Inteligencia de Negocio (2017-2018)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Práctica 2

Segmentación para Análisis Empresarial

javierbejarmendez@correo.ugr.es
Grupo 1, Vier. 9:30-11:30

45337539-p

Javier Béjar Méndez

December 19, 2017

Contents

List of Figures

List of Tables

1 Introducción

Veremos el uso de técnicas de aprendizaje no supervisado para análisis empresarial. Se trabajará con un conjunto de datos sobre el que se aplicarán distintos algoritmos de agrupamiento (clustering). A la luz de los resultados obtenidos se crearán informes y análisis lo suficientemente profundos. El estudio se ha realizado en python[17] apoyandonos en las librerias pandas[14], matplotlib[13], sklearn[3] y seaborn[15].

Se ha estructurado el estudio de la siguiente forma, primero explicaremos el problema ha abordar; con la base de datos elegida, se explicarán los casos de estudio específicos, y los algoritmos elegidos. Después expondremos los resultados obtenidos para cada caso de estudio y analizaremos dichos resultados, con el conocimiento extraido consecuentemente, y veremos el impacto que tiene la parametrización de algunos algoritmos en cada caso de estudio.

1.1 Descripción del Problema

Una compañíaa aseguradora quiere comprender mejor las dinámicas en accidentes de tráfico en España. Para ello, a partir de diversas variables que caracterizan el accidente, se pretende encontrar grupos de accidentes similares y relaciones de causalidad que expliquen tipos y gravedad de los accidentes. Para ello se cuenta con los datos publicados por la Dirección General de Tráfico (DGT)[16], que incluye información desagregada (microdatos) de más de 30 variables entre los años 2008 y 2015. En esta práctica, nos centraremos en los datos para el año 2013 (89.519 accidentes). En la web de la asignatura se incluye el conjunto de datos[2] —procesado a partir de la fuente original— sobre el que se trabajará en esta práctica.

1.2 Casos de Estudio

Se ha decidido realizar tres casos de estudio, en los que se pretende estudiar lo siguiente:

- Caso 1, Analizar la gravedad de los accidentes de colisión de dos o más vehículos.
- Caso 2, Estudiar la gravedad en los accidentes por las salidas de vía los fines de semana, relacionados con la hora.

1.3 Algoritmos Elegidos

Se han elegido 5 algoritmos para realizar el análisis de todos los casos de estudio, utilizaremos los algoritmos implementados por la librería Sklearn[4], en concreto usaremos los siguientes:

- K-means[7]
- Agglomerative Clustering[5]
- Birch[6]

- MeanShift[8]
- MiniBatchKMeans[9]

1.4 Medidas de Interés

A la hora de analizar los resultados nos centraremos en las medidas *Calinski-Harabaz Index*[1] y *Silhouette Coefficient*[12] para el rendimiento de los algoritmos, además del tiemplo empleado en el clustering. Para analizar los datos y clusters obtenidos por los algoritmos nos apoyaremos en dendogramas y mapas de calor, generados mediante la función *clustermap*[10] y en los ScatterMatrix generados mediante la función *pairplot*[11].

1.5 Leer, Profesor

La memoria es excesivamente extensa debido a que cada scatterMatrix es una página, el contenido importante se encuentra en el análisis inicial y el análisis de la segmentación.

2 Caso 1

En este caso vamos a analizar la gravedad de los accidentes en colisiones múltiples de vehículos, para ello nos centraremos en los accidentes en los que estén implicados 2 o más vehículos. Centrandos en los atributos de mortalidad, heridos leves, heridos graves, víctimas totales y número de vehículos implicados. Con esto pretendemos encontrar grupos o patrones en los que se vea reflejado que accidentes tienen mayor o menor gravedad en función de los atributos mencionados. La segmentación realizada de la base de datos para este caso de estudio la podemos observar en la siguiente figura:

```
#preparación de datos------------------------------------------------------------
accidentes = pd.read_csv('accidentes_2013.csv')

subset = accidentes

#seleccionar accidentes de tipo 'colisiÃ³n de vehÃculos'
subset = subset[subset['TIPO_ACCIDENTE'].str.contains('Colisión de vehículos')]

#seleccionar variables de interÃes para clustering
var_interes = ['TOT_VICTIMAS', 'TOT_MUERTOS', 'TOT_HERIDOS_GRAVES', 'TOT_HERIDOS_LEVES',
               'TOT_VEHICULOS_IMPLICADOS']
X = subset[var_interes]
```

Figure 2.1: Preparación de datos para el caso de estudio 1.

2.1 Análisis Inicial

Se han obtenido 3 dendogramas con mapas de calor asociados, mediante el algoritmo ward y con un sampleado de 3 000 muestras de la base de datos descrita anteriormente. Primero generamos el clustermap simple:

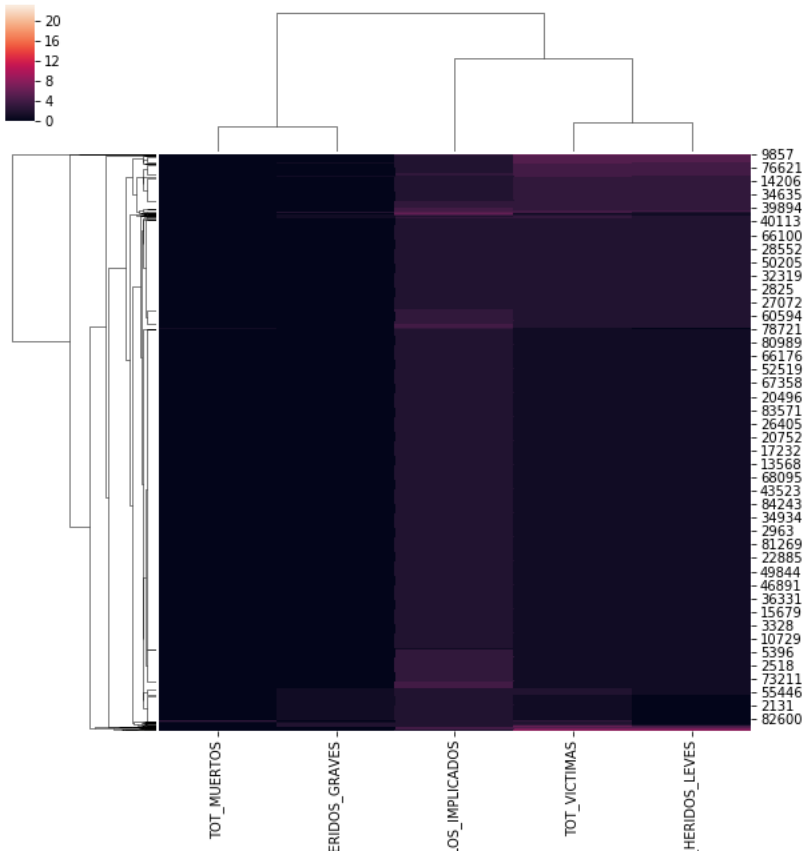

Figure 2.2: ClusterHeatMap obtenido para el caso 1.

A priori no podemos obtener mucha información del gráfico, podemos aventurar que a mayor número de víctimas hay mayor número de heridos leves, algo intuible desde la propia lógica, esto se observa en que el color de dichas variables varia equitativamente en su mayoría. Para confirmarlo podemos observar el siguiente clustermap, en el que el mapa de calor se obtiene a partir de la correlación:

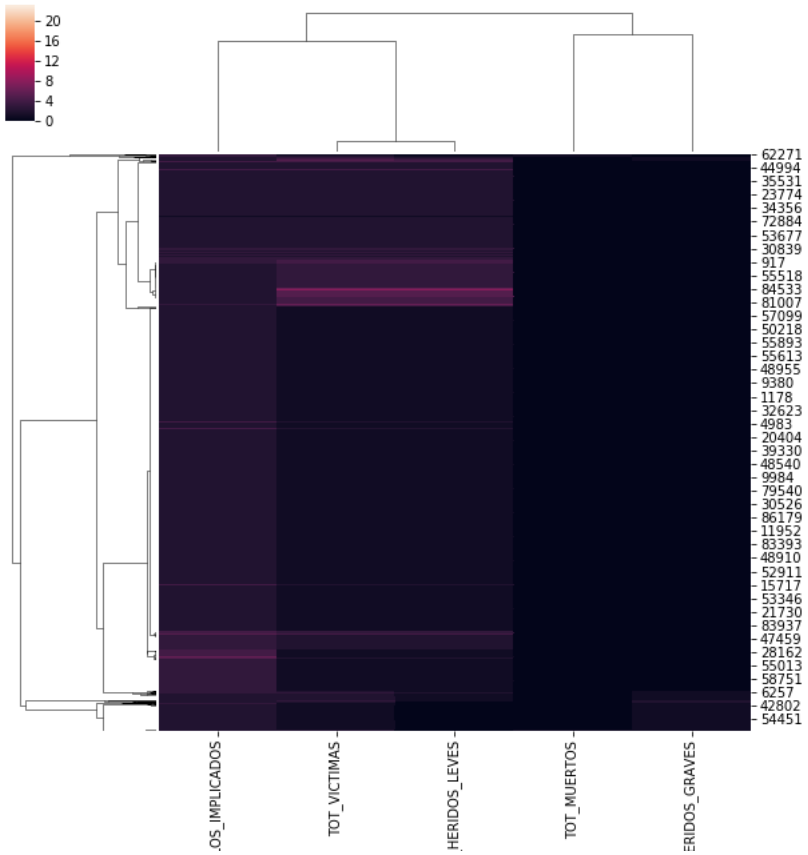

Figure 2.3: ClusterHeatMap utilizando la correlación como medida de distancia obtenido para el caso 1.

Observamos que dichas variables tienen una gran correlación. No podemos decir más sobre este gráfico, aparte de que todas las variables estám bastante correlacionadas y se empieza a intuir distintos grupos con distintas propiedades. Para poder observar estos grupos con una mayor fidelidad generamos el siguiente clustermap con los valores normalizados por filas:

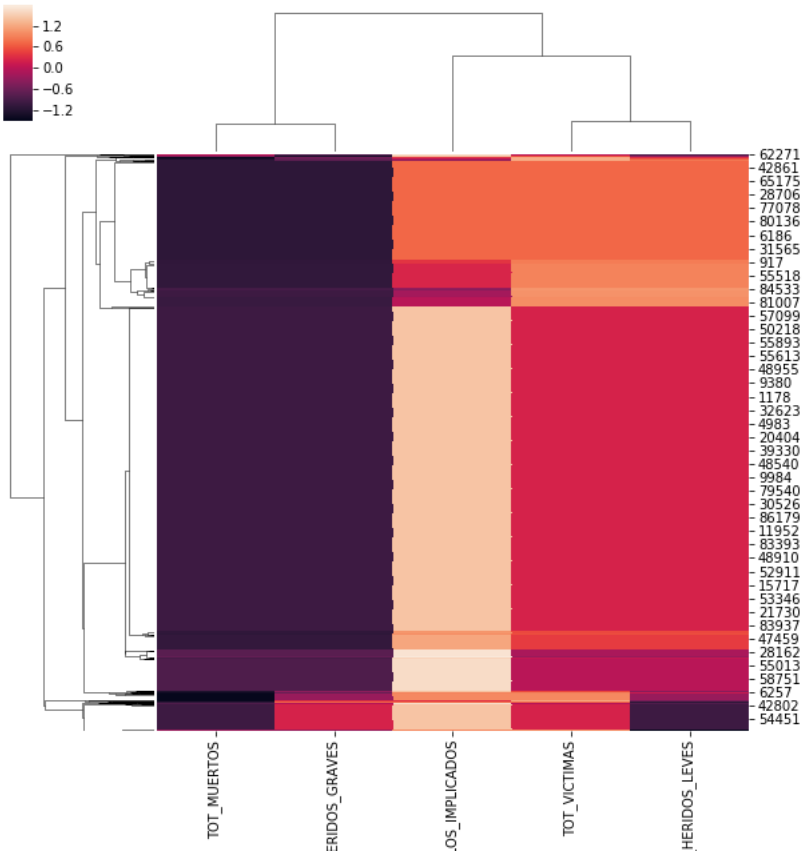

Figure 2.4: ClusterHeatMap normalizando valores por filas para el caso 1.

