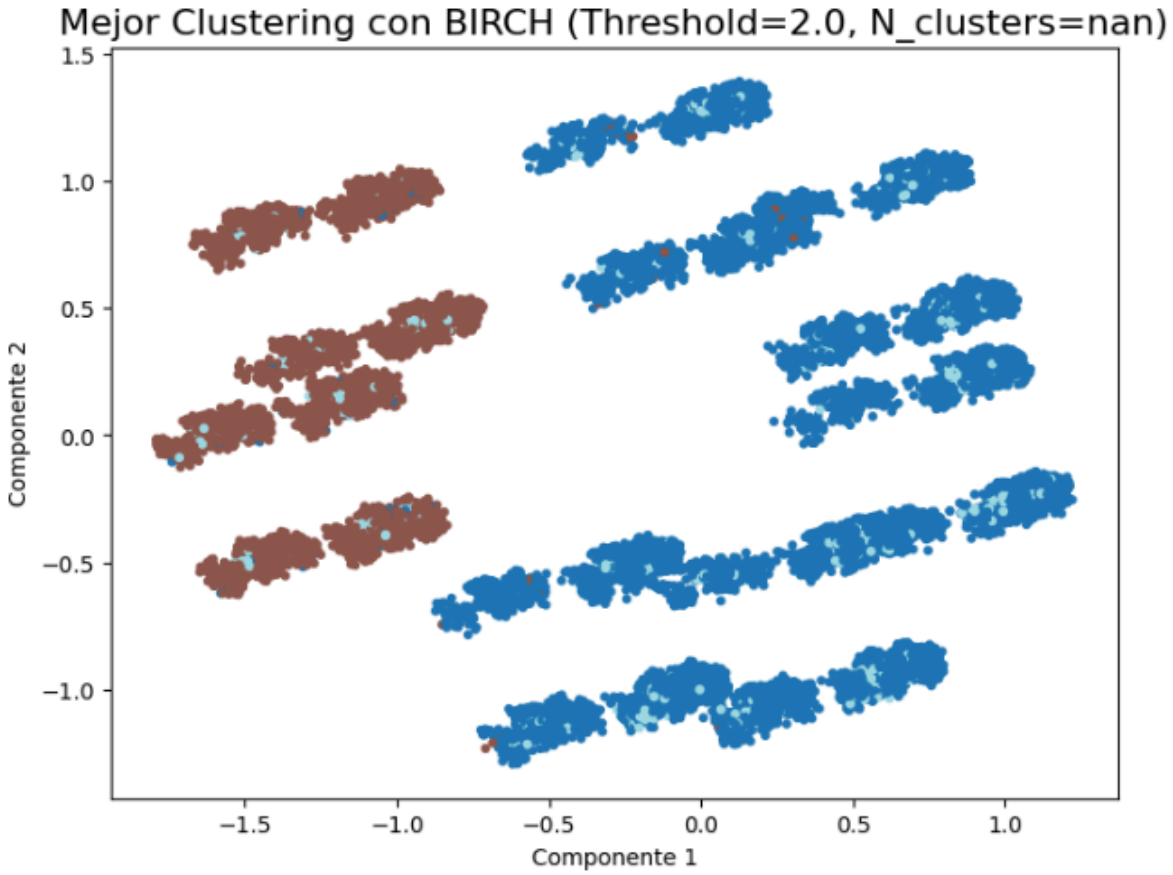
Distancia Manhattan:
El método de enlace que parece que tiene un mejor rendimiento con esta distancia, es average. Los resultados son ligeramente peores que con las otras dos métricas, aunque según el índice de Calinski-Harabasz Manhattan funciona realmente mal.

Distancia Coseno:
Con el método de enlace single, la distancia coseno no agrupa bien los datos dando lugar a un índice de Silhouette negativo. Sin embargo, el método average funciona bastante bien y logra aproximarse a la distancia euclídea. Además, aplicando los otros índices de evaluación vemos que se asemejan bastante los resultados a la distancia euclídea. Por lo que si queremos agrupar en 10 clusters nuestros datos, la distancia coseno con el método average es una buena opción ya que tarda menos en ejecutarse que la distancia euclídea.

Vamos a aplicar nuestro segundo algoritmo de clustering jerárquico. En el caso de Birch, vamos a estudiar los efectos de diferentes valores de n_clusters y threshold.

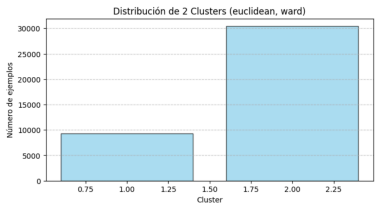


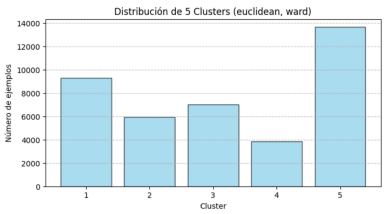
Podemos observar cómo los mejores resultados surgen cuando no especificamos un valor de n_clusters. Vamos a graficar el mejor modelo de Birch (threshold=2.0, n_clusters=None) para ver cómo agrupa los datos sobre la representación en 2 dimensiones de PCA.

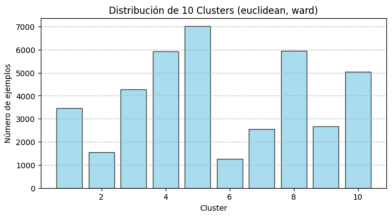


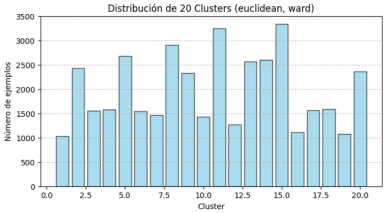
Tarea 2.2

Graficamos los dendrogramas de cada métrica y cada método de enlace, además de analizar la distribución de los datos por cluster. Los dendrogramas los vamos a analizar en el 2.3. Hemos utilizado distribuciones agrupando los datos en 2, 5, 10, y 20 clusters para obtener toda la información posible.

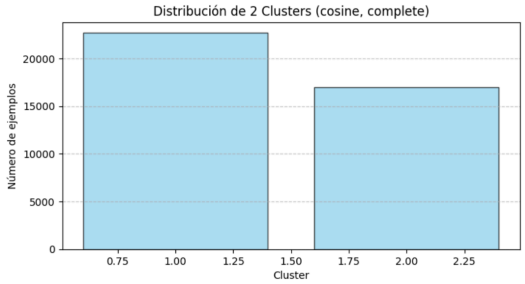




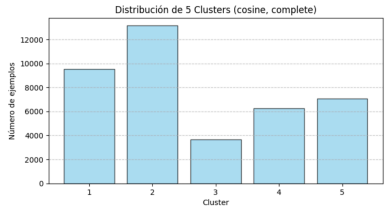


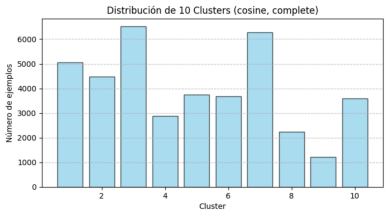


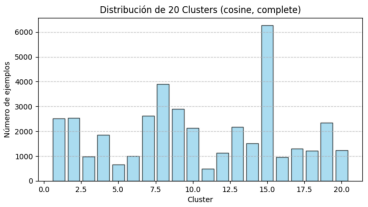
Esto es un ejemplo del método ward y la distancia euclídea. Vemos como no están uniformemente distribuidos los datos en los clusters. En el resto de métodos y distancias tampoco hay uniformidad entre el tamaño de los clusters. Quizá la mayor aproximación a una distribución uniforme, la encontramos con la distancia coseno y el método de enlace completo.



Sin embargo, al aumentar el número de clusters, se van desnivelando los tamaños de las agrupaciones.

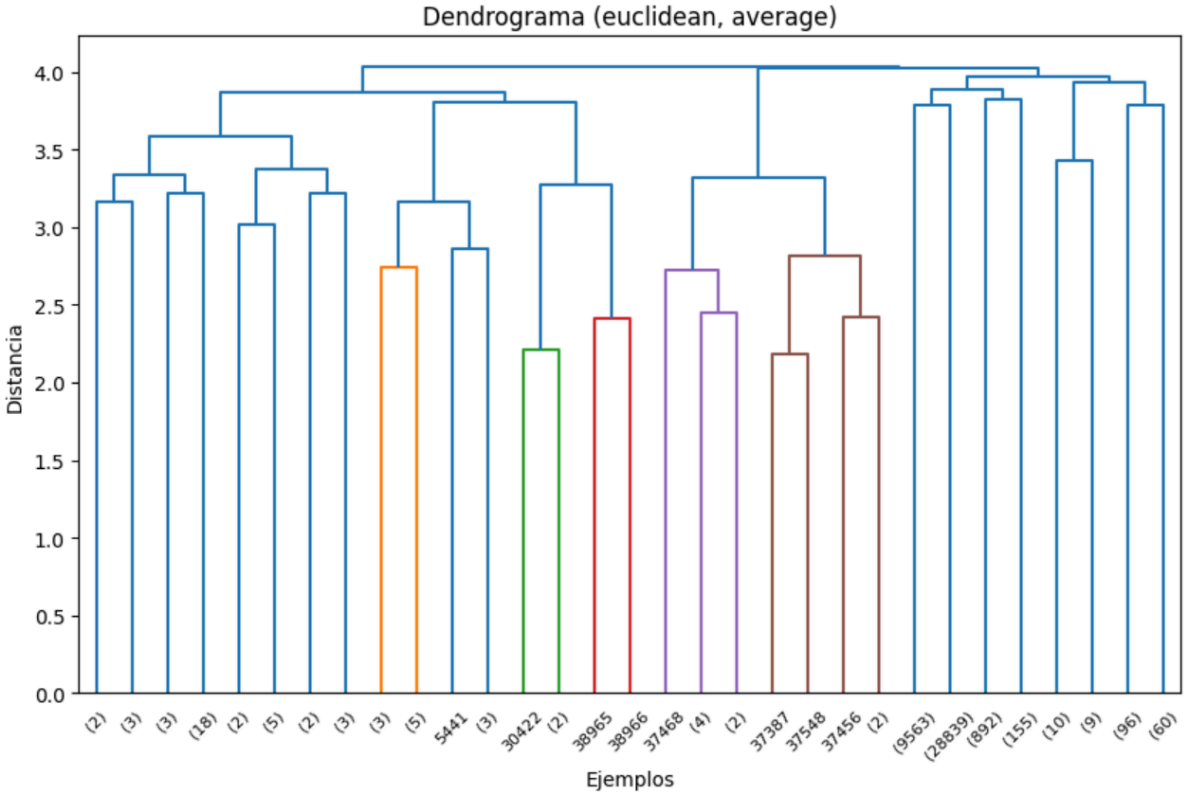






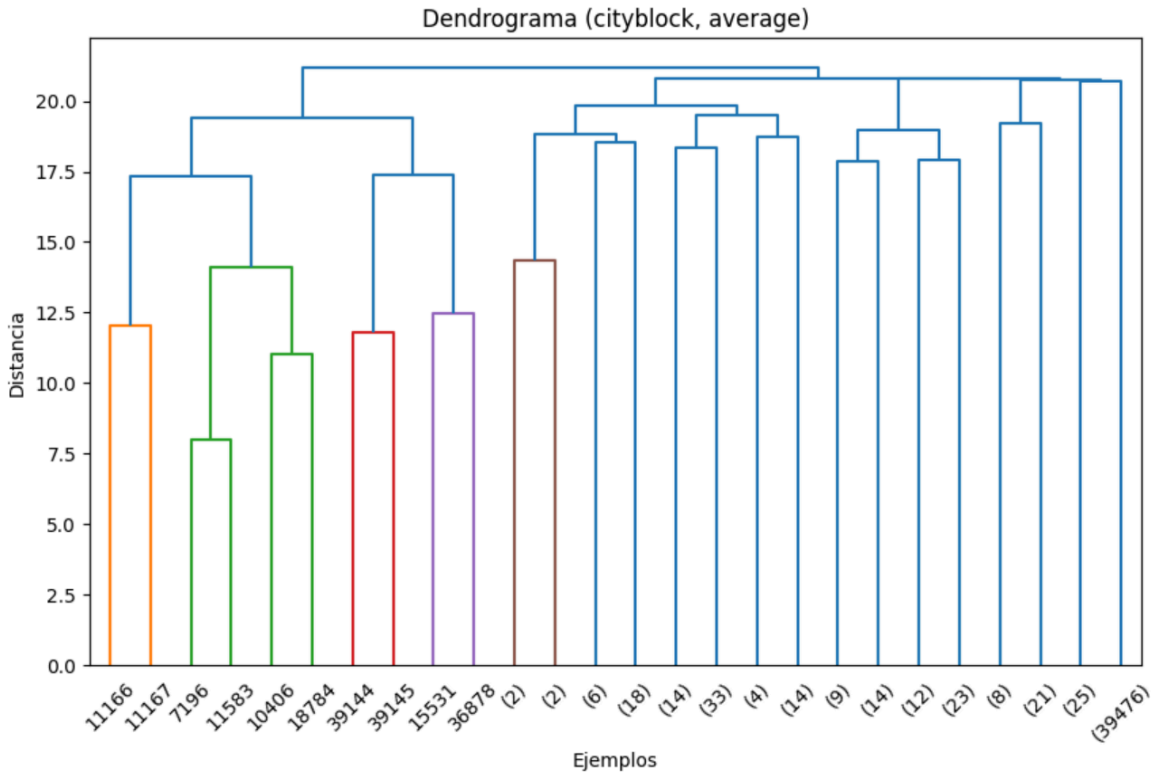
Tarea 2.3

Para el análisis de dendrogramas, vamos a fijarnos en el método de enlace average de cada distancia.

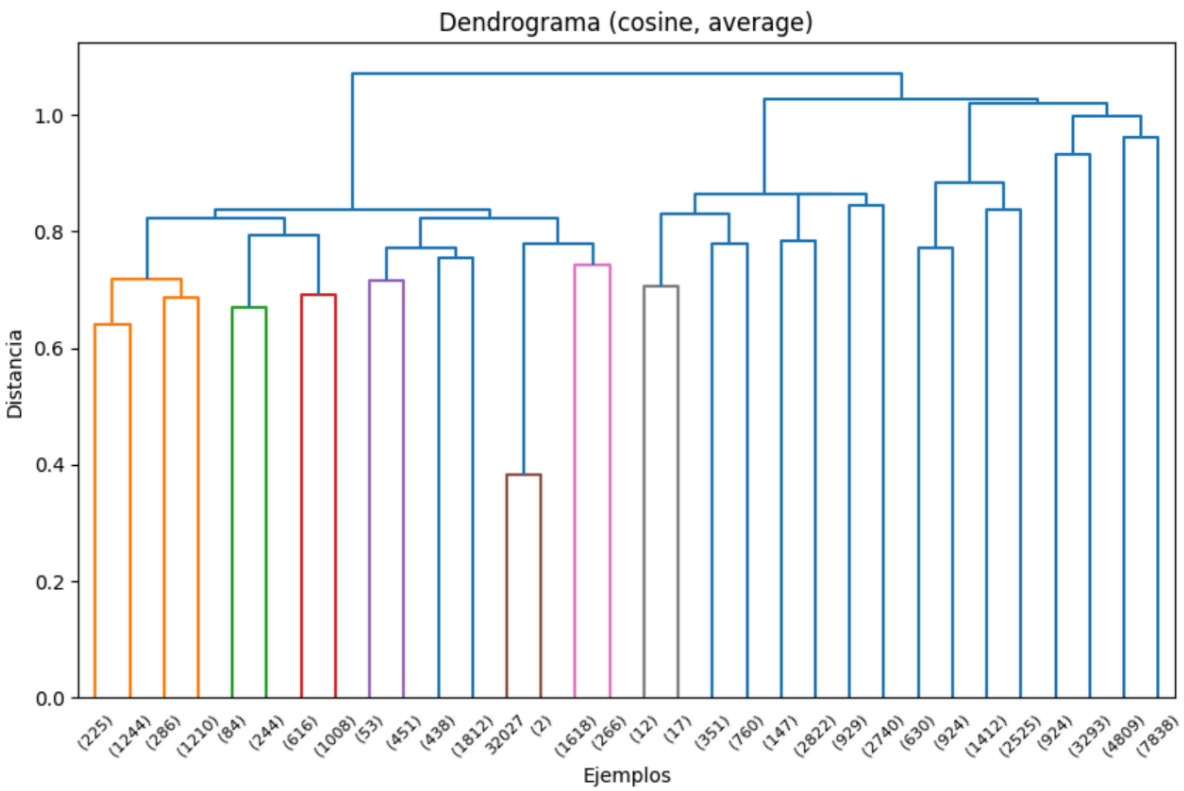


Con la distancia euclídea, medimos la distancia geométrica directa entre dos puntos en un espacio multidimensional. La unión de los clusters ocurre a grandes diferencias absolutas entre las observaciones. Esto es característico de esta distancia, ya que es sensible a magnitudes absolutas.

Los clusters pequeños (como (3) o (5) de color naranja) se forman rápidamente, pero su unión a otros grupos ocurre a distancias relativamente grandes. Esto puede ser debido a que algunos valores extremos pueden estar afectando a la estructura del dendrograma.



En este dendrograma, observamos algunas agrupaciones tempranas como 11166 y 11167 o (7196) y (11583), lo cual sugiere que estas observaciones tienen diferencias absolutas pequeñas. Sin embargo, en la parte de la derecha del gráfico, la unión de grupos más grandes ocurre a distancias mucho mayores (e.g., (25) y (39476)), reflejando la influencia de los valores totales en varias dimensiones. A diferencia de la métrica Euclidiana, aquí las distancias extremas no dominan tanto la estructura.



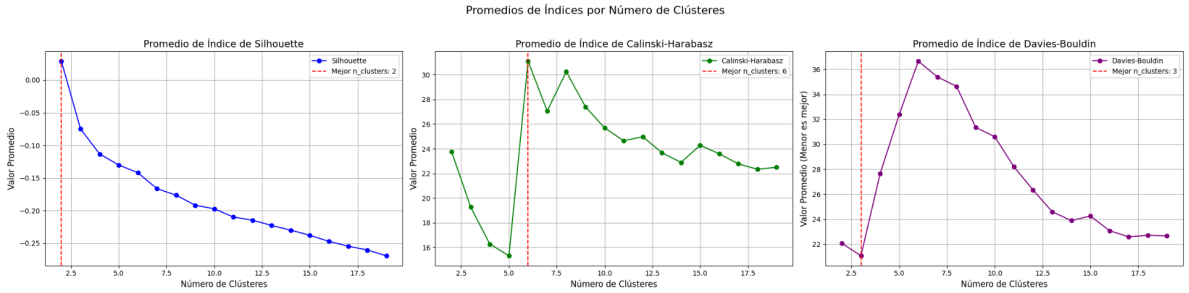
Las uniones de los clusters de la parte izquierda del gráfico, ocurren en distancias pequeñas. Esta distancia establece la similitud con patrones y no se guía tanto por la magnitud como las otras dos distancias. Por ejemplo, los clusters como (225) y (1244) se agrupan rápidamente, al igual que otros clusters por la parte izquierda del gráfico.

Por otro lado, observamos que otros valores como (1812) o (924) permanecen separados por más tiempo. Este dendrograma da lugar a una estructura más compacta, con grupos que parecen ser más homogéneos en términos de orientación de sus datos.

Analizando un poco a nivel general los tres dendrogramas, podemos deducir que hay grandes grupos de datos bastante alejados al resto independientemente de la distancia, por lo que las uniones en los dendrograma ocurren bastante tarde y por esta razón puede que más adelante, agrupar por 2 clusters nos dé buenos resultados.

Tarea 2.4

En este apartado, hemos utilizado tres índices de calidad de clustering distintos: silhouette, calinski harabasz y davies bouldin. Para obtener el mejor resultado posible, contamos con un amplio rango de n_clusters. Para cada valor, aplicamos Agglomerative Clustering con todas las combinaciones de métricas/métodos de enlace posibles. Luego calculamos los tres índices mencionados anteriormente y almacenamos dichos valores. Por último vamos a hacer un promedio de todos los resultados y así obtener el promedio de cada índice por cada n_clusters. El promedio más alto será el valor de n_clusters más óptimo.



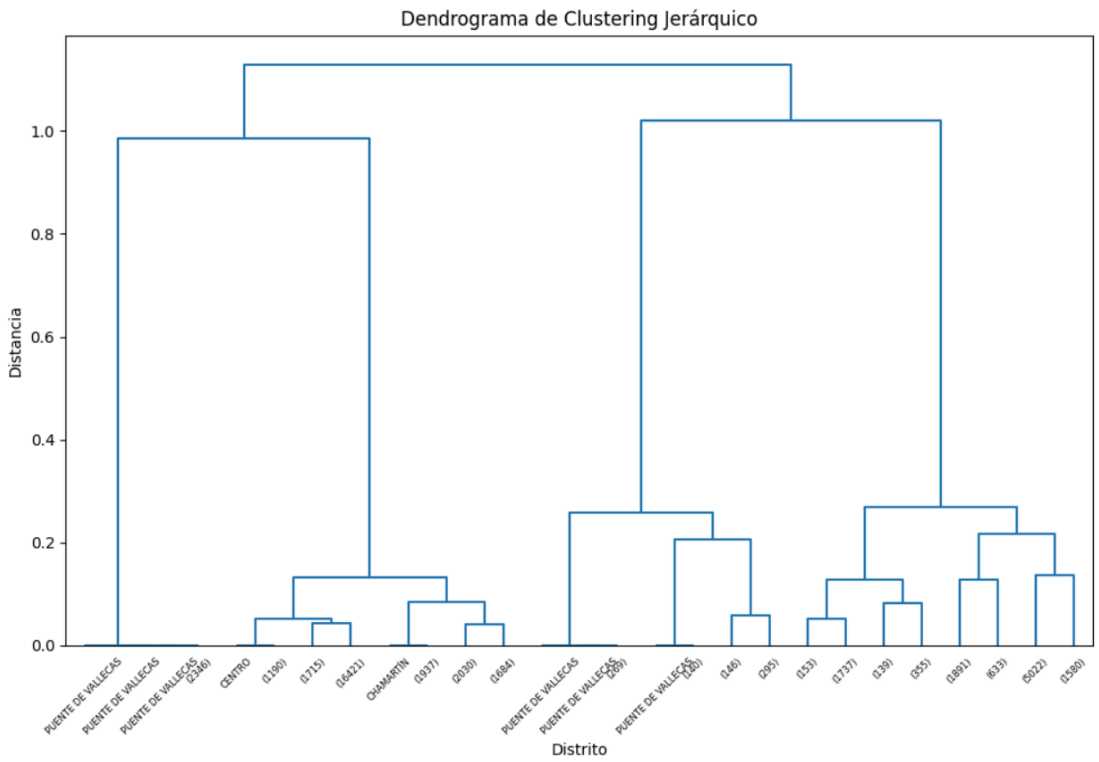
Este es el resultado de los promedios. Cada índice nos da un resultado distinto si observamos estas gráficas. Sin embargo podemos observar que con 2 clusters, conseguimos buenos resultados en los 3 tipos de índices, siendo el mejor promedio de índice de Silhouette y el segundo mejor con Davies-Bouldin.
Vamos a almacenar los mejores valores de cada métrica por cada valor n_cluster en un DataFrame para poder hacer una comparación final entre algoritmos de clustering.

Tarea 2.5

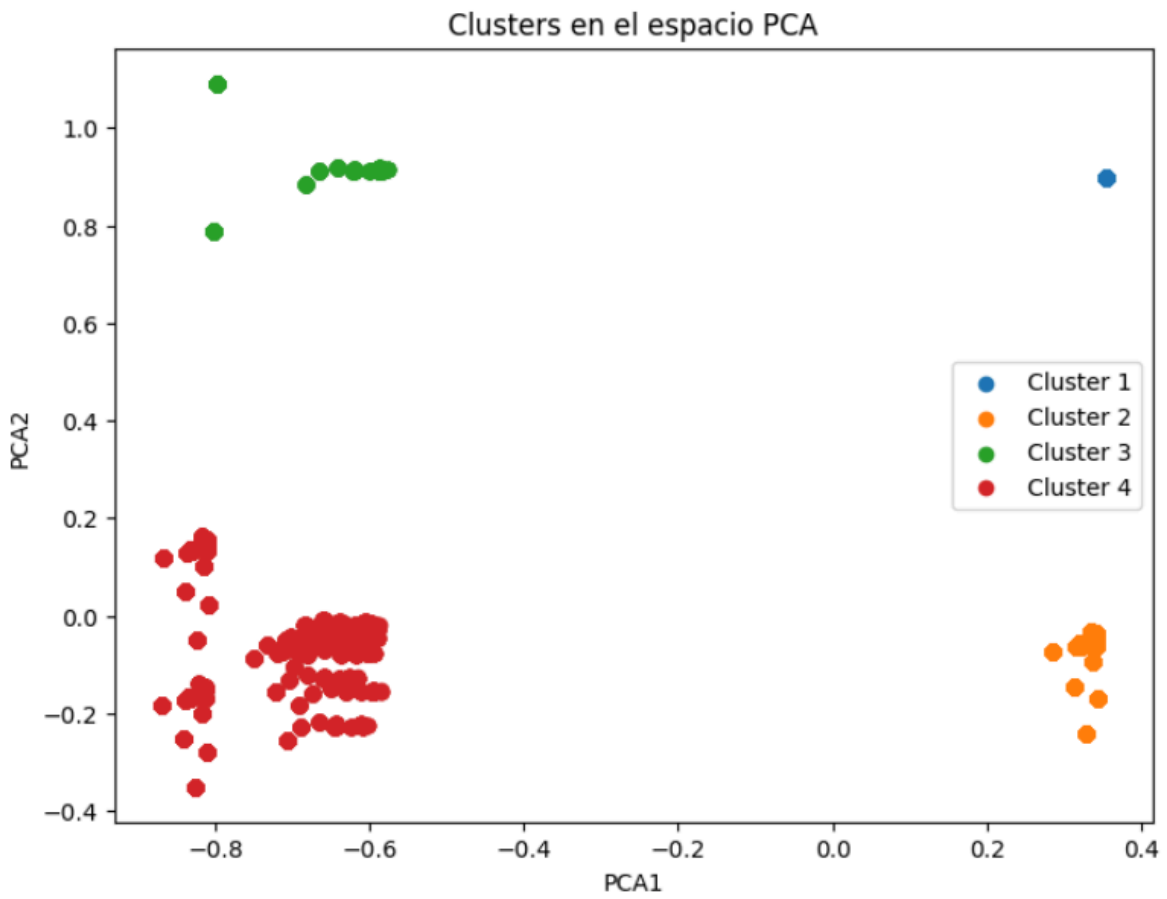
Como hemos podido observar en las gráficas anteriores, el número óptimo de clusters es 2. El índice de Silhouette es la métrica más fiable de las tres y además este valor logra buenos resultados con el resto de métricas.
De esta forma, podemos concluir que utilizando algoritmos de clustering jerárquico, agrupar nuestros datos en 2 clusters es la mejor decisión según los índices elegidos.

Tarea 2.6

Para realizar este ejercicio, vamos a hacer un enfoque inicial para entender el problema. vamos a agrupar las columnas de vehículo sin especificar el tipo. Para ello, vamos a utilizar las columnas referentes a los tipos de vehículos y a las zonas/distritos donde han ocurrido accidentes. Les aplicamos reducción de dimensionalidad con PCA para poder ver los resultados gráficamente y estudiamos su dendograma.
.



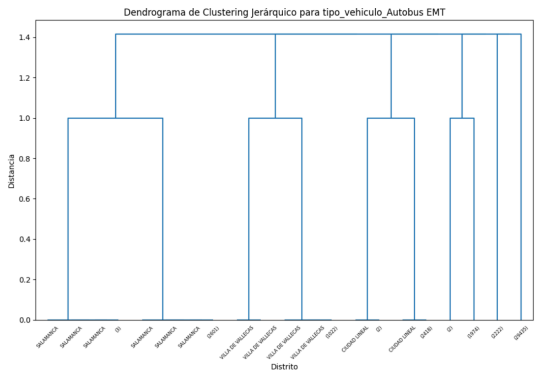
En función del dendrograma, vemos claramente cómo se agrupan en seguida 4 clusters y están muy separados entre sí, por lo que todo apunta a que 4 es el número óptimo de clusters. De esta forma, ejecutamos un algoritmo de clustering jerárquico con 4 clusters y graficamos con la representación de pca que hemos generado en la celda anterior.