En este gráfico observamos claramente que cuando hay un número alto de vehículos implicados, y pocas victimas/heridos leves, tenemos una mortalidad mayor y un número de heridos graves mayor.

De todas maneras la generación de estos mapas es costosa, y para ello se ha sampleado 3 000 muestras aleatorias de las 50 000 muestras que tiene el subconjunto de datos aproximadamente, por lo que no refleja una información lo suficientemente precisa. En las siguientes secciones podremos observar si estas primeras declaraciones son verdaderas y concretar y concisar mejor.

2.2 Kmeans

Este algoritmo se ha definido mediante el siguiente código:

$$kmeans = KMeans(init =' k-means++', n_{clusters} = n_{Clust}, n_{init} = 5)$$

Y analizaremos los resultados para los números de clusters iguales a 2, 4 y 8. Este algoritmo se ha lanzado con el subconjunto de datos completo(49280 muestras), ya que es rápido. En las siguientes tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
KMeans	62847.198	0.725	0.07224	2
KMeans	96112.886	0.811	0.10434	4
KMeans	157434.646	0.884	0.18080	8

Table 2.1: Métricas Kmeans, Caso 1.

Cluster	Población
0	13553
1	35727

Table 2.2: Pobalción clusters Kmeans, Caso 1, número de clusters 2.

Cluster	Población
0	32508
1	9525
2	2764
3	4483

Table 2.3: Pobalción clusters Kmeans, Caso 1, número de clusters 4.

Cluster	Población
0	8760
1	29073
2	2270
3	4417
4	339
5	3250
6	924
7	247

Table 2.4: Pobalción clusters Kmeans, Caso 1, número de clusters 8.

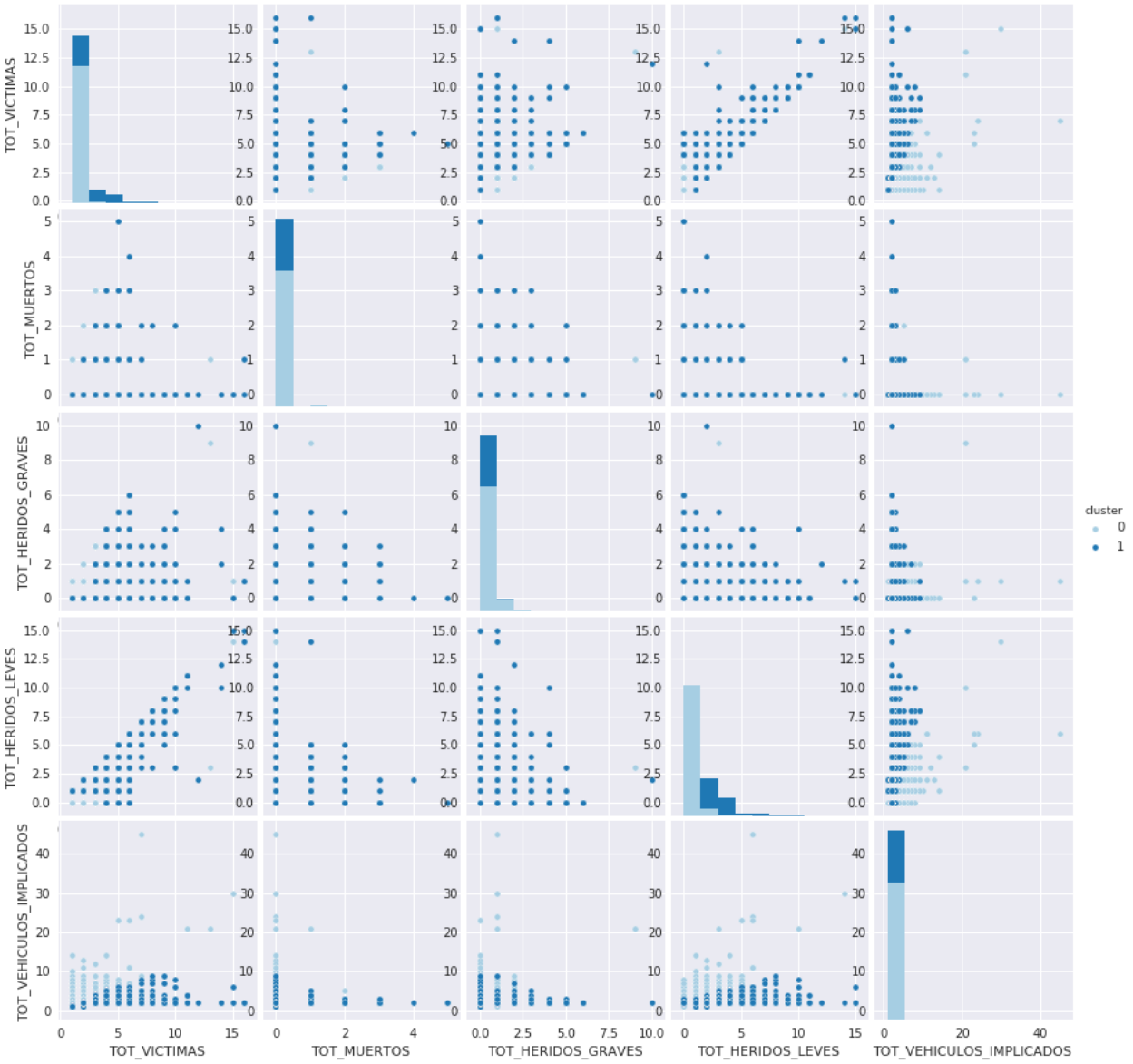

Figure 2.5: ScatterMatrix de Kmeans, caso 1, número de clusters 2.

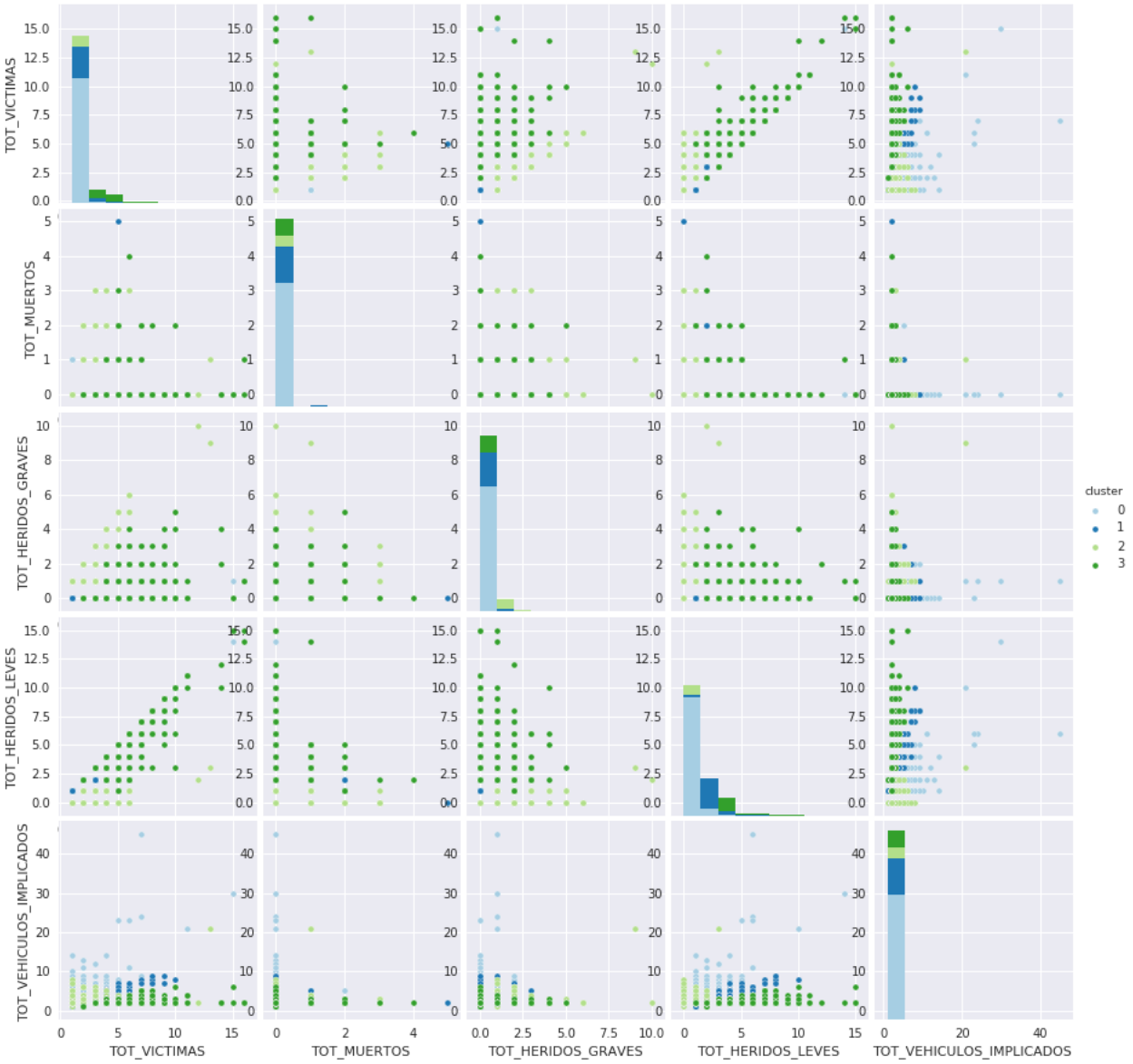

Figure 2.6: ScatterMatrix de Kmeans, caso 1, número de clusters 4.