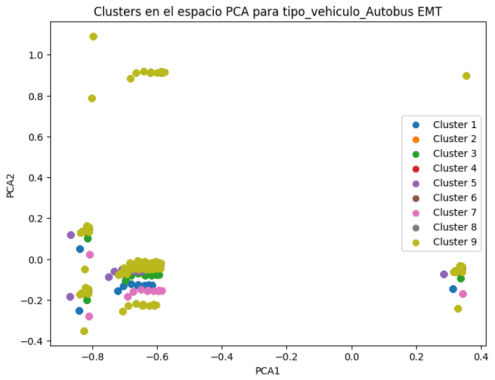


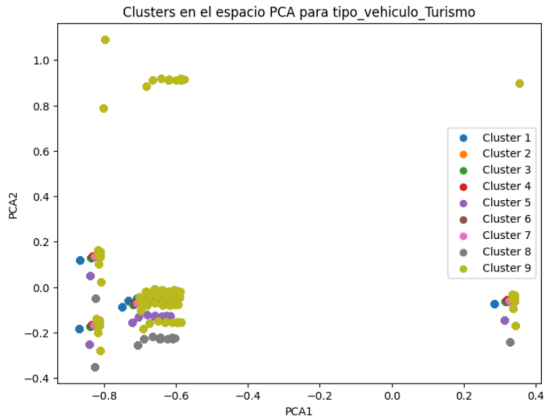
Vemos cómo hemos agrupado en cuatro conjuntos nuestros datos. Ahora vamos a hacerlo iterando por cada tipo de vehículo.

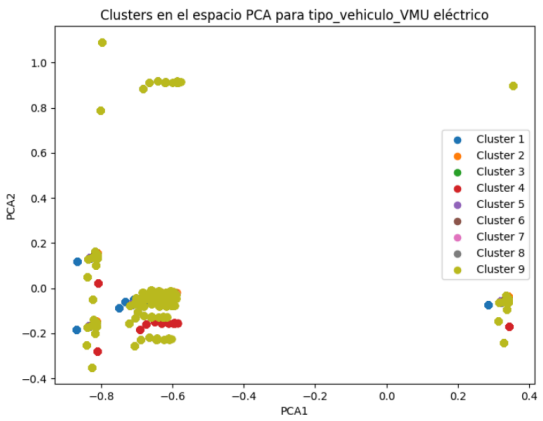
- **Enfoque por tipo de vehículo.**

La estrategia es la misma, solo que en vez de analizar el dendrograma a mano, utilizamos el índice de Silhouette para automatizar el proceso. Los resultados de los clusters por cada vehículo, los vamos metiendo como nuevas columnas en nuestro DataFrame y así poder analizarlos posteriormente.

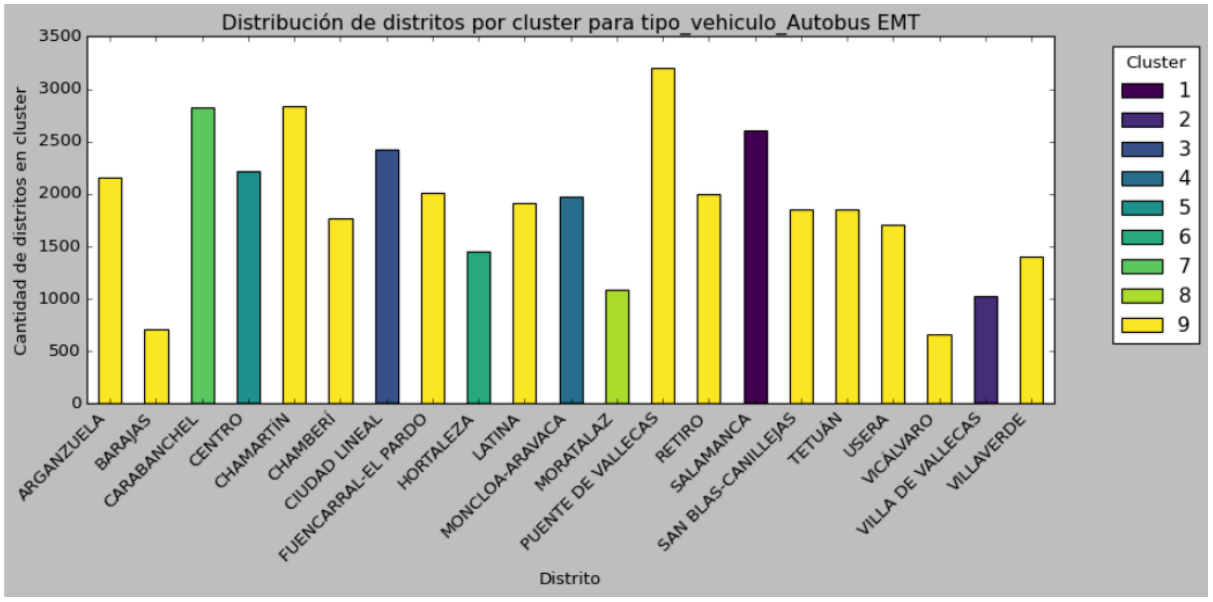


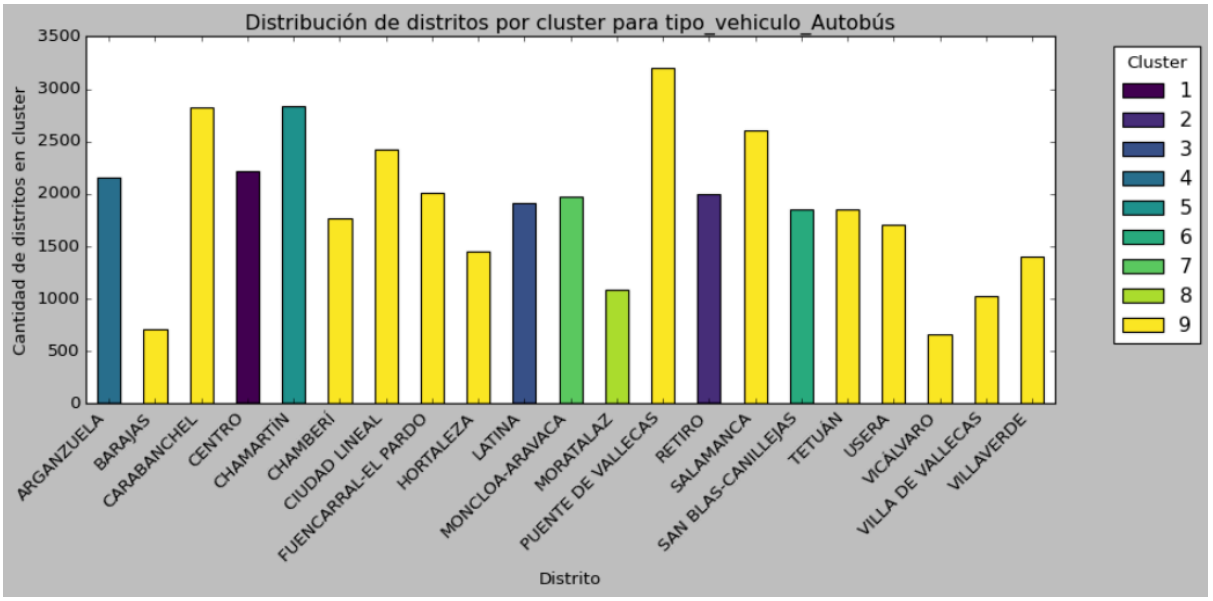


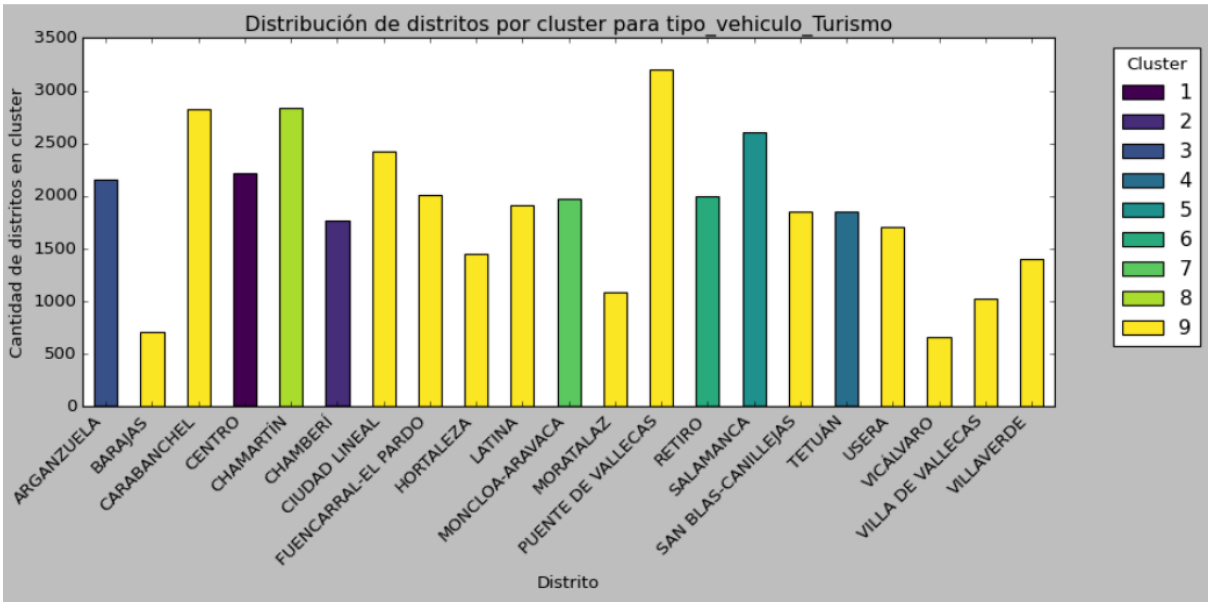




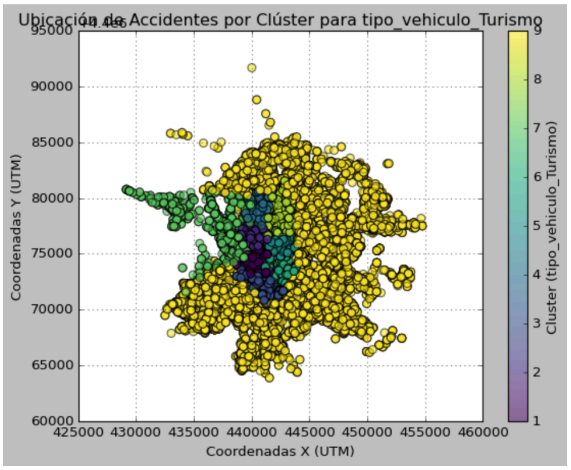
Hacemos un total de 21 dendrogramas y gráficas, una por cada tipo de vehículo. Para entender mejor los resultados, graficamos cómo se distribuyen los clusters por distritos.







Como podemos observar, zonas de Madrid sur como Puente de Vallecas, Usera, Vicálvaro, Arganzuela... y tirando un poco más hacia el este: Barajas, Canillejas... Suelen estar agrupadas en el mismo cluster, donde el índice de siniestralidad es mayor. Mientras que zonas del centro se agrupan en clusters distintos. Esto puede cambiar ya que por cada tipo de vehículo los resultados varían, pero en general los datos siguen esta línea . Vamos a verlo graficando las coordenadas de cada accidente como si fuera un mapa para entenderlo mejor.



En los mapas vemos mejor la distribución que hemos mencionado antes. Vamos a fijarnos en el gráfico del turismo ya que es el tipo de vehículo que más se repite con diferencia. vemos cómo se agrupan los accidentes por las zonas de madrid sur y el este debido a su proximidad causado por un gran número de accidentes cercanos. De este modo así lo agrupan las distancias de los algoritmos de clustering.
Por otro lado, vemos cómo se generan más clusters por el centro de Madrid y por la zona oeste donde la cantidad de accidentes es menor.

Tarea 3.1

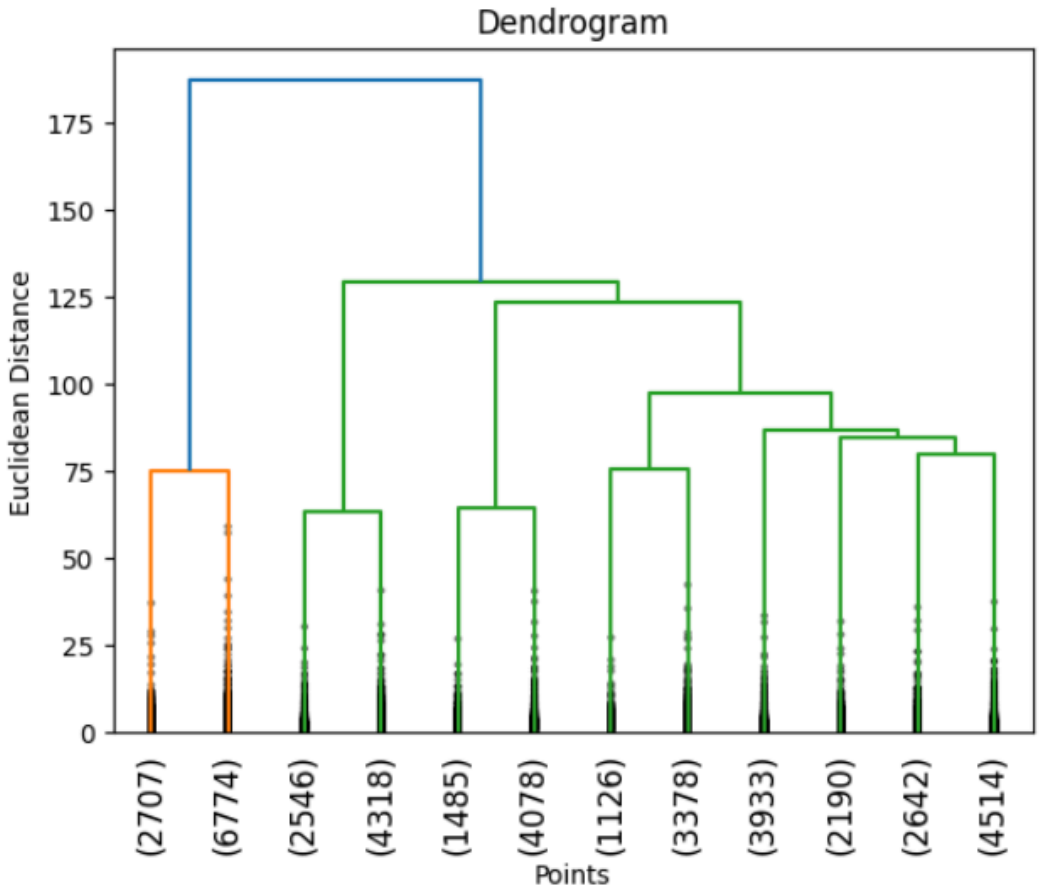
El preprocesamiento para aplicar el algoritmo de kmeans, consta de un normalizado con minmaxscaler. Esto es fundamental para preservar las distancias proporcionales entre los datos, pues el algoritmo de kmeans actua con distancias euclídeas.

Después se aplica un PCA para eliminar algunas columnas con menos relevancia, pero conservando un 95% de la variabilidad.

Tarea 3.2

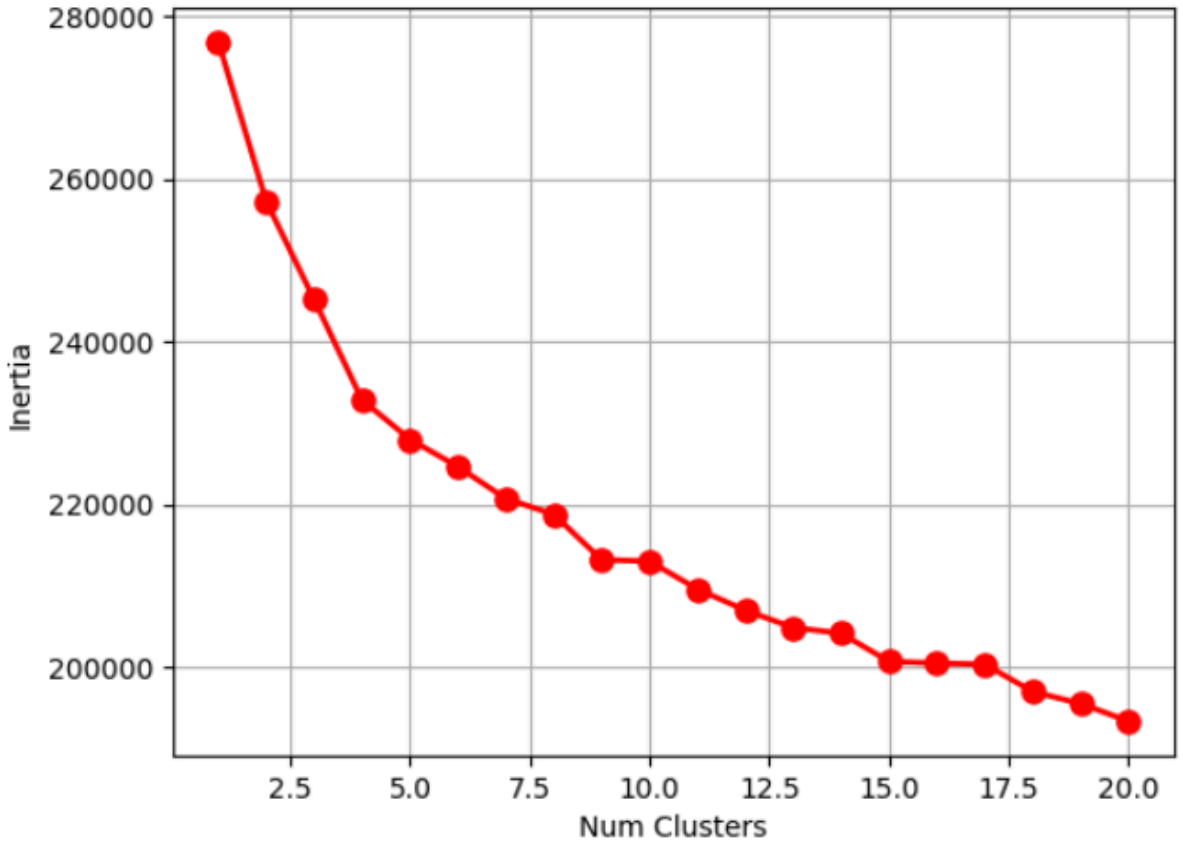
Para establecer el número más adecuado de clusters, se han empleado varios algoritmos. Se analizarán 1 por 1.

Dendograma



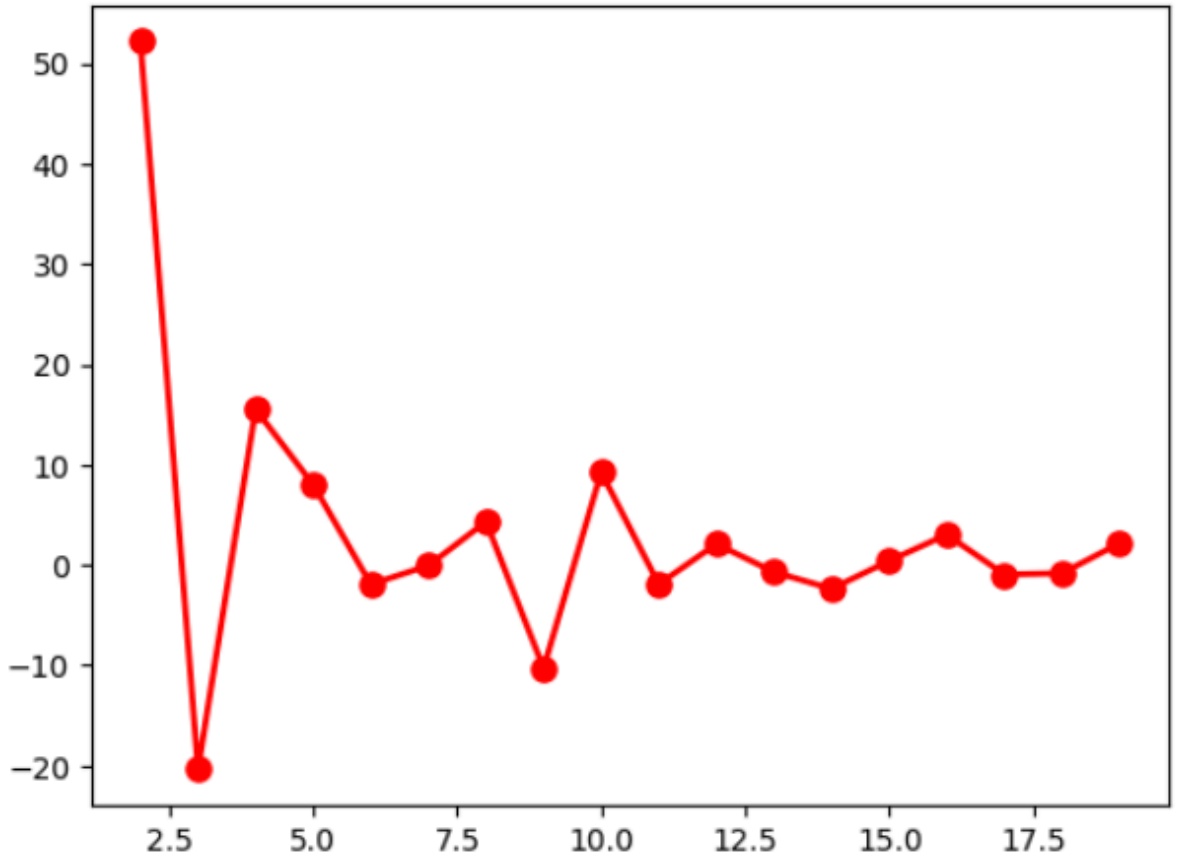
Se puede apreciar aquí que la mejor distancia parece ser una euclidean distance de 100 pues es cuando se puede ver un salto significativo de en la altura del dendograma, a partir de ahí los clusters comienzan a juntarse más rápidamente, por lo tanto aquellos que están por debajo de 100, son los clusters que están más separados. En total, se pueden contar 5 clusters.

Elbow Method



El Elbow Method evalúa cómo disminuye la inercia (suma de distancias cuadradas de los puntos a sus centroides) al aumentar el número de clusters en K-Means. El "codo" indica el punto donde agregar más clusters no reduce significativamente la inercia, señalando el número óptimo de clusters. Mirando la gráfica, destaca un codo para n_clusters=4, a partir de entonces el descenso de la pendiente se estabiliza.

GAP



El Gap compara la inercia de los clusters en los datos reales con la inercia en datos aleatorios simulados. El número óptimo de clusters es donde la diferencia entre estas inercias es máxima, indicando que los clusters son significativamente mejores que el ruido. Mirando la gráfica, este valor se encuentra para n_clusters=4, al igual que el Elbow Mehtod.

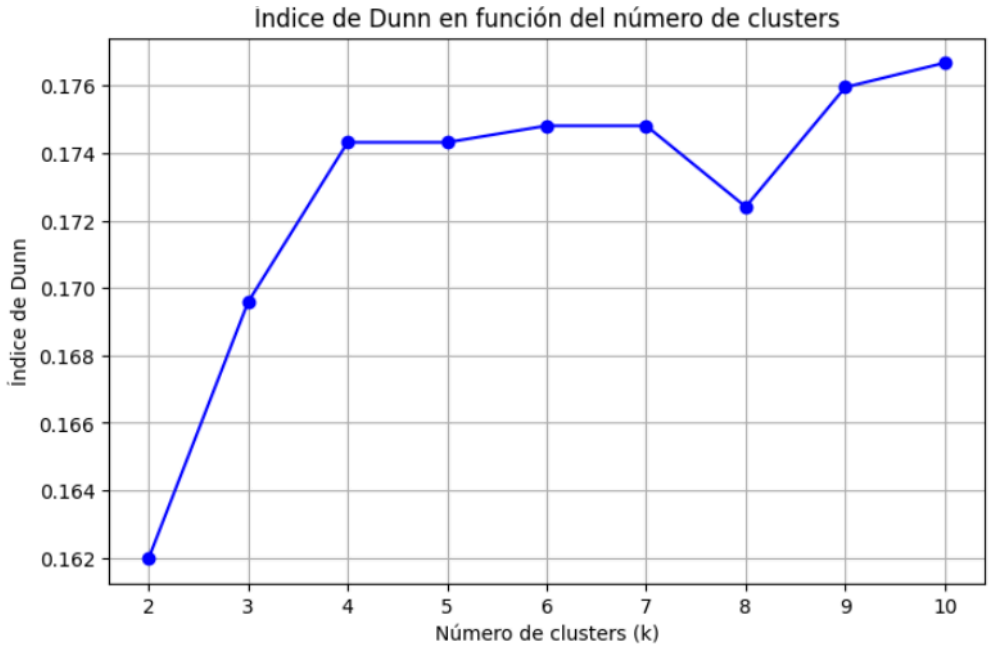
Conclusión

Como conclusión, el número óptimo de clusters que se ha elegido son 4, ya que tanto con el Elbow Method como con el GAP, se ha determinado que es el número óptimo. Además, con el dendograma el resultado ha sido de 5, lo cuál no está tan alejado.

Tarea 3.3

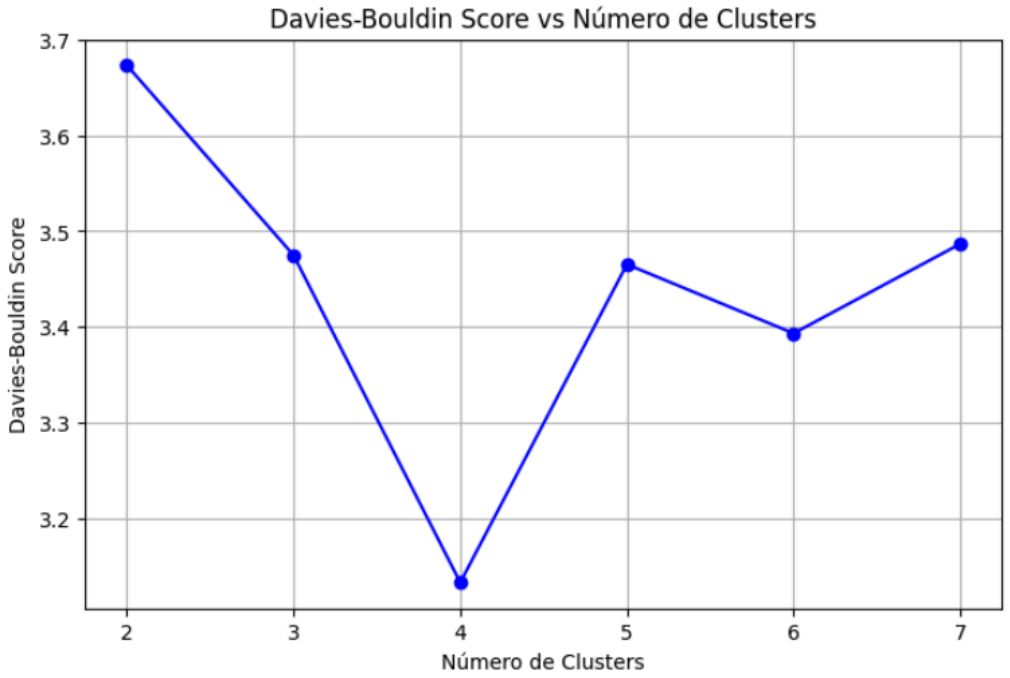
Para probar cómo varía la calidad del clustering con diferentes valores de k, se emplearon diferentes índices para valorarlo.