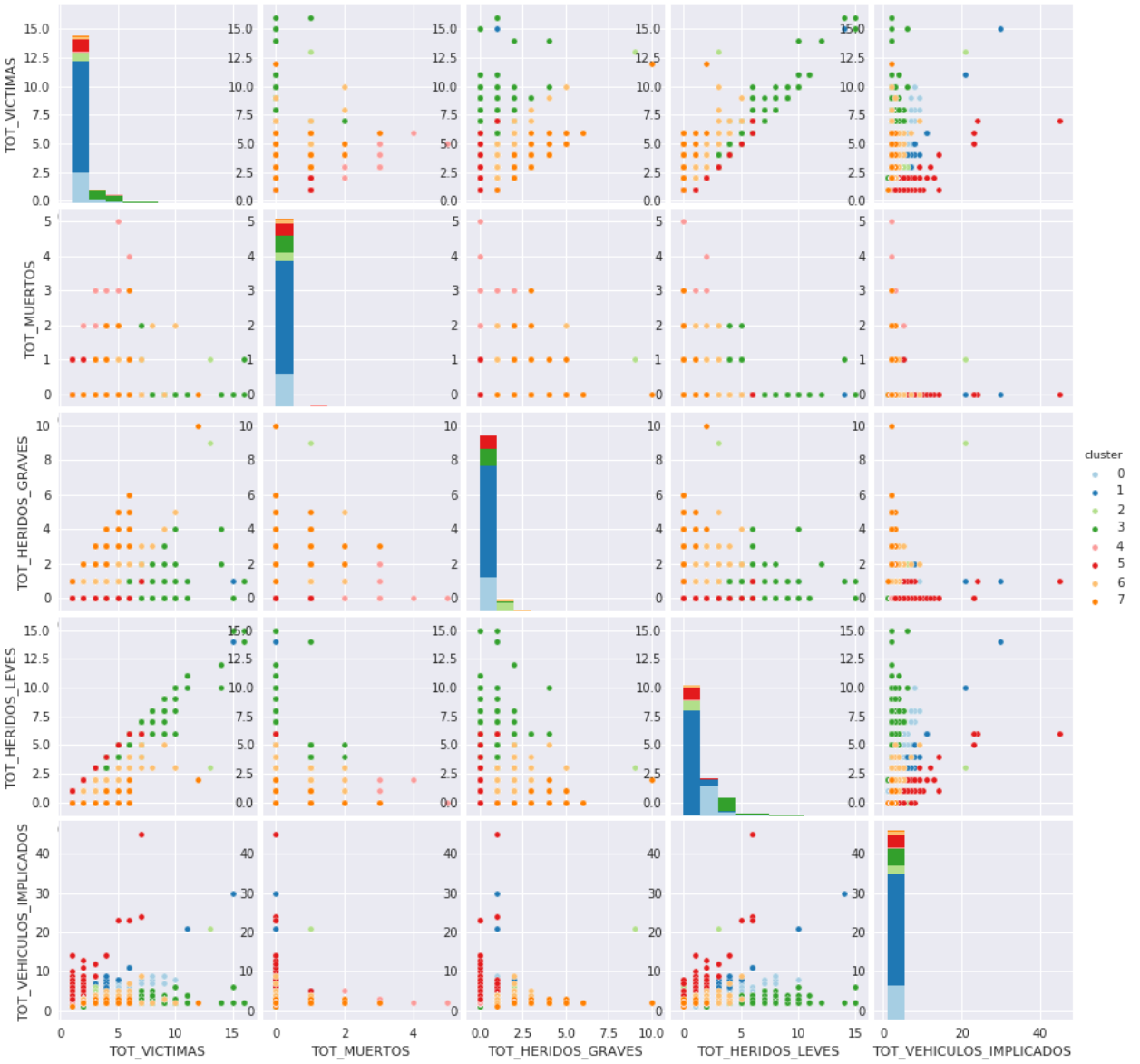

Figure 2.7: ScatterMatrix de Kmeans, caso 1, número de clusters 8.

2.3 Agglomerative Clustering

Este algoritmo se ha definido mediante el siguiente código:

$$Agg = AgglomerativeClustering(n_{clusters} = n_{Clust}, linkage =' ward')$$

Se analizarán los resultados para los números de clusters iguales a 2, 4 y 8. Dados que este algoritmo es más complejo se ha lanzado con un subconjunto de datos de la muestra

sampleado con 20 000 individuos, realizado de la siguiente manera:

```
#Sampleado especificos---------------------------------
#Agglomerative_Clustering
Xward = X.sample(20000, random_state=123456) #20 000
Xward_norm = preprocessing.normalize(Xward, norm='l2')
```

Figure 2.8: Sampleado de datos para el algoritmo Agglomerative Clustering, caso 1.

En las siguientes tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
AggClust	24584.466	0.705	7.88352	2
AggClust	38953.316	0.809	7.95663	4
AggClust	57923.649	0.906	8.46112	8

Table 2.5: Métricas Agglomerative Clustering, Caso 1.

Cluster	Población
0	14722
1	5278

Table 2.6: Pobalción clusters Agglomerative Clustering, Caso 1, número de clusters 2.

Cluster	Población
0	13239
1	1882
2	1483
3	3396

Table 2.7: Pobalción clusters Agglomerative Clustering, Caso 1, número de clusters 4.

Cluster	Población
0	394
1	1882
2	132
3	3396
4	1319
5	957
6	650
7	11270

Table 2.8: Pobalción clusters Agglomerative Clustering, Caso 1, número de clusters 8.

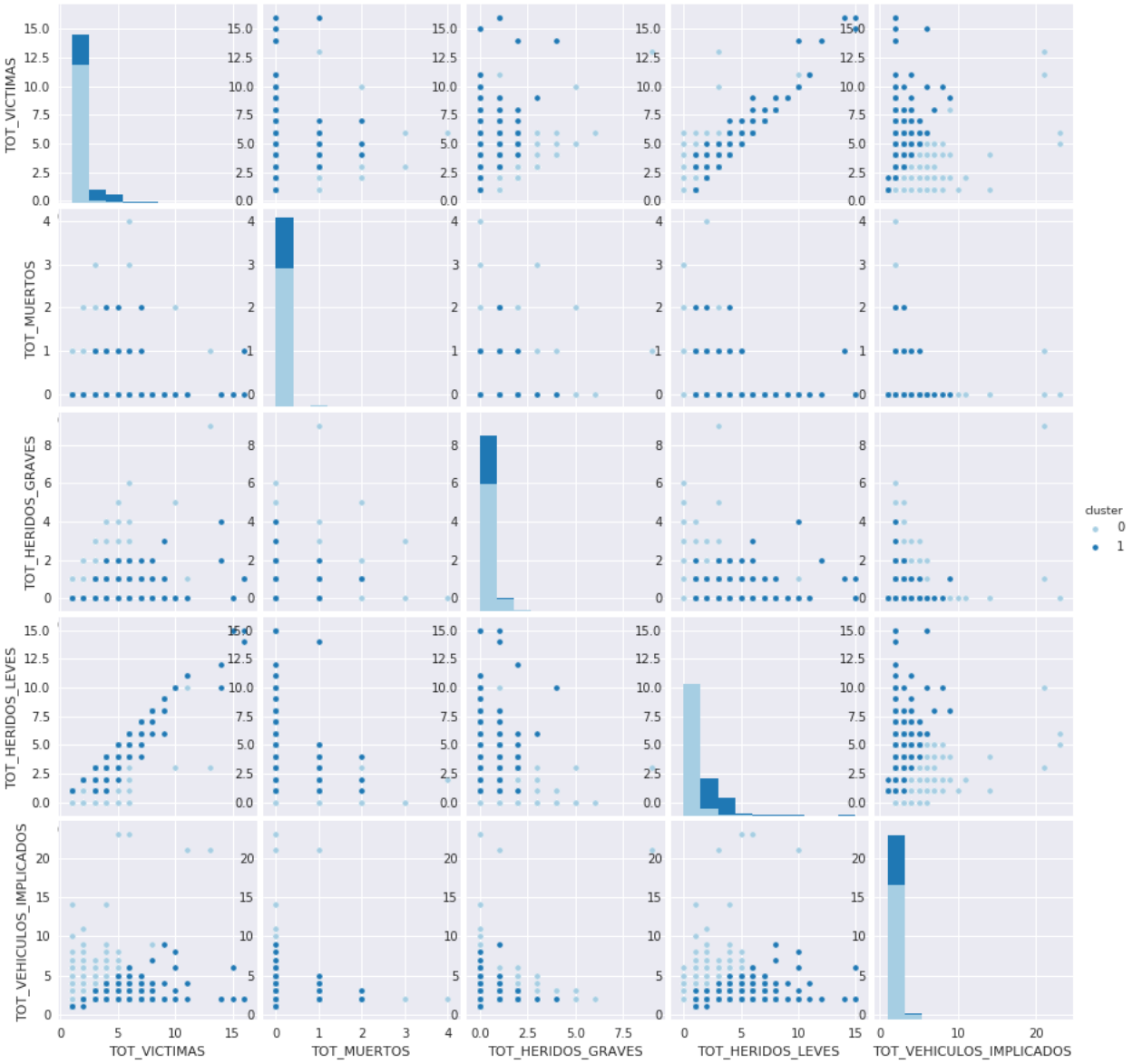

Figure 2.9: ScatterMatrix de Agglomerative Clustering, caso 1, número de clusters 2.

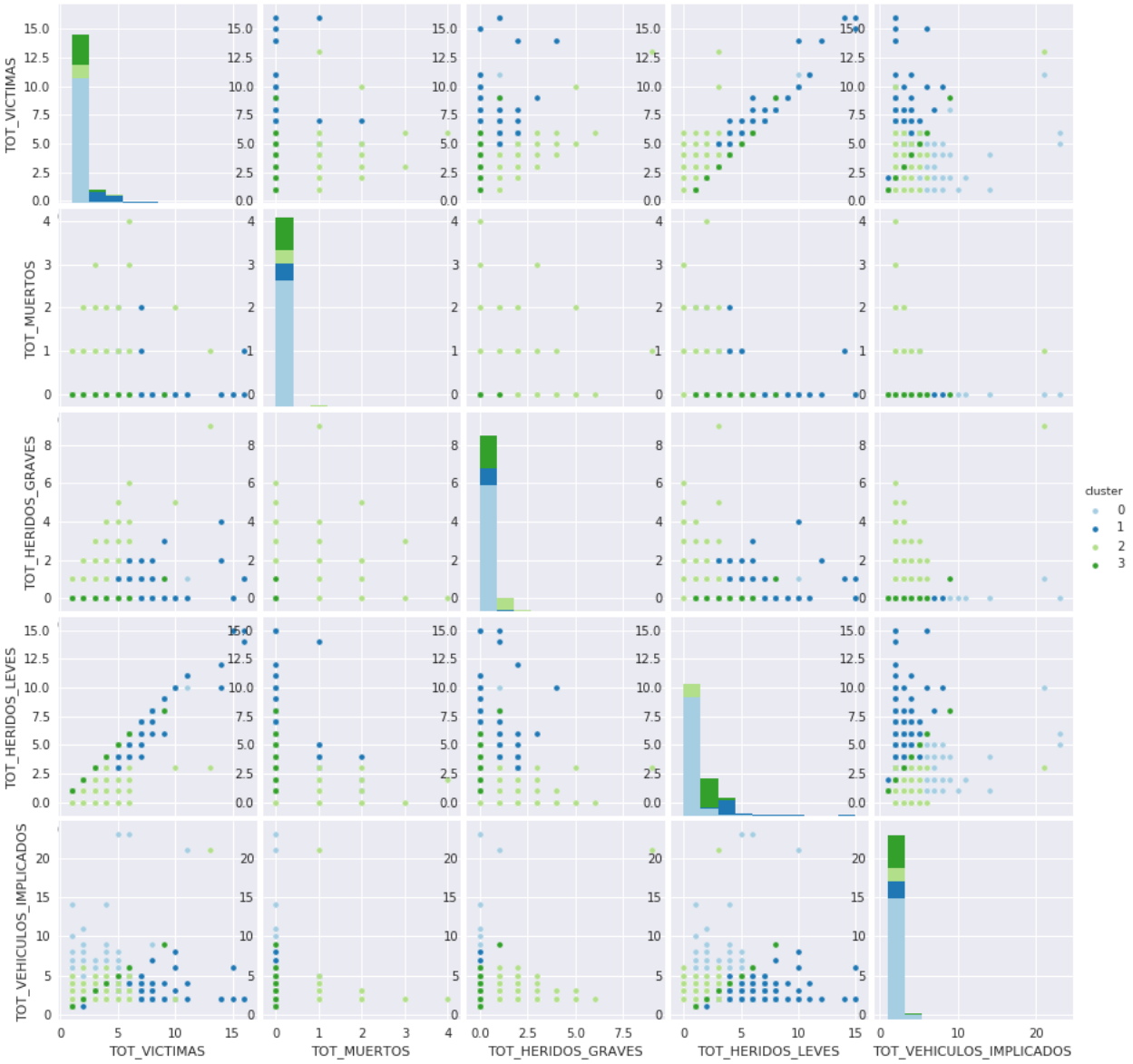

Figure 2.10: ScatterMatrix de Agglomerative Clustering, caso 1, número de clusters 4.

Figure 2.11: ScatterMatrix de Agglomerative Clustering, caso 1, número de clusters 8.