Dunn



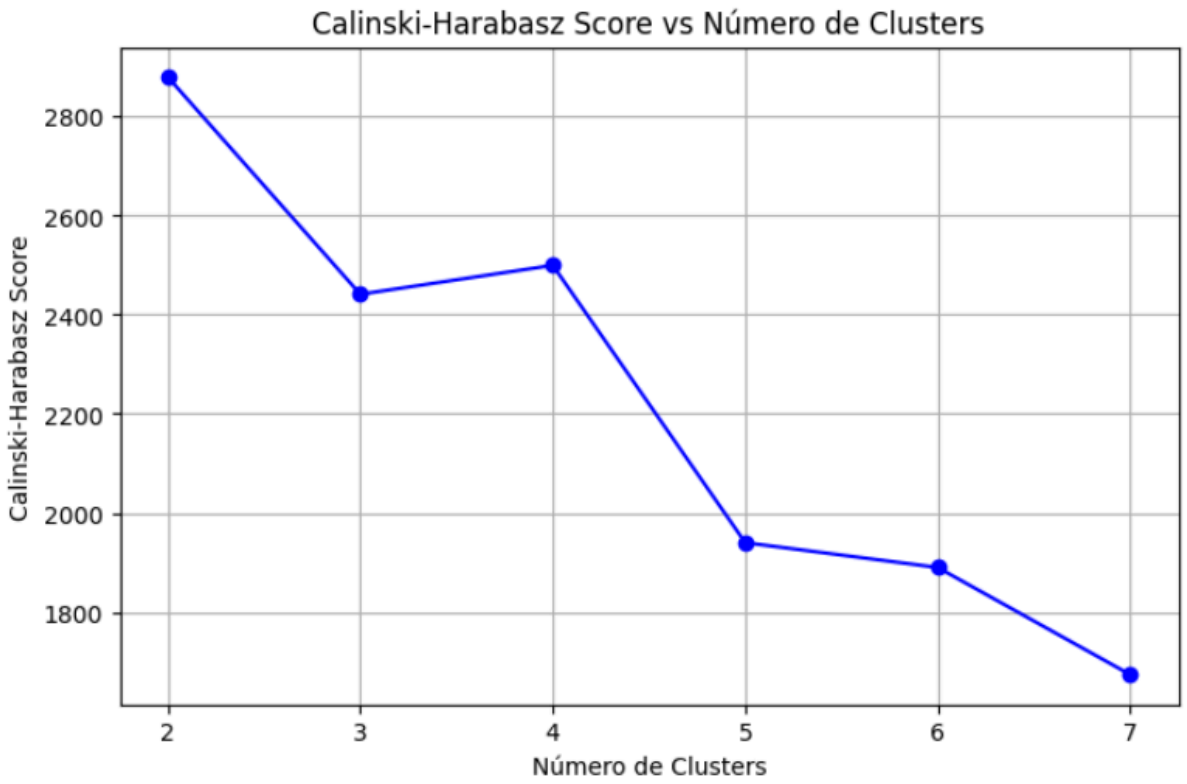
Con el índice de Dunn se puede apreciar que el número de clusters establecidos anteriormente (4), obtiene un buen valor en el índice de Dunn. Al parecer 9 o 10 clusters ofrecen valores ligeramente mejores que 4, es por ello que se comparan más índices para obtener una comparativa más globalizada.

Davies-Bouldin



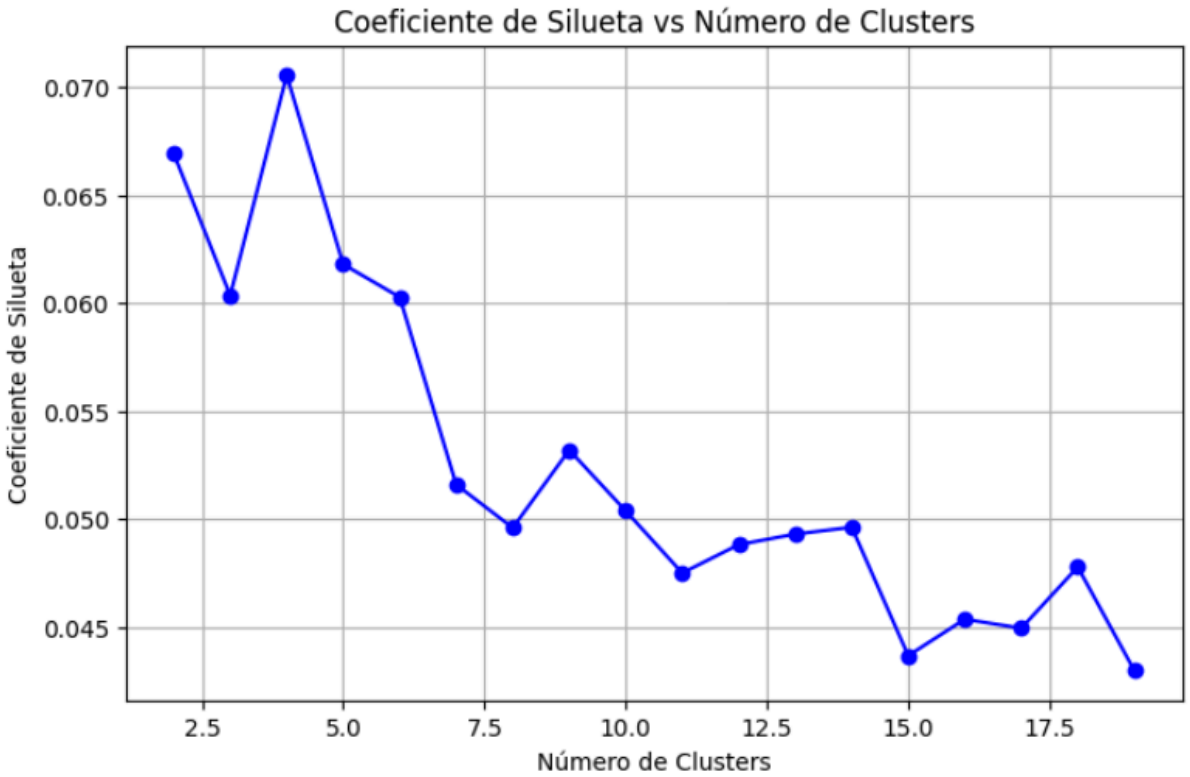
Para el índice de Davies-Bouldin, el mejor número de clusters, es aquel que obtiene el menor valor para dicho índice que en este caso es claramente el 4.

Calinski-Harabasz



Con Calinski-Harabasz, el número de clustetrs que mejores resultados da, es 2, seguido después de 4.

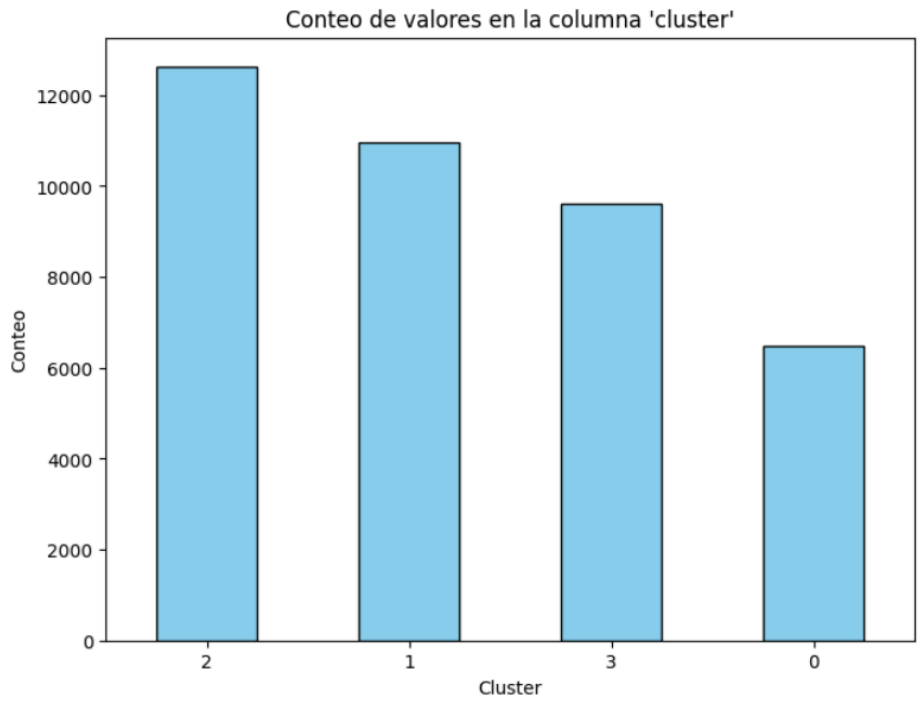
Silhouette



Observando el gráfico, se aprecia que el índice de silueta da como mejor valor claramente el 4, estando muy por encima del resto de valores.

Tarea 3.4

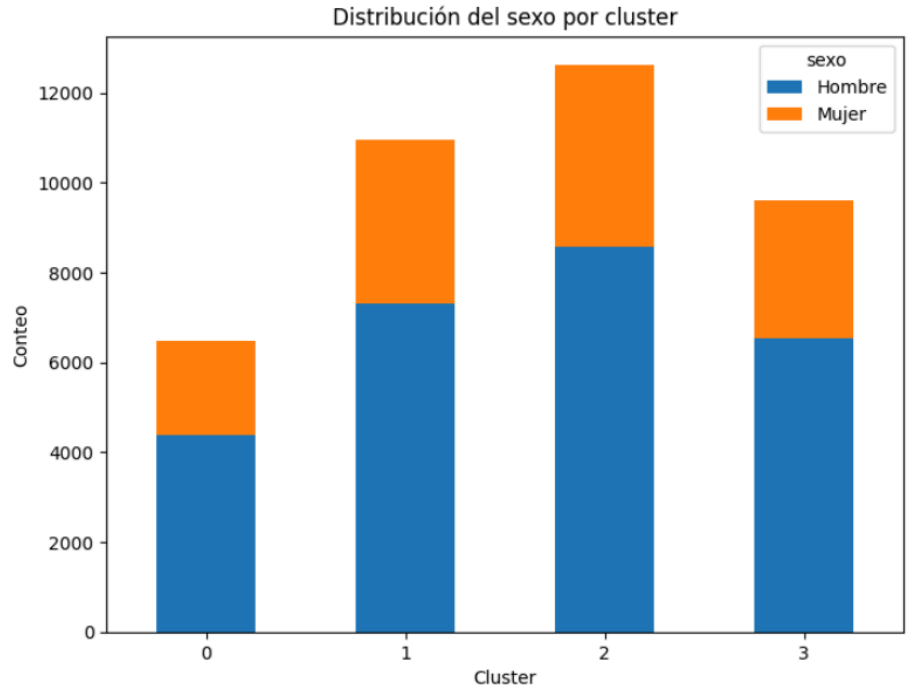
Dado el análisis anterior, vamos a agrupar nuestros datos en 4 clusters. A continuación, vamos a estudiar qué elementos encontramos en cada cluster y tratar de encontrar las características que diferencian uno de otro. Para ello, añadimos una nueva columna a nuestro dataframe indicando el cluster al que pertenece cada fila.



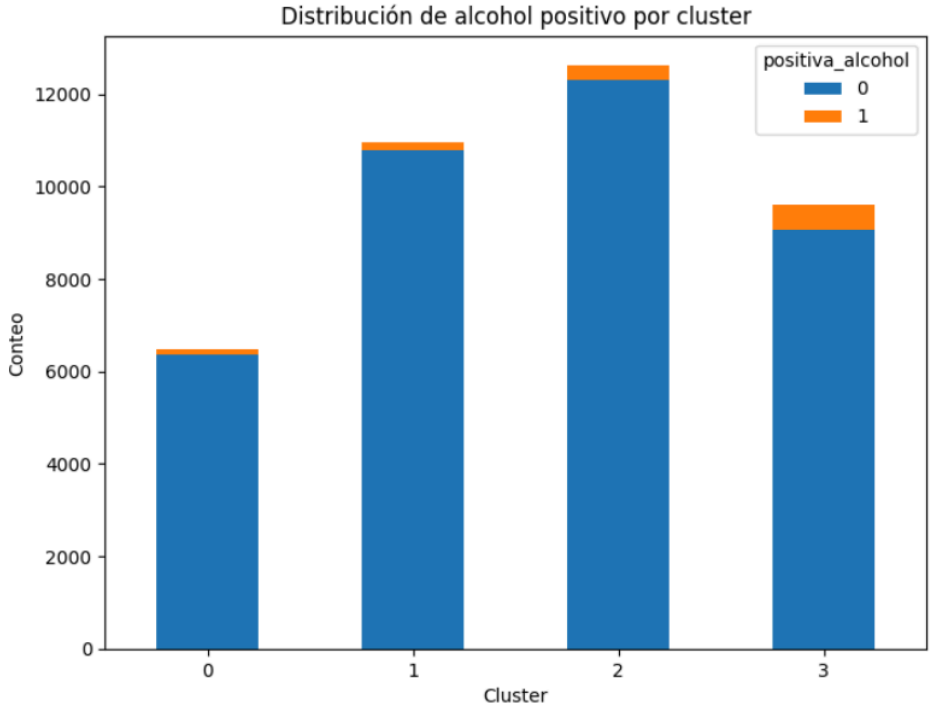

```
cluster
2    12633
1    10950
3     9618
0     6490
```

Vemos que la distribución de los ejemplos por cluster no es uniforme y que el cluster contiene poco más de la mitad de datos que el cluster 2. Vamos a analizar de qué están compuestos cada uno.