2.4 Birch

Este algoritmo se ha definido mediante el siguiente código para los casos de número de clusters 2 y 4:

$$birch = Birch(n_clusters = n_Clust, threshold = 0.3)$$

Y el siguiente código para el caso de número de clusters 8:

$$birch = Birch(n_clusters = n_Clust, threshold = 0.2)$$

Analizaremos los resultados para los números de clusters iguales a 2, 4 y 8. Este algoritmo se ha lanzado con el subconjunto de datos completo(49280 muestras), ya que es rápido. En las siguientes tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
Birch	19187.018	0.658	0.78650	2
Birch	89114.988	0.822	0.80961	4
Birch	47238.344	0.823	0.74986	8

Table 2.9: Métricas Birch, Caso 1.

Cluster	Población
0	3603
1	45677

Table 2.10: Pobalción clusters Birch, Caso 1, número de clusters 2.

Cluster	Población
0	1139
1	13008
2	2464
3	32669

Table 2.11: Pobalción clusters Birch, Caso 1, número de clusters 4.

Cluster	Población
0	2488
1	757
2	19
3	13309
4	99
5	191
6	32311
7	106

Table 2.12: Pobalción clusters Birch, Caso 1, número de clusters 8.

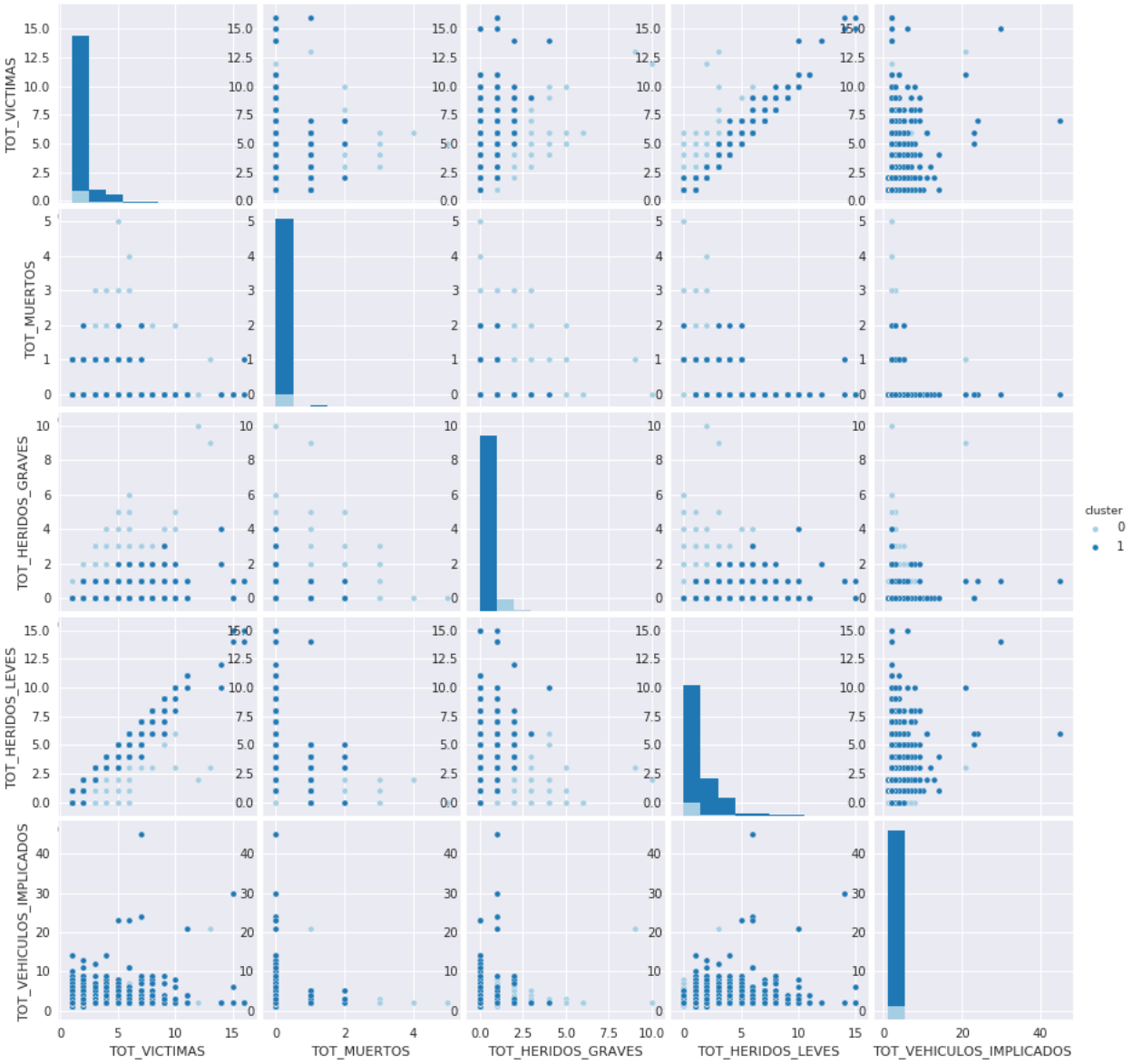

Figure 2.12: ScatterMatrix de Birch, caso 1, número de clusters 2.

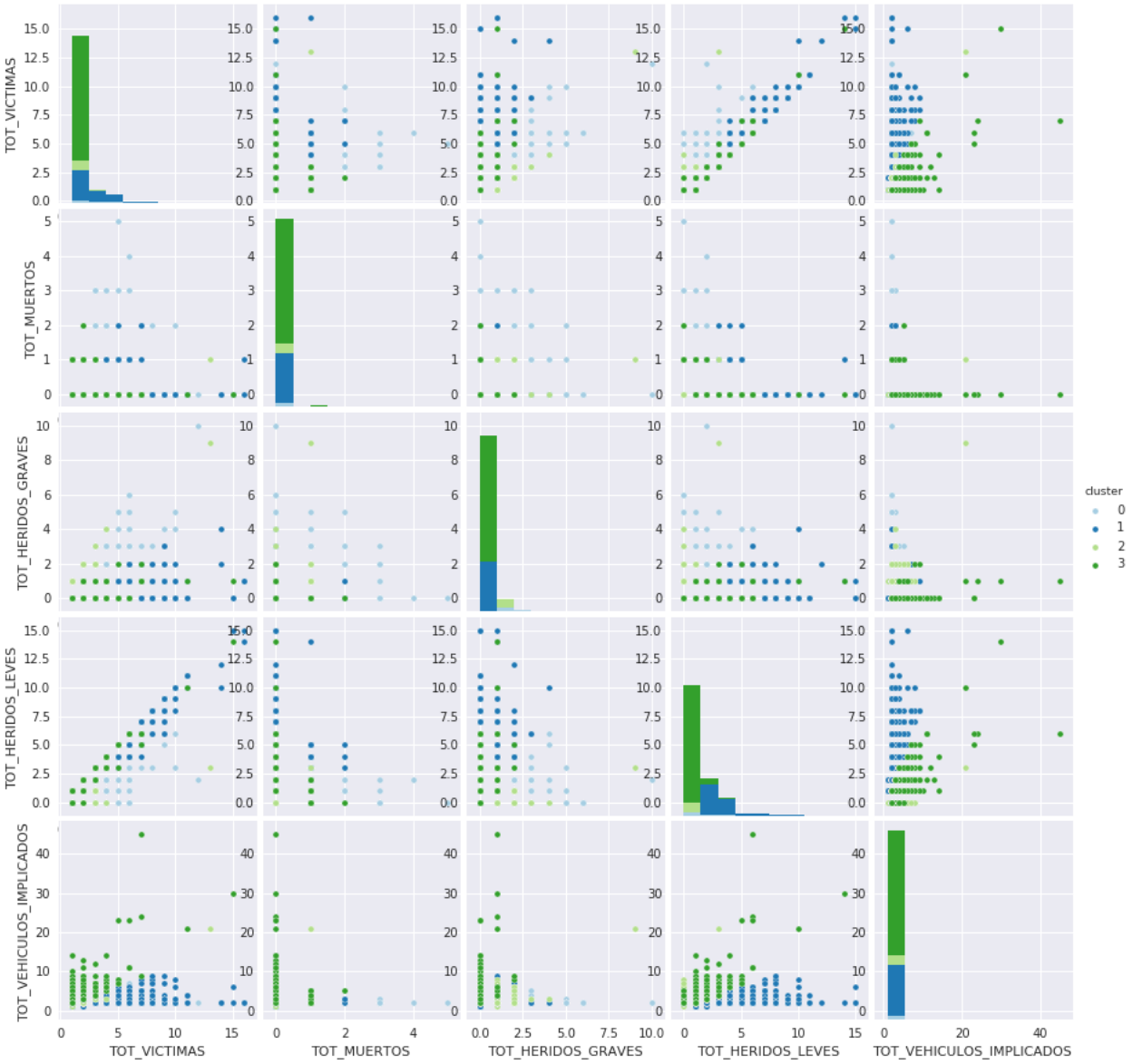

Figure 2.13: ScatterMatrix de Birch, caso 1, número de clusters 4.

Figure 2.14: ScatterMatrix de Birch, caso 1, número de clusters 8.

2.5 MeanShift

Este algoritmo se ha definido mediante el siguiente código:

$$meanshift = MeanShift(bin\_seeding = True)$$

Dado que es un algoritmo costoso computacionalmente se ha sampleado la muestra con una población de 10 000 individuos, como observamos a continuación:

```
#Mean_Shift
Xmns = X.sample(10000, random_state=123456) #10 000
Xmns_norm = preprocessing.normalize(Xmns, norm='l2')
```

Figure 2.15: Sampleado de datos para el algoritmo Mean Shift, caso 1.

En este algoritmo no podemos establecer el número de clusters que obtendremos, los resultados obtenidos son los siguientes:

Algoritmo	CH	SC	Tiempo	nClusters
MeanShift	16079.273	0.880	25.40174	29

Table 2.13: Métricas MeanShift, Caso 1.

0	5926
1	1762
2	828
3	681
4	430
5	100
7	43
6	34
8	31
9	31
10	29
11	25
12	19
13	11
14	9
15	6
17	5
16	5
18	4
19	3
20	3
21	3
25	2
27	2
24	2
22	2
23	2
26	1
28	1

Figure 2.16: Pobalción clusters MeanShift, Caso 1, número de clusters 29.

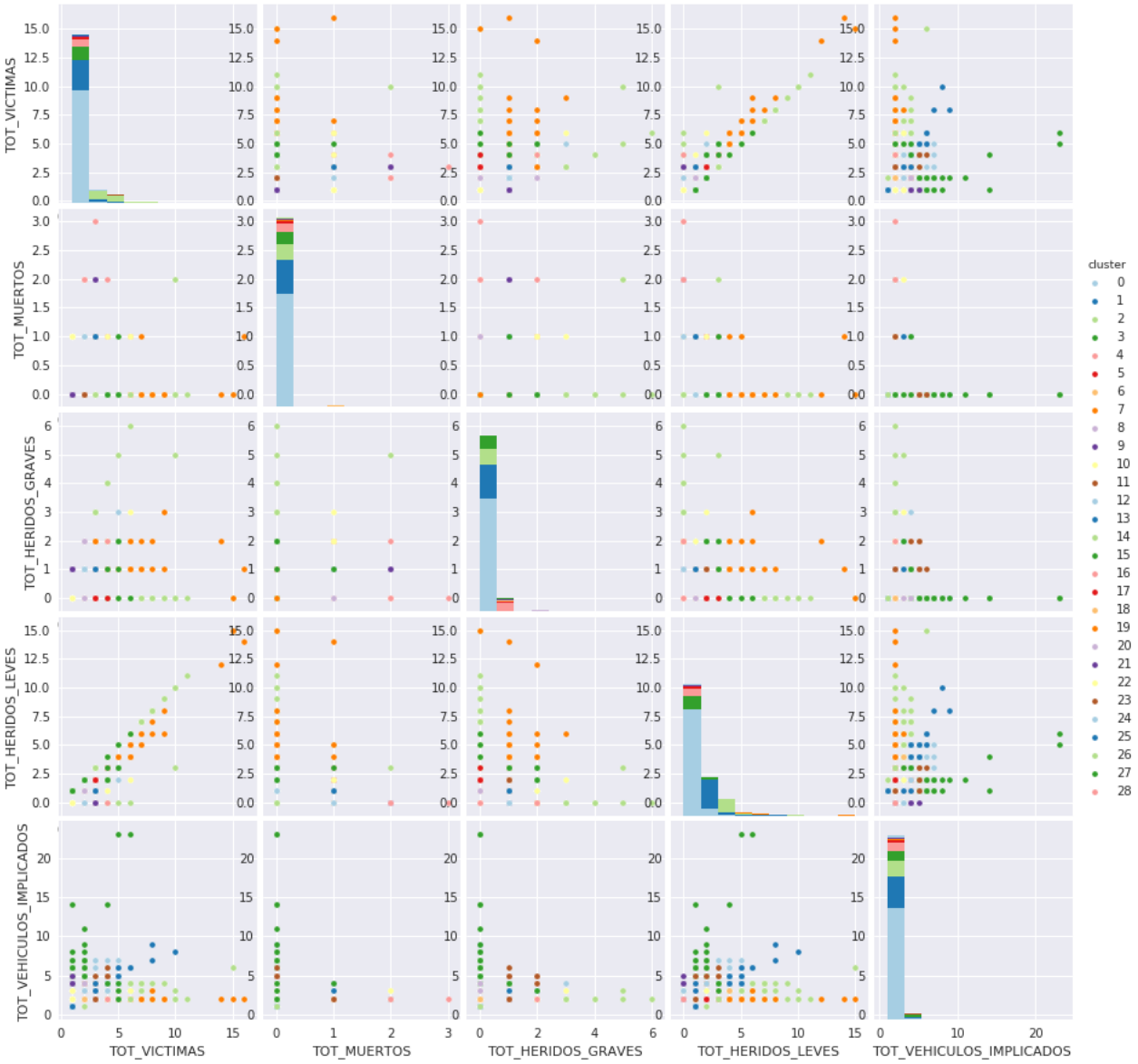

Figure 2.17: ScatterMatrix de MeanShift, caso 1, número de clusters 29.

2.6 MiniBatchKMeans

Este algoritmo se ha definido mediante el siguiente código:

$$mbkm = MiniBatchKMeans(n_{clusters} = n_{Clust})$$

Analizaremos los resultados para los números de clusters iguales a 2, 4 y 8. Este algoritmo se ha lanzado con el subconjunto de datos completo(49280 muestras), ya que es rápido.

En las siguientes tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
MinBKMeans	62847.198	0.725	0.07599	2
MinBKMeans	89482.367	0.822	0.14840	4
MinBKMeans	130304.843	0.865	0.20451	8

Table 2.14: Métricas MiniBatchKMeans, Caso 1.

Cluster	Población
0	35727
1	13553

Table 2.15: Pobalción clusters MiniBatchKMeans, Caso 1, número de clusters 2.

Cluster	Población
0	32474
1	13008
2	2658
3	1140

Table 2.16: Pobalción clusters MiniBatchKMeans, Caso 1, número de clusters 4.

Cluster	Población
0	29069
1	229
2	2447
3	4417
4	3250
5	343
6	989
7	8536

Table 2.17: Pobalción clusters MiniBatchKMeans, Caso 1, número de clusters 8.

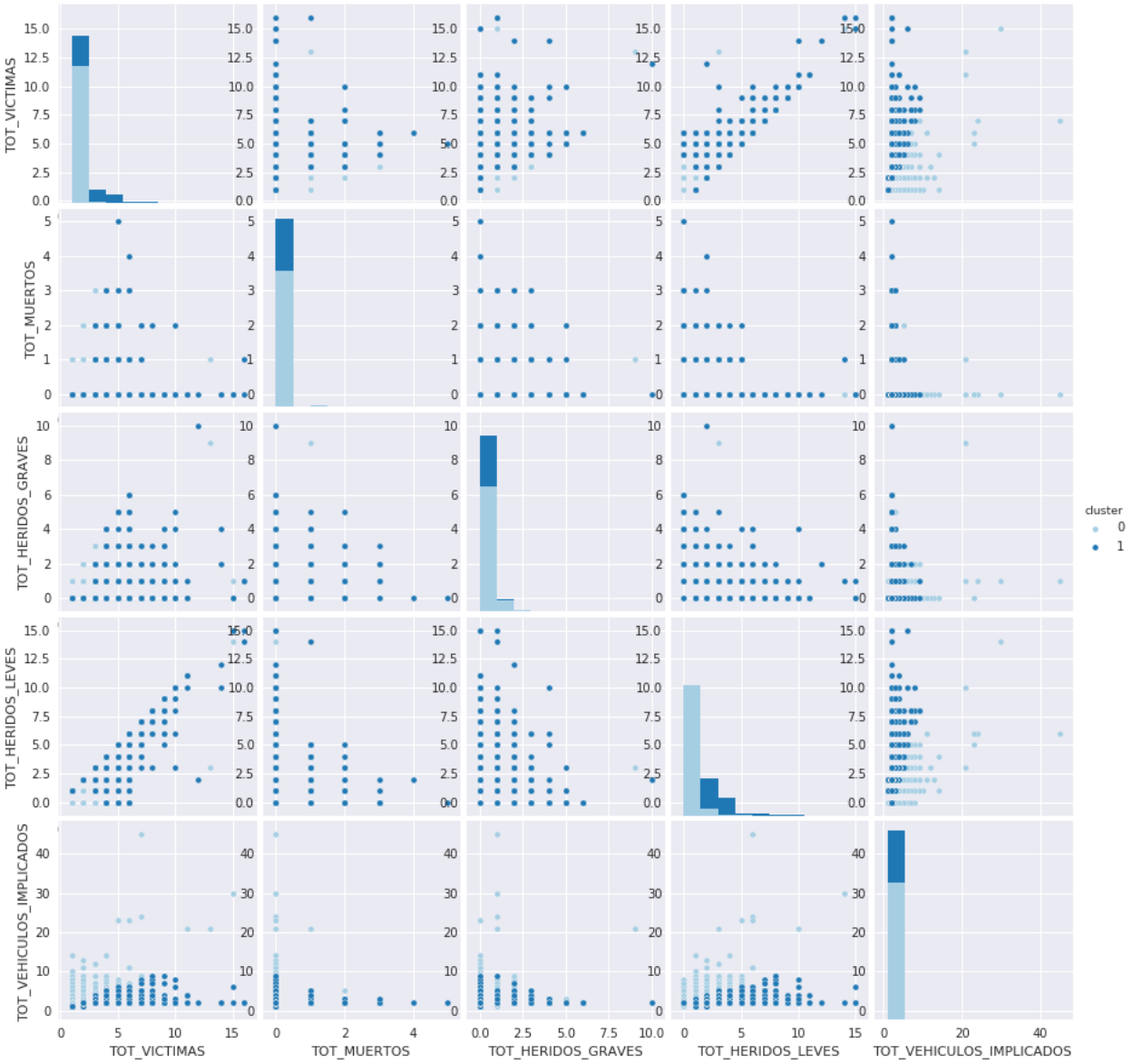

Figure 2.18: ScatterMatrix de MiniBatchKMeans, caso 1, número de clusters 2.

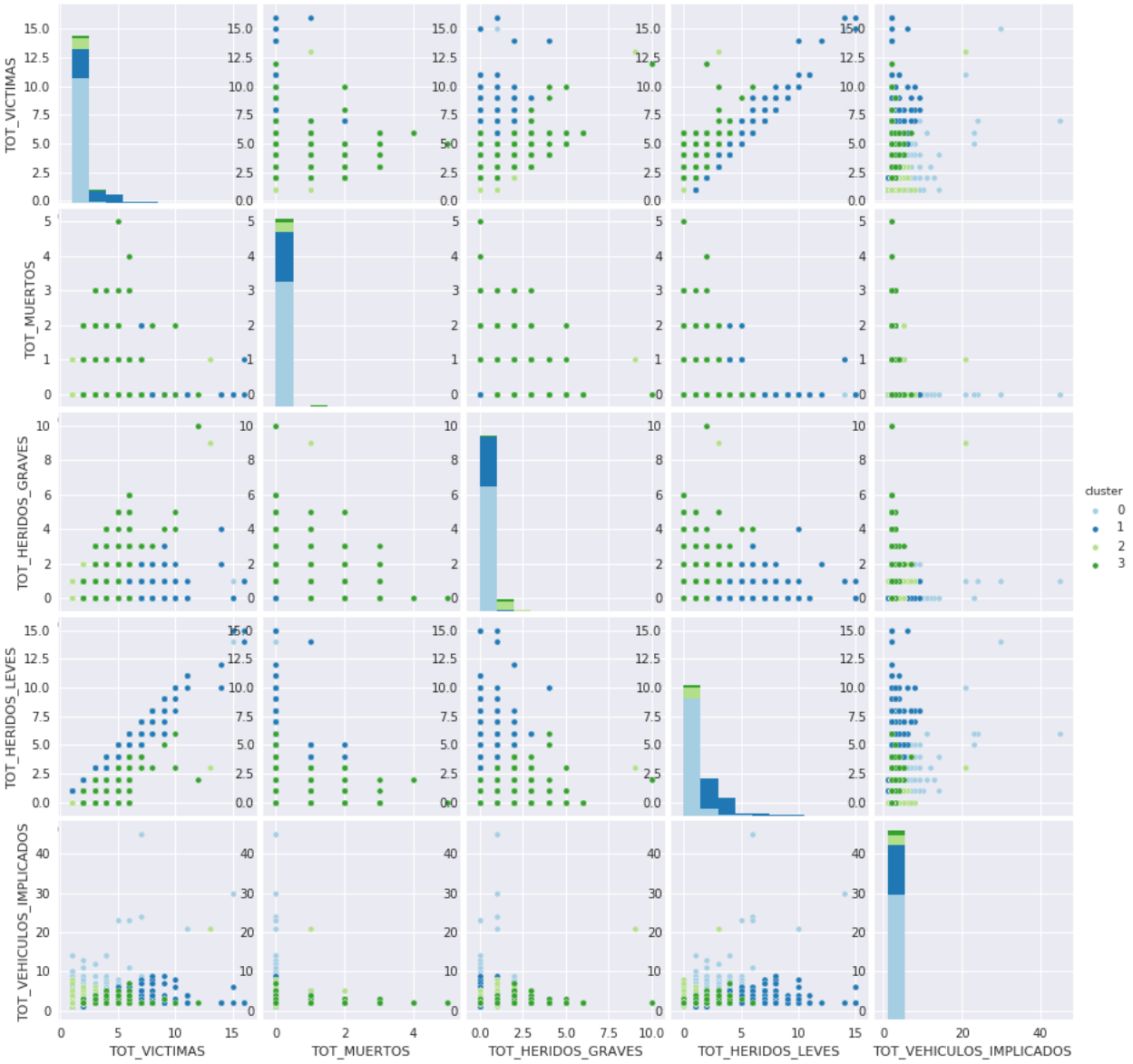

Figure 2.19: ScatterMatrix de MiniBatchKMeans, caso 1, número de clusters 4.

Figure 2.20: ScatterMatrix de MiniBatchKMeans, caso 1, número de clusters 8.

2.7 Interpretación de la Segmentación

Una vez expuestos todos los resultados vamos a centrarnos en las datos de interés obtenidos, no vamos a explicar todos los resultados puesto que es demasiado extenso y son muy parecidos entre los distintos algoritmos. Primero comentaremos el rendimiento general de los algoritmos, agruparemos los estadísticos en las siguientes tablas:

Algoritmo	CH	SC	Tiempo
KMeans	62831.637	0.725	0.07192
AgglomerativeClustering	24584.466	0.706	7.88352
Birch	19187.018	0.659	0.78650
MeanShift	16079.273	0.881	25.40174
MiniBatchKMeans	62847.198	0.725	0.07599

Table 2.18: Estadísticos generales del caso 1, para un número de clusters 2.