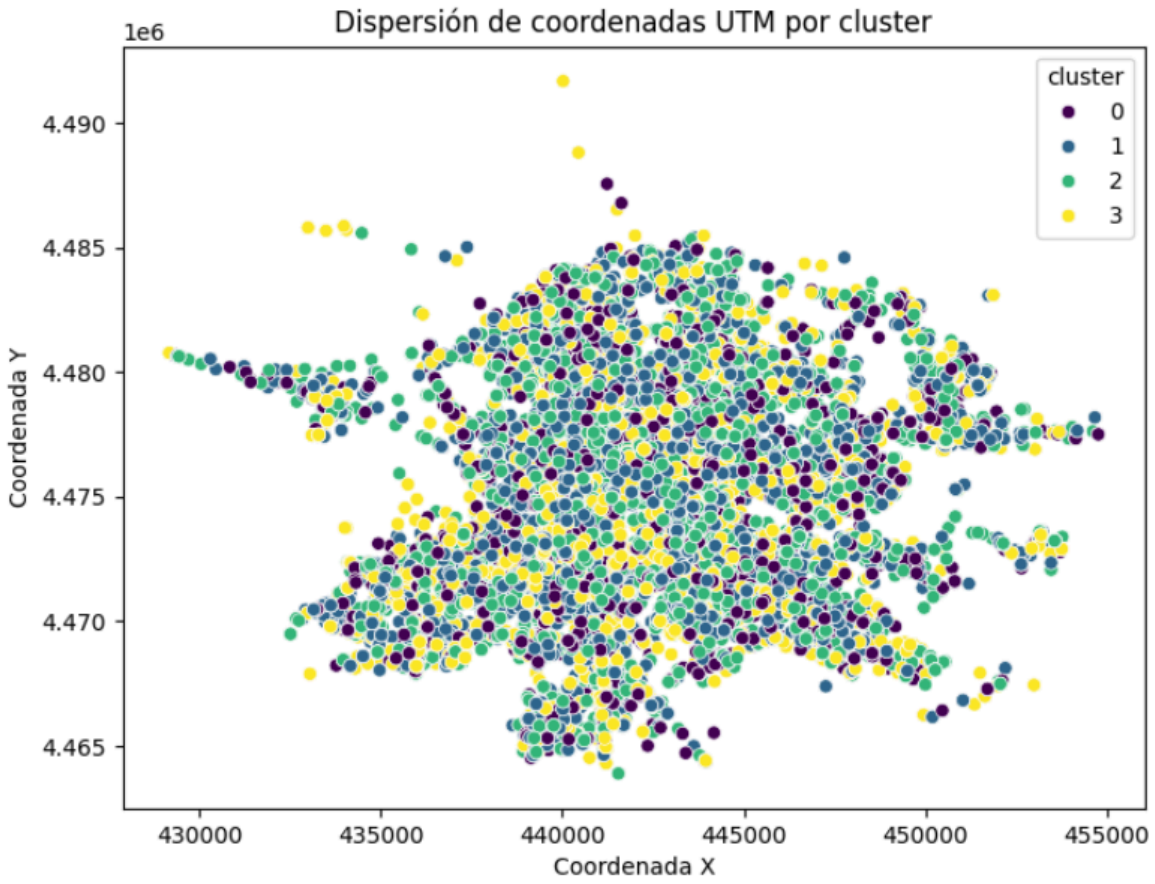
Estudiamos columna por columna la distribución de las variables en los clusters para ver si conseguimos encontrar relaciones de agrupación.



Vemos cómo el sexo se distribuye de una manera muy equitativa en todos los clusters. Vamos a ir probando con el resto de características.

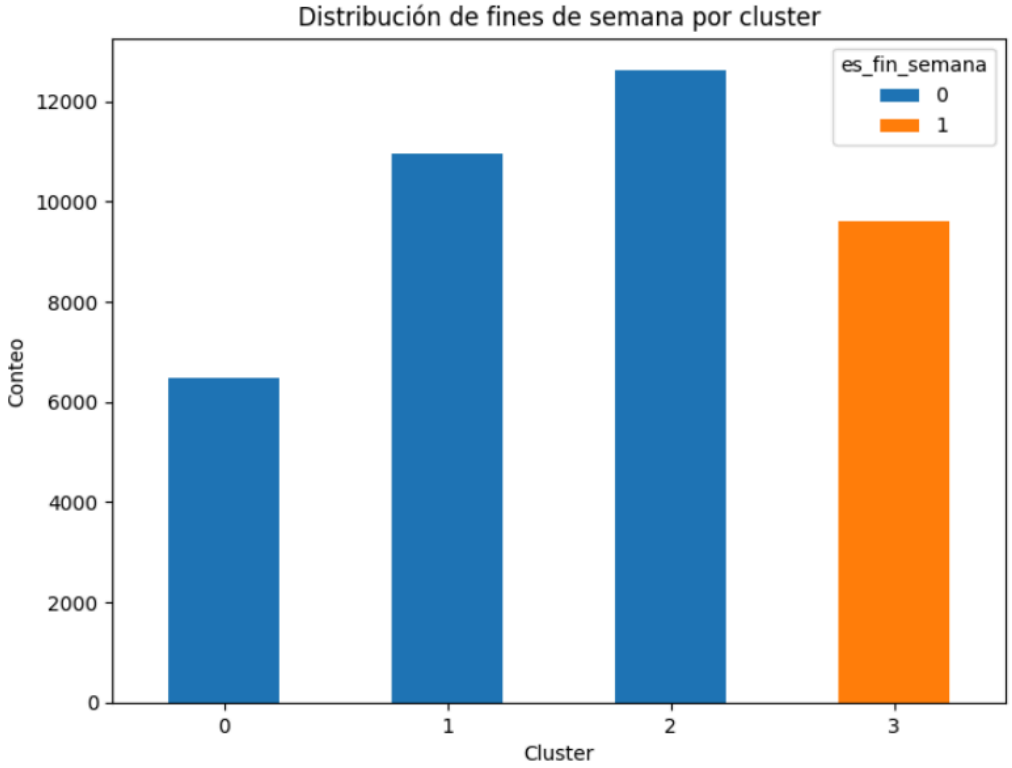


Esta variable también está distribuida por todos los clusters.



Hemos graficado un mapa con las coordenadas para ver si se agrupan siguiendo alguna relación geográfica pero no es el caso.
Seguimos haciendo un estudio a fondo con muchas columnas hasta que encontramos lo siguiente:

```
Distribución en porcentaje de fines de semana por cluster:
es_fin_semana      0      1
cluster
0              100.0    0.0
1              100.0    0.0
2              100.0    0.0
3                0.0  100.0
```

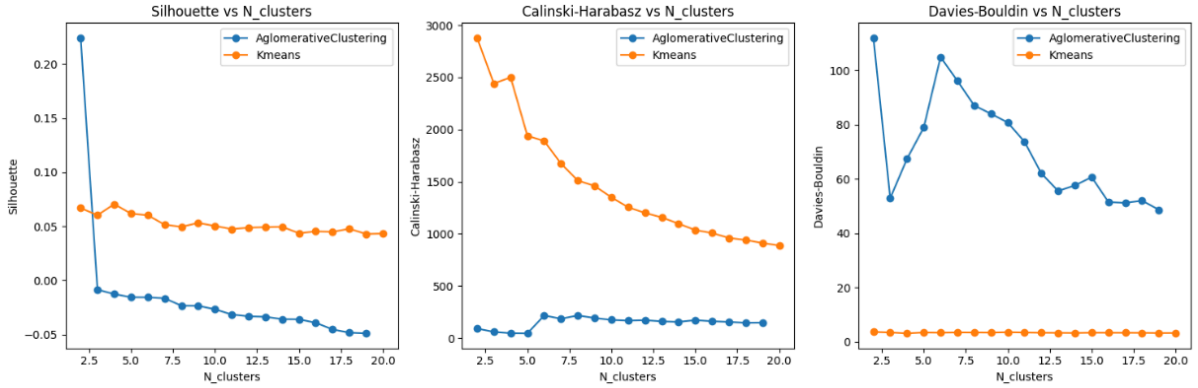


Aquí hemos encontrado una variable que separa uno de los clusters. Vemos como en el cluster 3 todos los datos son de accidentes producidos en fines de semana mientras que en el resto de clusters no encontramos ninguno.

Las relaciones que Kmeans utiliza para separar los datos son complejas y algo complicadas de analizar ya que trabaja con todas las columnas a la vez y hace cálculos con todas ellas. Sin embargo, hemos logrado encontrar la variable que diferencia a uno de los clusters por completo.

Tarea 3.4

En este apartado, importamos los resultados de nuestro algoritmo jerárquico para realizar mejor las comparaciones. Ahora, vamos a realizar un kmeans de 2 a 20 clusters para compararlo con los mismos datos con los que creamos el DataFrame jerárquico. Analizamos con los mismos tres índices y almacenamos los resultados.



Analizando las gráficas, vemos resultados muy diferentes en función del índice utilizado.

Según Silhouette:
La agrupación con AglomerativeClustering de 2 clusters es la mejor con diferencia obteniendo los mejores resultados. Sin embargo, con el resto de valores de n_clusters el kmeans nos ofrece mejores resultados que aparentemente logran mantenerse bastante elevados. Por otro lado, para cualquier valor de n_clusters distinto de 2, el algoritmo jerárquico tiene métricas de Silhouette negativas.

Según Calinski-Harabasz:
La agrupación mediante Kmeans supera con creces en todos los casos estudiados al clustering jerárquico. La mejor agrupación se obtiene con dos clusters, mientras que la mejor agrupación del algoritmo jerárquico, con 6.

Según Davies-Bouldin:
El K means domina la métrica en cada uno de los diferentes valores de n_clusters.

Teniendo todo esto en cuenta, nuestra conclusión es que si queremos hacer dos clusters, la mejor opción es utilizar AglomerativeClustering, ya que Silhouette es el índice de evaluación más fiable y la diferencia con el resto de valores es demasiado grande como para obviarla. Sin embargo, si queremos agrupar nuestros datos con cualquier otro valor de N_clusters, lo mejor es utilizar kmeans. Este algoritmo domina al algoritmo jerárquico en casi todos los aspectos y seguramente con dos clusters también haría un buen trabajo, pero creemos que la mejor opción es utilizarlo cuando N_clusters != 2.

Tarea 4.1

El algoritmo de DBSCAN no funciona muy bien para datasets con mucha densidad y con alta dimensionalidad. El dataset con el que se está trabajando tenía 100 columnas numéricas, por tanto fue necesario aplicar primeramente un PCA, para seleccionar las componentes que garantizaran que se mantenía el 95% de la variabilidad. Lo cuál reducía notablemente la dimensionalidad de los datos.

Previo a aplicar PCA, es necesario normalizar los datos, de forma que se aplicó MinMaxScaler, pues ofrecía una mejor separación de los clusters que StandardScaler.

Tarea 4.2

Determinar eps

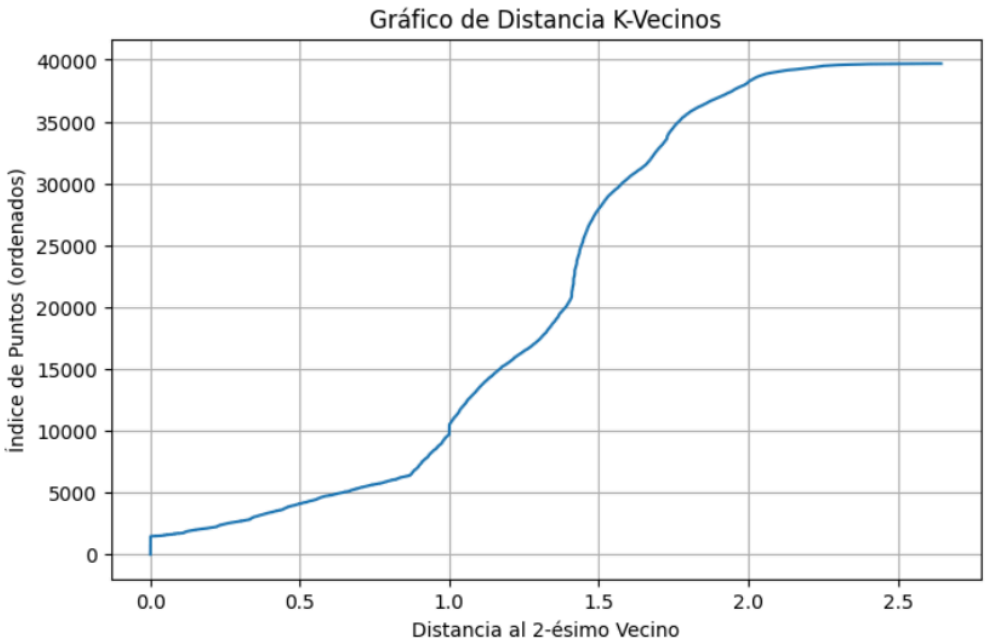
DBSCAN es un algoritmo que agrupa puntos en función de su densidad en el espacio, pero su efectividad depende en gran medida de dos parámetros:

- **eps**: Define el radio máximo dentro del cual se buscarán vecinos para formar un cluster.
- **min_samples**: Establece el número mínimo de puntos necesarios dentro del radio eps para considerar que un punto pertenece a un cluster.

Una elección incorrecta de estos parámetros puede resultar en clusters mal definidos o en un exceso de puntos clasificados como ruido. Por esta razón, es fundamental determinar valores adecuados para ambos, especialmente para **eps.**

Para encontrar un valor apropiado para eps, se utilizó KNN. La idea es que los puntos en regiones densas tendrán distancias pequeñas a sus vecinos más cercanos, mientras que los puntos más aislados o en ruido tendrán distancias significativamente mayores. Graficar estas distancias ayuda a identificar un "codo" que separa las regiones densas del ruido.

El gráfico de KNN que se obtuvo fue:



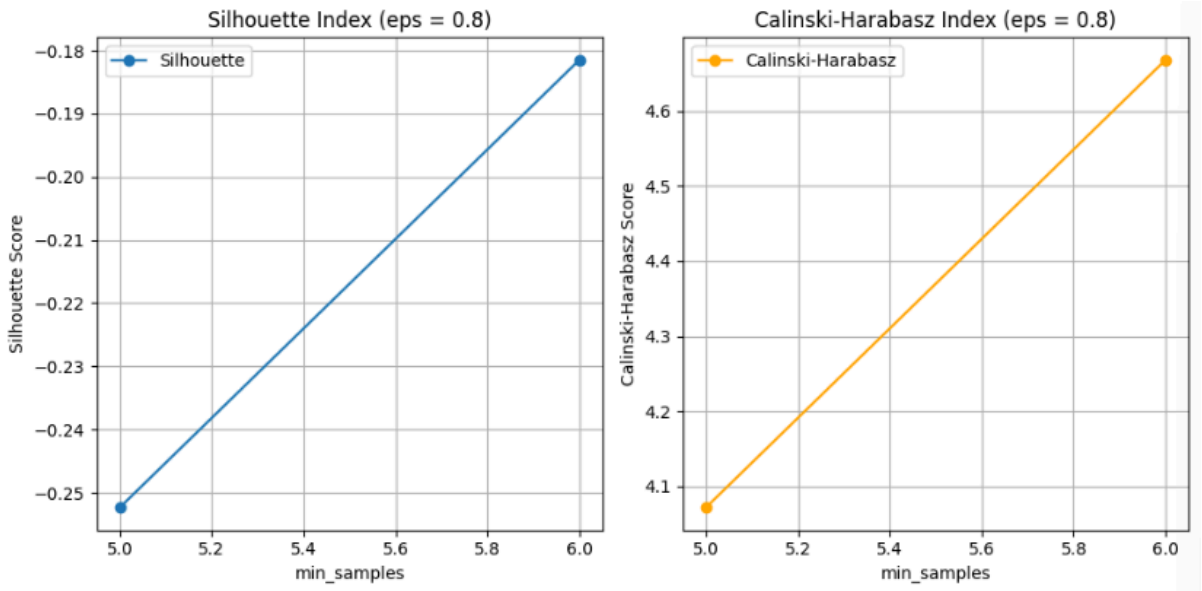
Se puede ver que aproximádamente para eps=0.8 hay un codo, donde el gráfico empieza a crecer exponencialmente.
Por tanto se tomará el valor de eps=0.8, al menos como hipótesis inicial.