Algoritmo	CH	SC	Tiempo
KMeans	96112.886	0.811	0.10434
AgglomerativeClustering	38953.316	0.810	7.95663
Birch	89114.988	0.823	0.80961
MeanShift	16079.273	0.881	25.02240
MiniBatchKMeans	89482.367	0.823	0.14840

Table 2.19: Estadísticos generales del caso 1, para un número de clusters 4.

Algoritmo	CH	SC	Tiempo
KMeans	157355.631	0.885	0.18275
AgglomerativeClustering	57923.649	0.907	8.26727
Birch	47238.344	0.824	0.74986
MeanShift	16079.273	0.881	25.64855
MiniBatchKMeans	130304.843	0.866	0.20451

Table 2.20: Estadísticos generales del caso 1, para un número de clusters 8.

Primero observemos que el algoritmo MeanShift no varía de una tabla a otra ya que el número de clusters obtenidos es el mismo, es un parámetro que calcula automáticamente el algoritmo. En el resto de algoritmos podemos observar que a medida que incrementa el número de clusters lo hace también el *Calinski-Harabaz Index* ya que este índice mide la *similitud de los datos en el mismo cluster*, luego al haber un mayor número de clusters los datos agrupados en cada cluster son mas *parecidos*. Pero esta medida solo puede ser tenida en cuenta en el mismo conjunto de datos, con el mismo número de clusters y el mismo método de clusterización, luego podriamos decir que deseamos un número mayor número en este indice, contemplando que los clusters obtengan un conjunto de datos elevados, ya que demasiados cluster con poca población no nos ayuda en la tarea de encontrar grandes patrones, si no casos específicos. Sin embargo el *Silhouette Coefficient* podemos compararlo entre los distintos algoritmos ya que este al final nos indica la diferenciación (distancia) de los datos de un cluster a los datos de los demás clusters, respecto a este indice observamos que con 8 clusters obtenemos una muy buena medida (muy cercana a 1), siendo el Agglomerative Clustering el que mejor resultado obtiene, pero recordemos que dicho algoritmo esta trabajando con la mitad de los datos aproximádamente. Observamos que con un número de clusters igual a 2 este indice baja drásticamente, ya que necesita de más clusters para lograr una mayor diferenciación. Luego los parámetros mas representativos son con un número de clusters 4 y 8, aunque usaremos todos para extraer la información subyacente en los datos.

Una vez hablado de los algoritmos vamos a centrarnos en la interpretación de los Scatter-Matrix, para poder verificar lo expuesto en el análisis inicial y descubrir nuevos patrones. Observemos la siguiente fiugra:

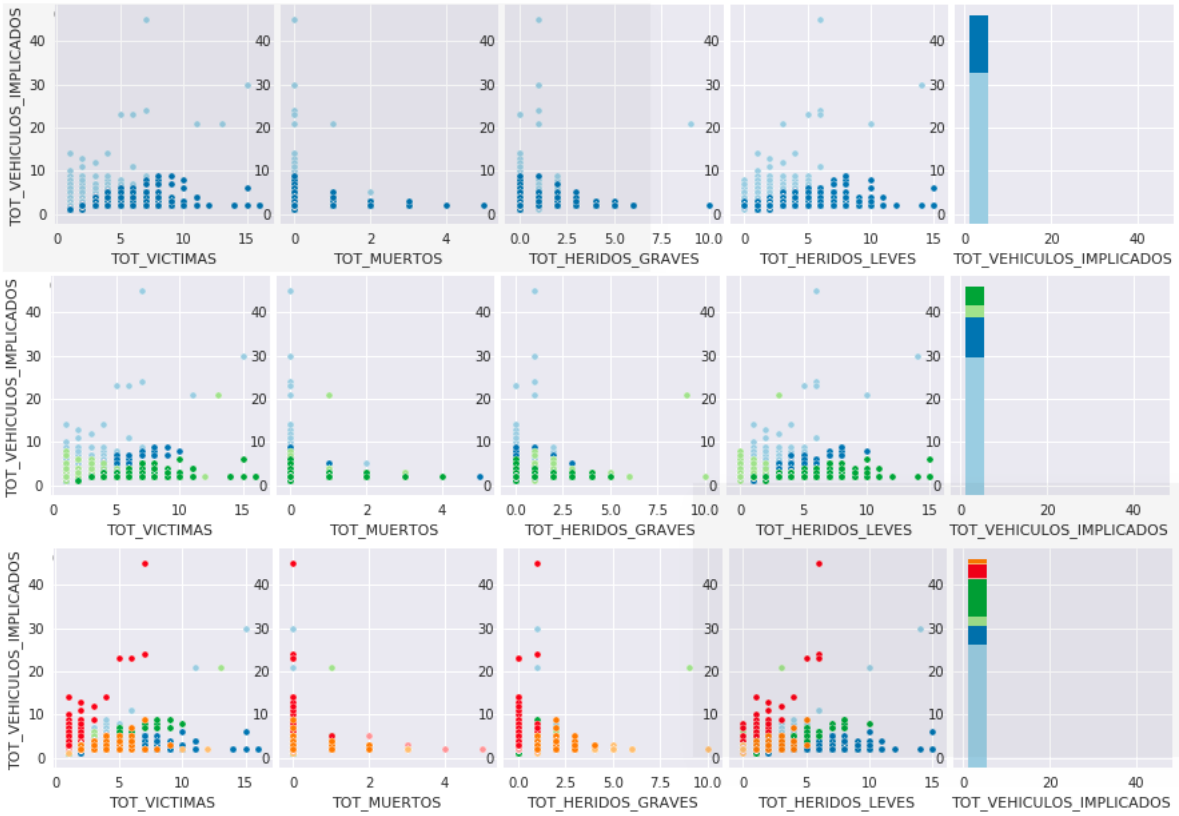

Figure 2.21: ScatterMatrix agrupada del algoritmo Kmeans.

En estos estractos del algoritmo KMeans observamos como a mayor números de vehiculos implicados no implica que la mortalidad sea mayor, de hecho, suele ser lo contrario, a mayor número de vehiculos mayor número de heridos leves y menor número de heridos graves y muertos. Concretemos estos casos:

- La mayoria de accidentes múltiples el número de vehículos implicados es cercano a 10, la mortalidad es 0 o muy cercana a 0, el número de heridos graves no supera el 2.5 y los heridos leves están entorno al 5. Podemos verificarlo, por ejemplo, a partir del ScatterMatrix con 8 clusters del álgoritmo MiniBatchKMeans, cluster 0, que tiene 29069 individuos.
- Otro grupo bien diferenciado es el que tiene mayor mortalidad, 4 o más, que se produce cuando hay pocos vehículos implicados, 2 o 3 mayoritariamente, en los que hay muy pocos heridos leves y prácticamente no hay heridos graves, pero son casos aislados. Esto lo podemos observar en el ScatterMatrix para 8 clusters del algoritmo Birch, cluster 2, con 19 muestras.
- Identificamos patrones presentes prácticamente en todos los resultados, hay un pico cuando el total de victimas es ligeramente mayor a 5, hasta ese punto crecen

el número de heridos leves y graves, y a partir de ahí decrece el número de heridos graves mientras que el de heridos leves sigue en aumento, al mismo tiempo el número de vehículos implicados se suele mantener por debajo de 10, decreciendo cuando el total de víctimas es superior a 10.

- A la luz de los resultados podemos confirmar lo expuesto en el análisis inicial, generalmente un mayor número de heridos leves, implica un menor número de heridos graves y muertos, y este patron es el predominante en los accidentes en los que hay mayor número de vehículos implicados.

3 Caso 2

En este caso analizaremos la gravedad de los accidentes provocados por salida de la via en los fines de semana, relacionando con la hora a la que se producen. Las variables de interés en este caso son la mortalidad, heridos leves y graves, víctimas totales y la hora del accidente. Con esto pretendemos encontrar patrones que nos muestren a que hora se producen los accidentes y la fatalidad de los mísmos. Se ha segmentado la base de datos de la siguiente manera:

```
#preparación de datos----------------------------------------------------------------
accidentes = pd.read_csv('accidentes_2013.csv')

subset = accidentes

#salidas de via los fines de semana
subset = accidentes.loc[(accidentes['DIASEMANA']>=5) & (accidentes['DIASEMANA']<=7)]
subset = subset[subset['TIPO_ACCIDENTE'].str.contains('Salida de la vía')]


#seleccionar variables de interÃ©s para clustering
var_interes = ['TOT_VICTIMAS', 'TOT_MUERTOS', 'TOT_HERIDOS_GRAVES', 'TOT_HERIDOS_LEVES',
               'HORA']
X = subset[var_interes]
```

Figure 3.1: Preparación de datos para el caso de estudio 2.

3.1 Análisis Inicial

Se han obtenido 3 dendogramas con mapas de calor asociados, mediante el algoritmo ward sobre la base de datos descrita anteriormente. Primero generamos el clustermap simple:

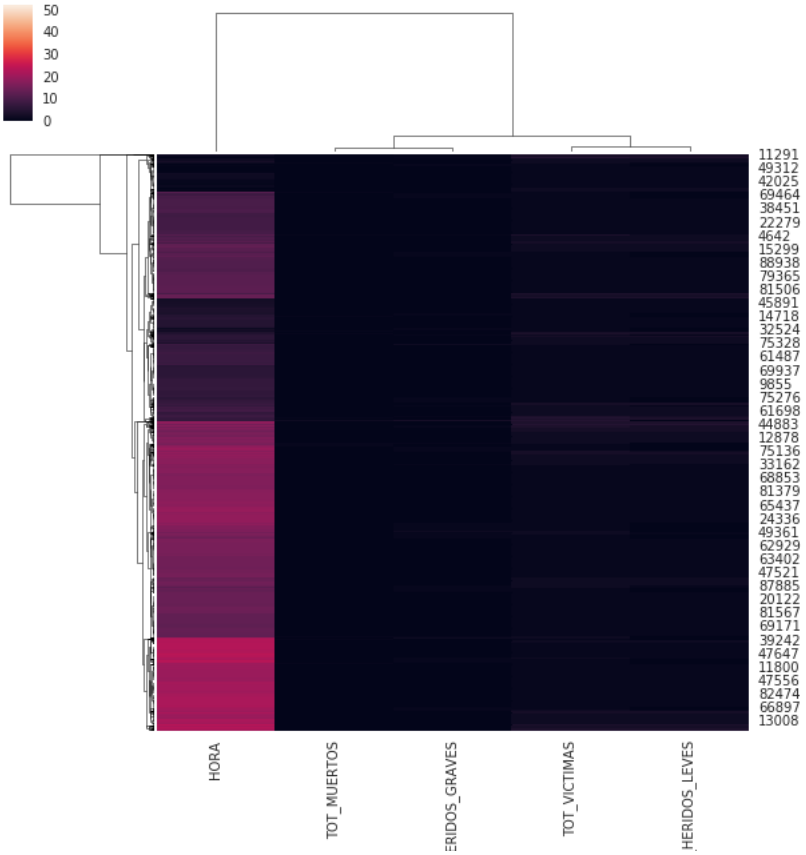

Figure 3.2: ClusterHeatMap obtenido para el caso 2.

Podemos observar como el número de víctimas es mayor para determinadas horas, pero es dificil aventurar las horas por el color, podemos sacar la misma conclusión para la mortalidad. Observemos el cluster map de correlaciones a ver si podemos obtener más información:

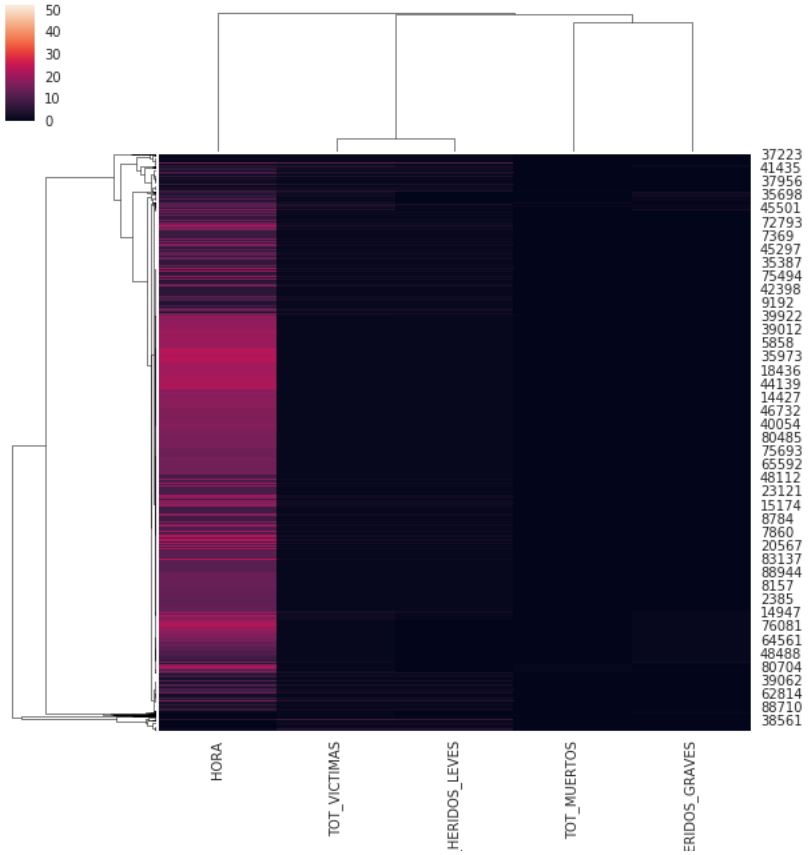

Figure 3.3: ClusterHeatMap utilizando la correlación como medida de distancia obtenido para el caso 2.

Lógicamente los atributos total de víctimas y heridos leves están muy correlacionadas, esto también lo dedujimos en el caso de estudio anterior. Respecto a la hora se intuye que hay ciertos intervalos que afectan al resto de los atributos, pero sigue siendo dificil sacar alguna conclusión certera. Observemos por último el clusterheatmap normalizado por filas:

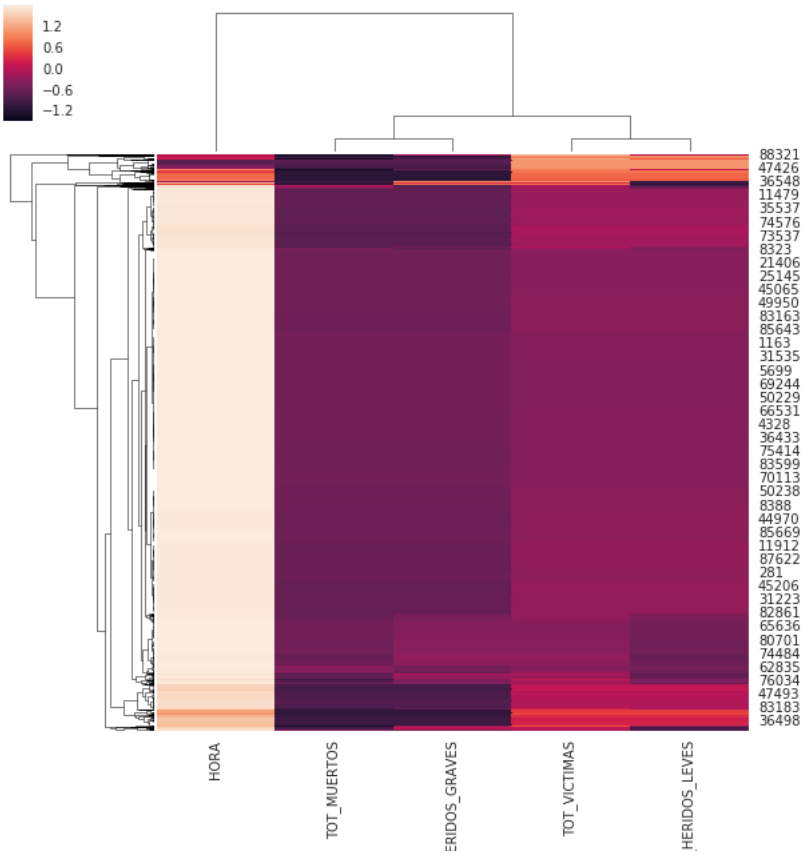

Figure 3.4: ClusterHeatMap normalizando valores por filas para el caso 2.

En este gráfico observamos claramente algunos patrones mencionados en el caso anterior, como la relación heridos leves, heridos graves, total de víctimas y muertos, que en este caso no se ve afectado por la hora. Sin embargo podemos observar como hay ciertas horas en las que los accidentes repercuten en el número de víctimas y heridos leves(parte superior e inferior del grágico de calor).

En este caso los dendogramas y mapas de calor no aportan mucha información sobre como afecta la hora a la gravedad de los accidentes por salidas de la vía, más allá de que hay ciertos intervalos en los que se producen mayores salidas de vía con mayor o menor gravedad. Tendremos que analizar los ScatterMatrix para concretar con mayor precisión.

3.2 Kmeans

Este algoritmo se ha definido mediante el siguiente código:

$$kmeans = KMeans(init =' k - means + +', n_{clusters} = n_{Clust}, n_{init} = 5)$$

Y analizaremos los resultados para los números de clusters iguales a 2, 4 y 8. Este algoritmo se ha lanzado con el subconjunto de datos completo(7872 muestras). En las siguientes tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
KMeans	14408.581	0.805	0.01829	2
KMeans	11903.052	0.589	0.05829	4
KMeans	13082.849	0.519	0.08467	8

Table 3.1: Métricas Kmeans, Caso 2.

Cluster	Población
0	7237
1	635

Table 3.2: Pobalción clusters Kmeans, Caso 2, número de clusters 2.

Cluster	Población
0	1363
1	5856
2	220
3	433

Table 3.3: Pobalción clusters Kmeans, Caso 2, número de clusters 4.

Cluster	Población
0	4077
1	208
2	708
3	145
4	59
5	317
6	2099
7	259

Table 3.4: Pobalción clusters Kmeans, Caso 2, número de clusters 8.

Figure 3.5: ScatterMatrix de Kmeans, caso 2, número de clusters 2.

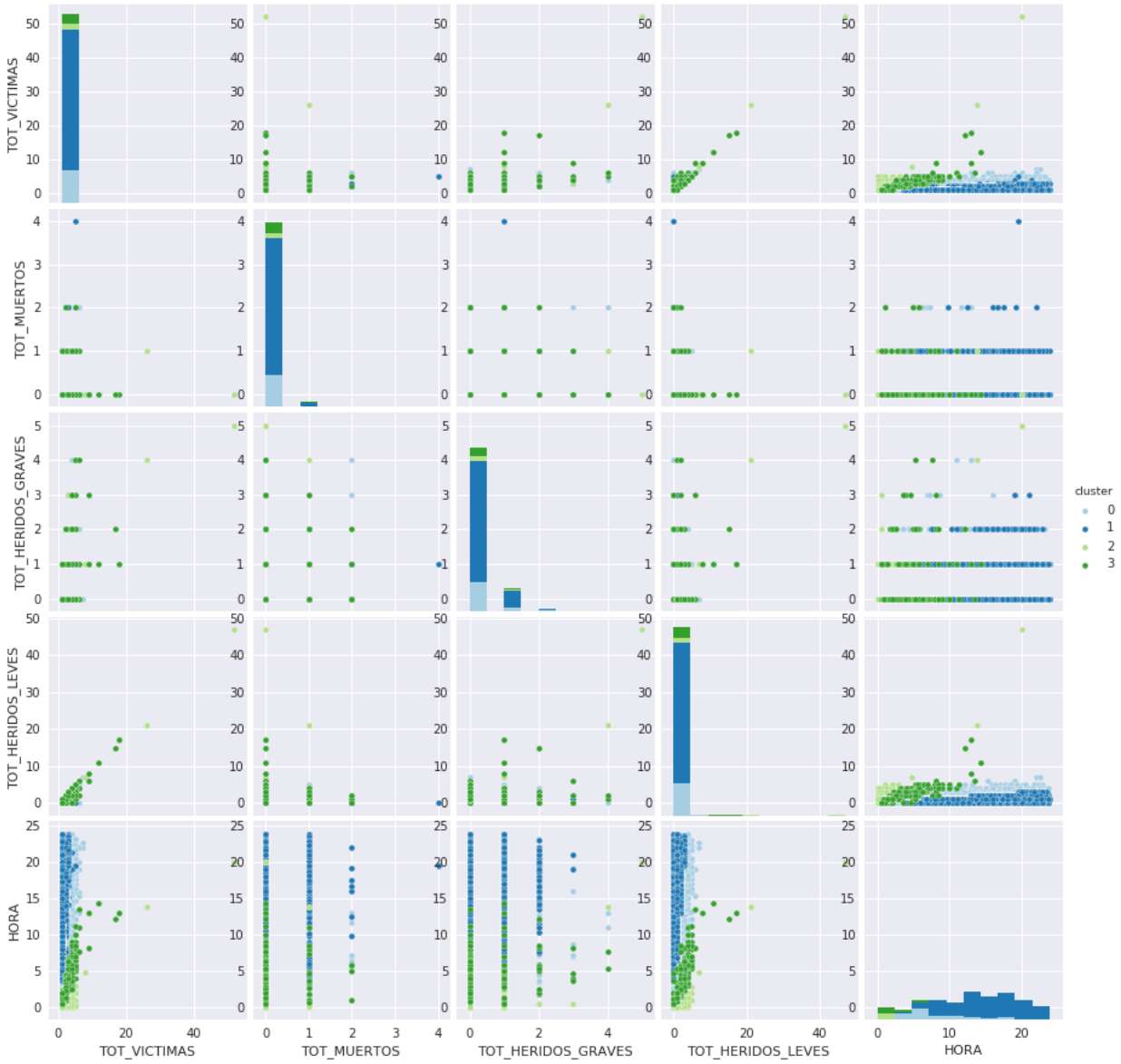

Figure 3.6: ScatterMatrix de Kmeans, caso 2, número de clusters 4.