Determinar min_samples

Una vez escogido el valor de eps, determinar min_samples es más simple, pues simplemente es necesario probar con diferentes valores y después analizar cuál es mejor.

Después de haber probado con varias combinaciones de valores para eps=0.8, solamente min_samples = 5 y min_samples = 6 cumplían con los requisitos que eran los siguientes:

- Separar los datos en al menos 2 clusters
- Separar los datos en menos de 100 clusters (ya que no tiene sentido para este problema en concreto).



Se utilizaron los índices de Silhouette y Calinski-Harabasz para evaluar la calidad del cluster con eps=0.8 y los valores 5 y 6 de min_samples.
Como se puede apreciar en las gráficas min_samples=6 ofrecía mejores resultados.

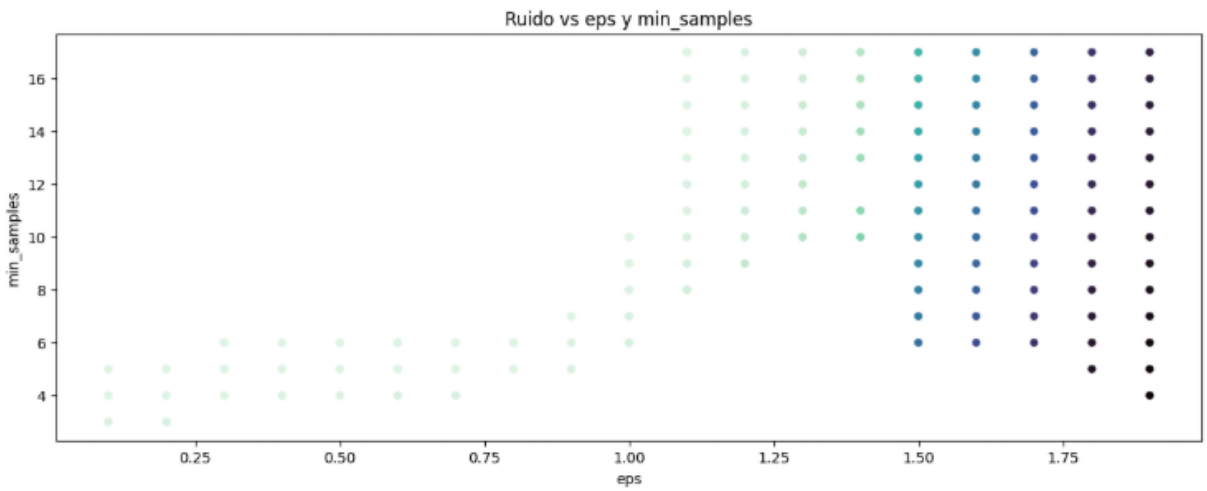
Al aplicar dichos parámetros con DBSCAN se obtienen 7 clusters.

Tarea 4.3

Para evaluar la calidad del clustering con diferentes valores de eps y min_samples, se plotearon las siguientes gráficas:

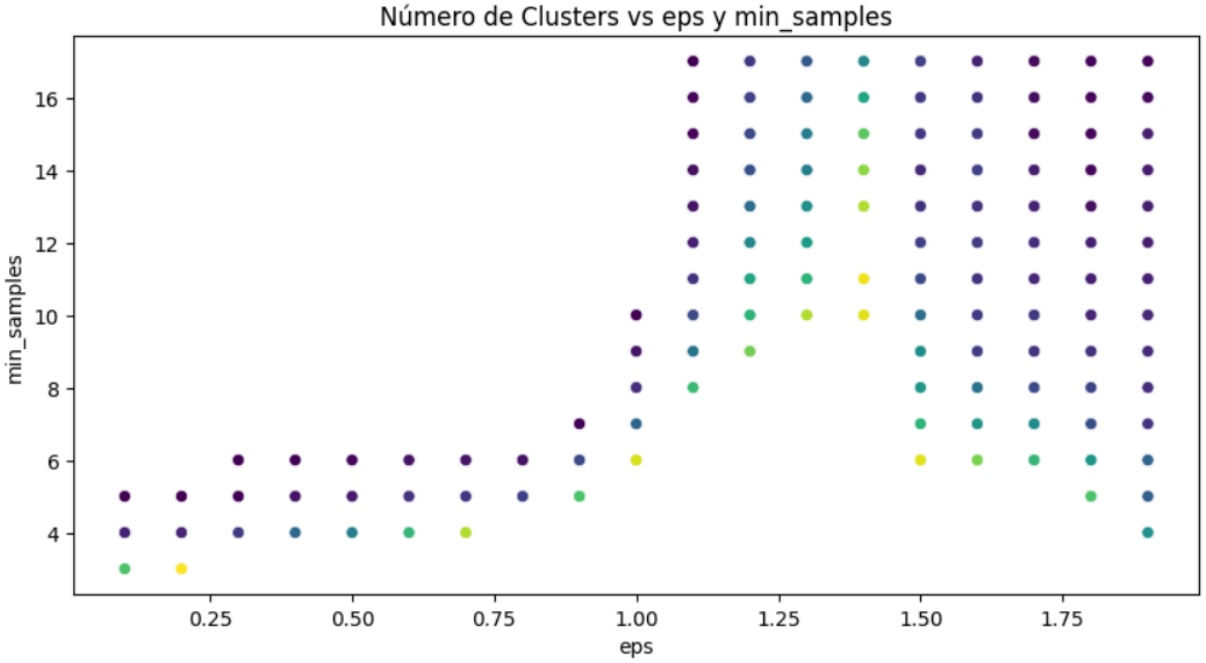
Los puntos de colores que aparecen en esta gráfica representan el número de puntos que son ruido del dataset, es decir, aquellos puntos que no pudieron ser clasificados en ningún cluster. Idealmente el número de puntos de ruido no debería ser demasiado alto, ya que esto significaría que el algoritmo no ha sido capaz de diferenciar los puntos correctamente. Y si el número de puntos es demasiado bajo, tal vez esté indicando que el valor de eps es demasiado alto y por tanto está mezclando puntos muy alejados entre sí como similares.

Mientras más oscuro sea el punto menos ruido hay para esa combinación de parámetros y mientras más claro sea el punto, más ruido. Yendo los valores desde 4100 hasta 39680 puntos de ruido.



Se puede apreciar como a medida que se va aumentando el valor de eps el numero de puntos de ruido va disminuyendo. Esto ocurre debido a que el radio para considerar vecinos se expande, lo que permite que más puntos sean incluidos en clusters en lugar de ser clasificados como ruido. Sin embargo, solamente analizando esta gráfica no es posible determinar cuáles son los mejores valores de eps y min_samples ya que también depende del número de clusters.

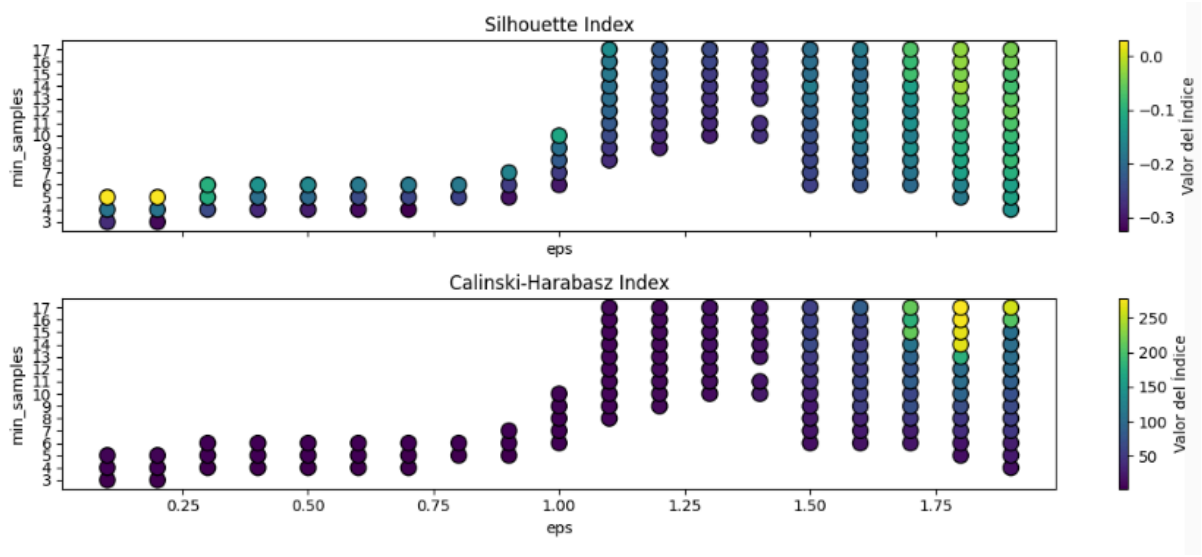
La escala para esta gráfica es similar, donde los puntos más oscuros representan menor cantidad de clusters y los puntos más claros representan más cantidad de clusters. El rango varía desde 2 a 99 clusters.



Observando el gráfico se puede llegar a la conclusión de que valores razonables para eps y min_samples son aquellos que permitan una cantidad razonable de clusters pero sin demasiado ruido. Esos valores se pueden encontrar encontrar a la derecha del gráfico. Aproximádamente en 1.8 de eps y min_samples en 10 o 12.

Estos valores no son los que se habían escogido anteriormente, debido a que los métodos previamente utilizados no son tan precisos.
Aunque si se observa de nuevo la gráfica de KNN se puede apreciar un codo en el valor 1.8 de eps.

Tarea 4.4

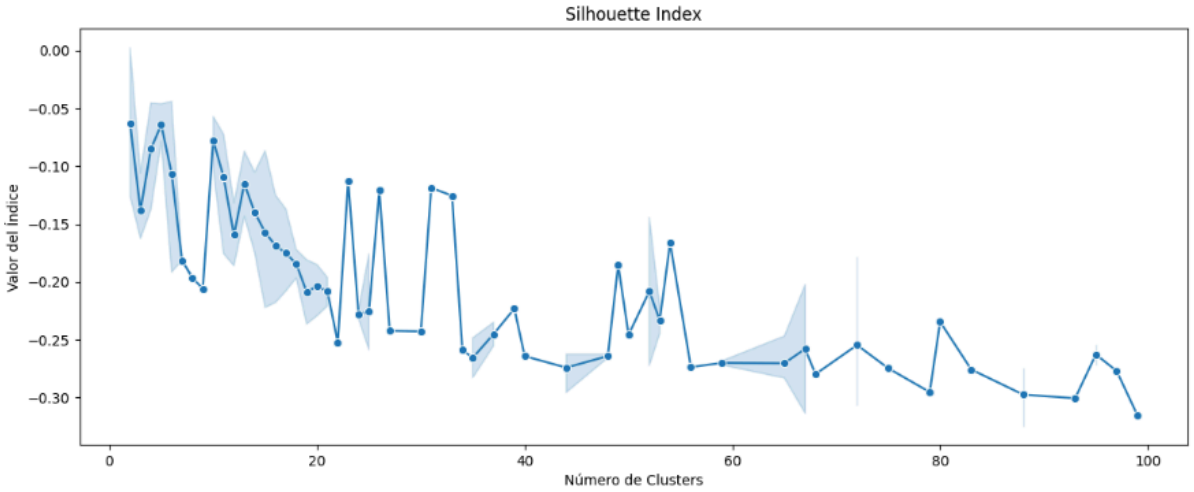


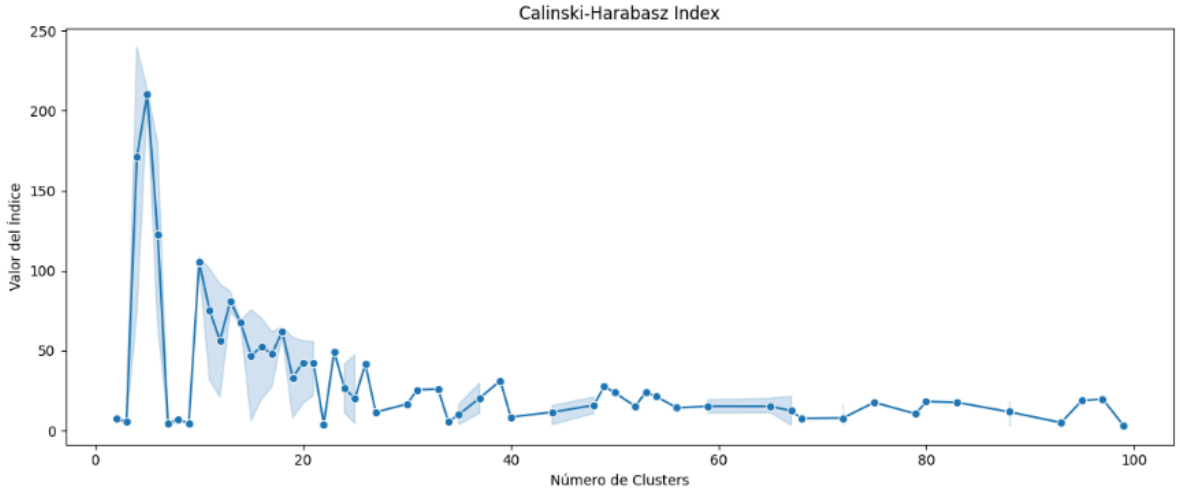
Observando las gráficas de los valores de los índices, se puede apreciar que efectivamente los valores de eps y min_samples que mejor resultados dan son: eps=1.8 aproximadamente y min_samples un poco más alto, en torno a 14 o 15.

Tarea 4.5

Técnicas como el método del codo o el gap statistic no se pueden aplicar directamente a DBSCAN porque este algoritmo no requiere un número fijo de clusters como entrada, sino que los determina automáticamente en función de la densidad de los datos (parámetros eps y min_samples). Además, DBSCAN permite la existencia de ruido (puntos no asignados a ningún cluster), lo que dificulta el uso de estas técnicas, ya que están diseñadas para algoritmos que siempre agrupan todos los puntos, como k-means, y suponen que todos los datos pertenecen a un cluster definido.

Es por ello que para determinar el número óptimo de clusters, se utilizaron como medida los índices anteriores.





Se puede ver como hay varias oscilaciones en los gráficos, proponiendo varios valores de clusters como correctos. Sin embargo, Podemos ver que hay un valor bastante alto en ambas gráficas para 4 clusters, lo cual encaja bastante bien para los valores de escogidos de eps y min_samples.
Por tanto, se determina que el número óptimo de clusters es 4.

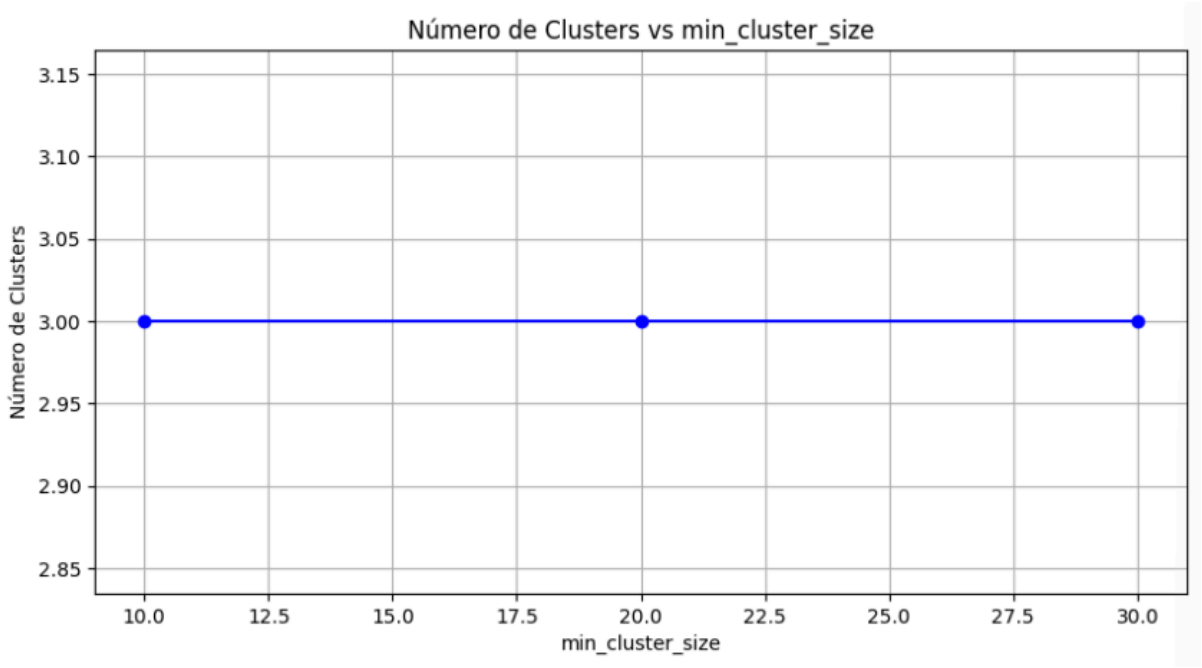
Tarea 5.1

Para aplicar HDBSCAN, al igual que pasa con DBSCAN es recomendable no solo escalar los datos para evitar que haya outliers que puedan hacer que el modelo se sesgue, si no también aplicar PCA para reducir la dimensionalidad, pues HDBSCAN también funciona bastante mal con datos con mucha densidad.

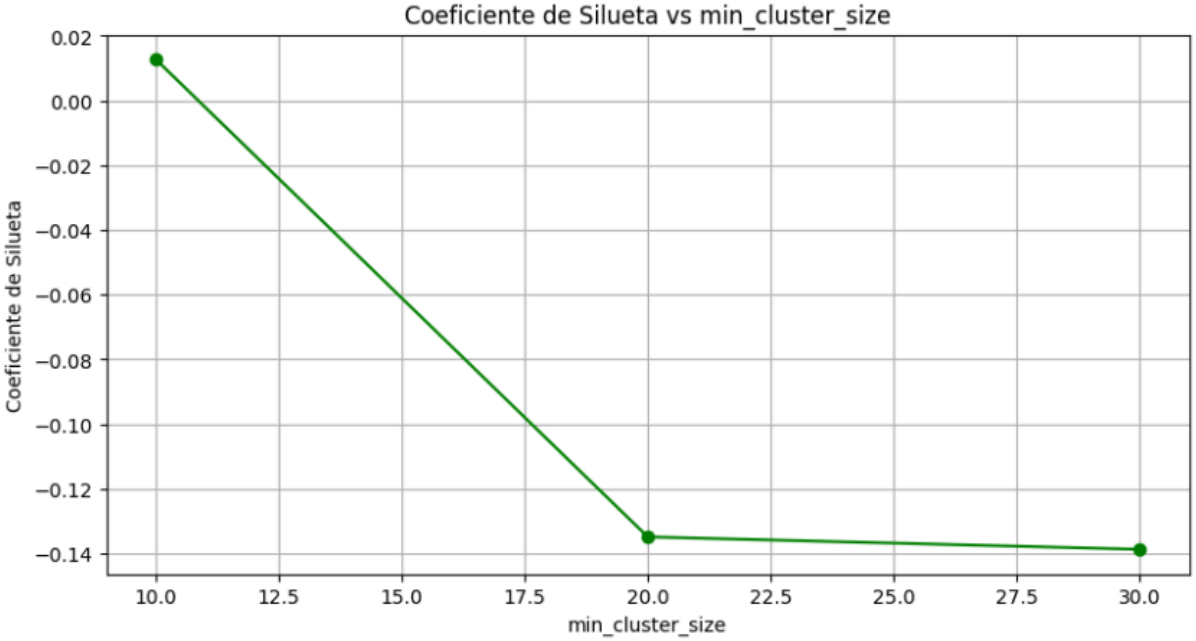
En HDBSCAN, se puede modificar el el parámetro min_cluter_size. Este parámetro define el tamaño mínimo que debe tener un grupo para ser considerado un cluster válido. Los conjuntos más pequeños se etiquetan como ruido. Esto controla qué tan compacto o significativo debe ser un cluster.

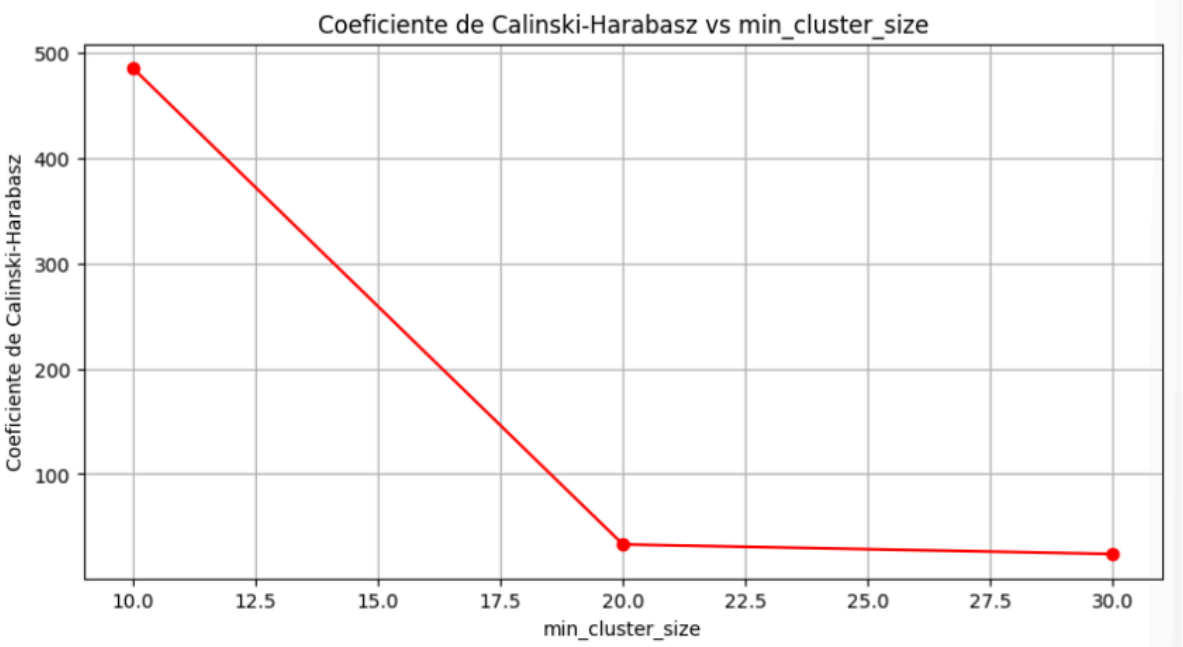
Para determinar qué valor era el óptimo para este parámetro y nuestros datos, se probaron diferentes valores en un rango de entre 10 y 500 con saltos de 10.

Los resultados fueron los siguientes:



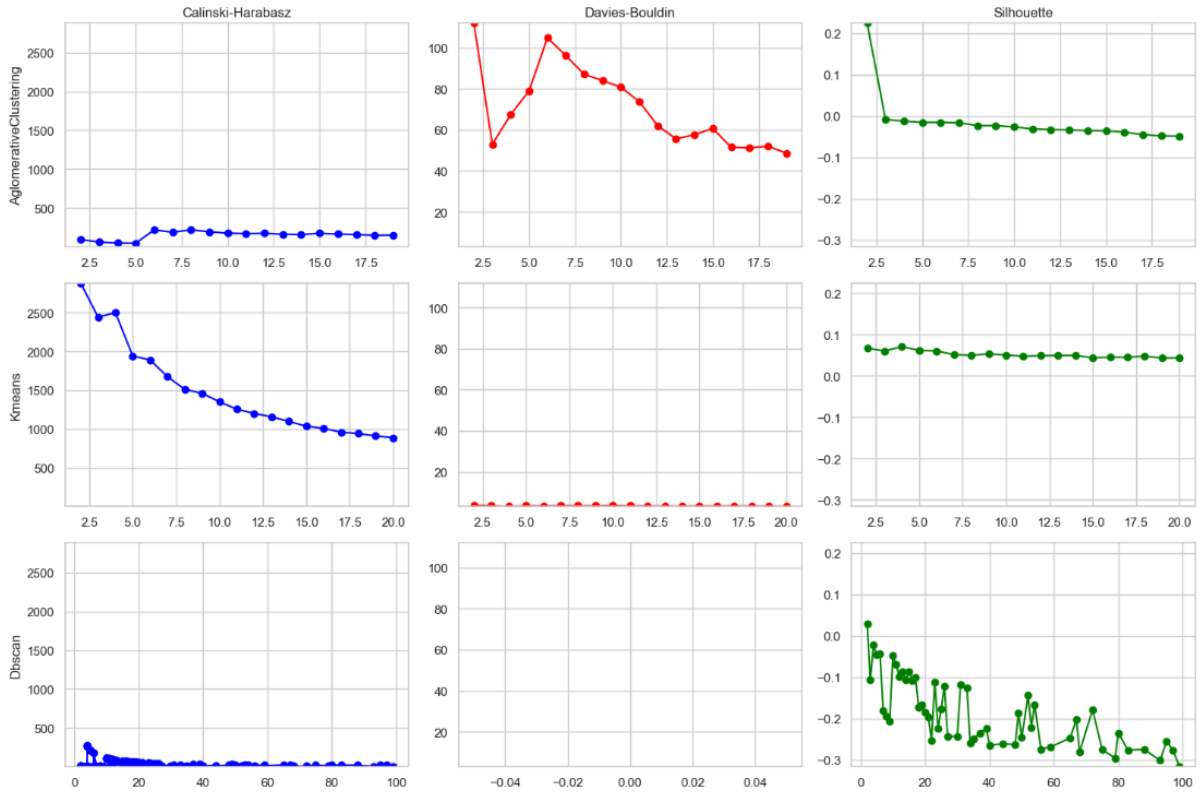
Se aprecia que el min_cluster_size solamente abarca desde el 10 al 30. Esto se debe a que solamente fueron considerados como válidos aquellos valores de min_cluster_size en los que se crearan 2 clusters o más. Ya que tener un solo cluster no aporta información sobre como se diferencian los datos. Para todos los valores posibles de min_cluster_size el número de clusters seguía siendo 3.





Si evaluamos los índices de Silueta y de Calinski-Harabasz con respecto al parámetro min_cluster_size se puede apreciar una disminucion de los valores con min_cluster_size 20 y 30, indicando que el tamaño del cluster no es tan grande.

Comparación con otros algoritmos



El algoritmo de Dbscan no funciona demasiado bien para este dataset, al menos según los índices de los que se disponen. Sus valores en todos los índices son significativamente peores que los del resto de algoritmos. Esto podría deberse a la naturaleza de Dbscan, que tiende a funcionar mejor en datasets con estructuras claras y densas, mientras que en este caso podría no estar captando adecuadamente las relaciones entre los puntos debido a la alta dimensionalidad o a la falta de densidad uniforme en los clústeres.

El algoritmo que mejor funciona es el de Kmeans, ya que sus valores para todos los índices son consistentes y bastante buenos. Esto indica que Kmeans logra encontrar separaciones claras entre los clústeres, lo que sugiere que el dataset puede tener grupos bien definidos que se adaptan al criterio de minimización de la varianza interna que utiliza este algoritmo.

En comparación con el HDBSCAN, los dos primeros algoritmos que se muestran son mejores. El HDBSCAN, al igual que el DBSCAN, no presenta resultados satisfactorios, lo cual podría estar relacionado con la alta dimensionalidad del dataset. Algoritmos basados en densidad como HDBSCAN y DBSCAN suelen enfrentarse a dificultades en espacios de alta dimensionalidad debido al fenómeno de la "maldición de la dimensionalidad", que reduce la eficacia de las métricas de distancia y densidad.