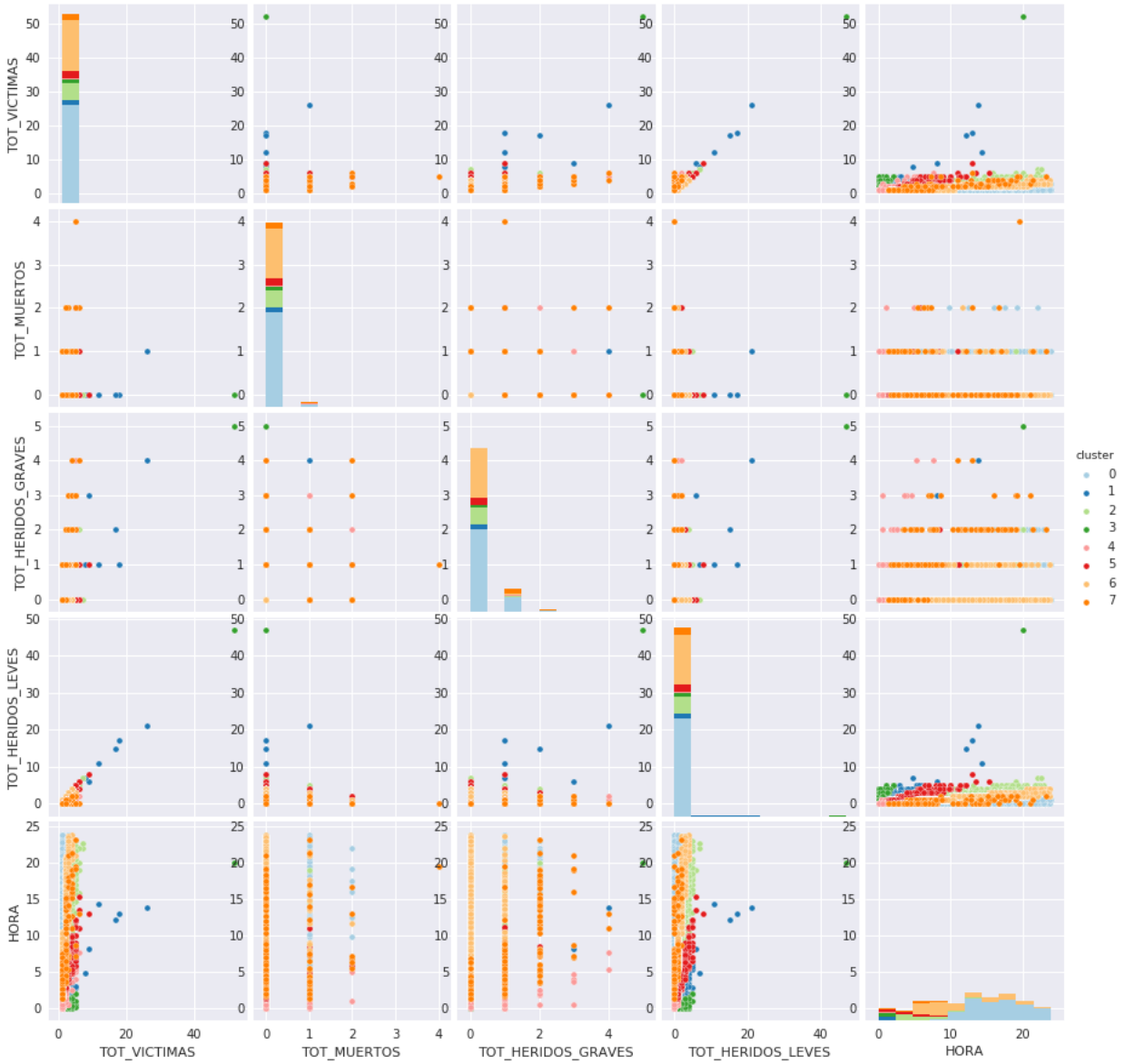

Figure 3.7: ScatterMatrix de Kmeans, caso 2, número de clusters 8.

3.3 Agglomerative Clustering

Este algoritmo se ha definido mediante el siguiente código:

$$Agg = AgglomerativeClustering(n_{clusters} = n_{Clust}, linkage =' ward')$$

Se analizarán los resultados para los números de clusters iguales a 2, 4 y 8. Se ha utilizado la base de datos completa(7872 casos), en las siguientes tablas podemos observar las

estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
AggClust	13663.904	0.79796	1.09614	2
AggClust	10809.871	0.52482	1.07347	4
AggClust	11068.809	0.50655	1.06837	8

Table 3.5: Métricas Agglomerative Clustering, Caso 2.

Cluster	Población
0	774
1	7098

Table 3.6: Pobalción clusters Agglomerative Clustering, Caso 2, número de clusters 2.

Cluster	Población
0	571
1	2258
2	203
3	4840

Table 3.7: Pobalción clusters Agglomerative Clustering, Caso 2, número de clusters 4.

Cluster	Población
0	4840
1	129
2	264
3	1458
4	57
5	385
6	203
7	536

Table 3.8: Pobalción clusters Agglomerative Clustering, Caso 2, número de clusters 8.

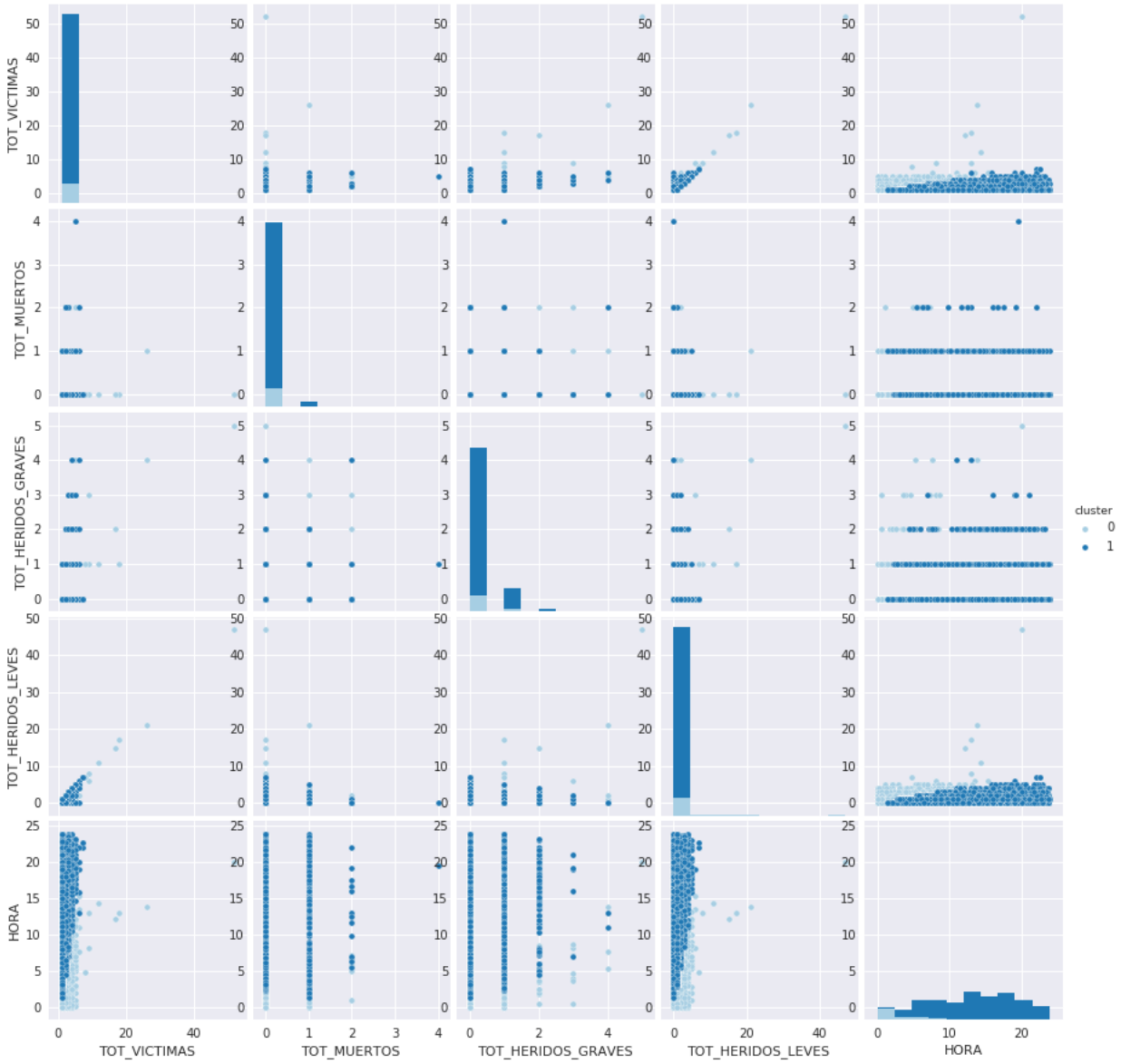

Figure 3.8: ScatterMatrix de Agglomerative Clustering, caso 2, número de clusters 2.

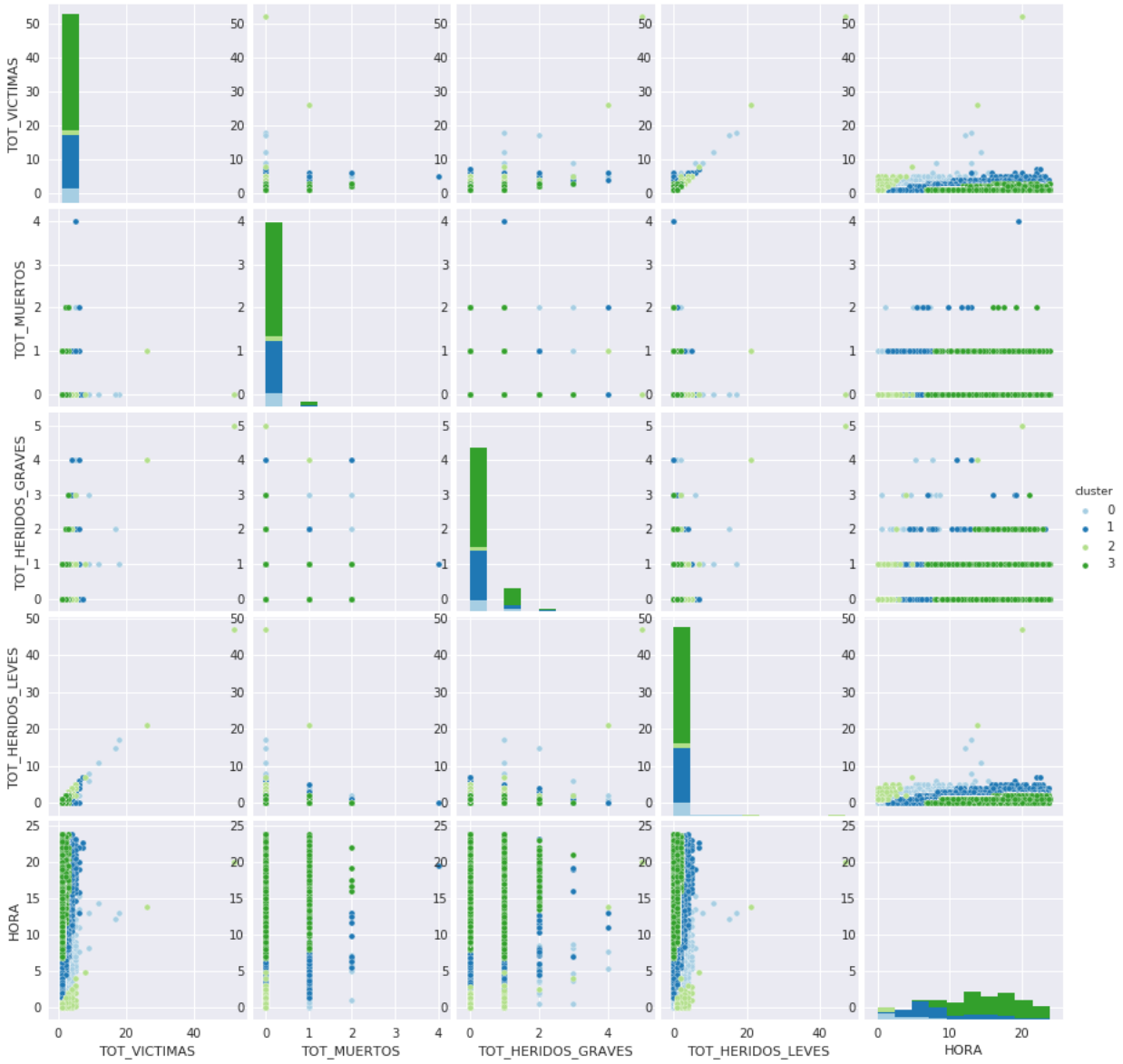

Figure 3.9: ScatterMatrix de Agglomerative Clustering, caso 2, número de clusters 4.

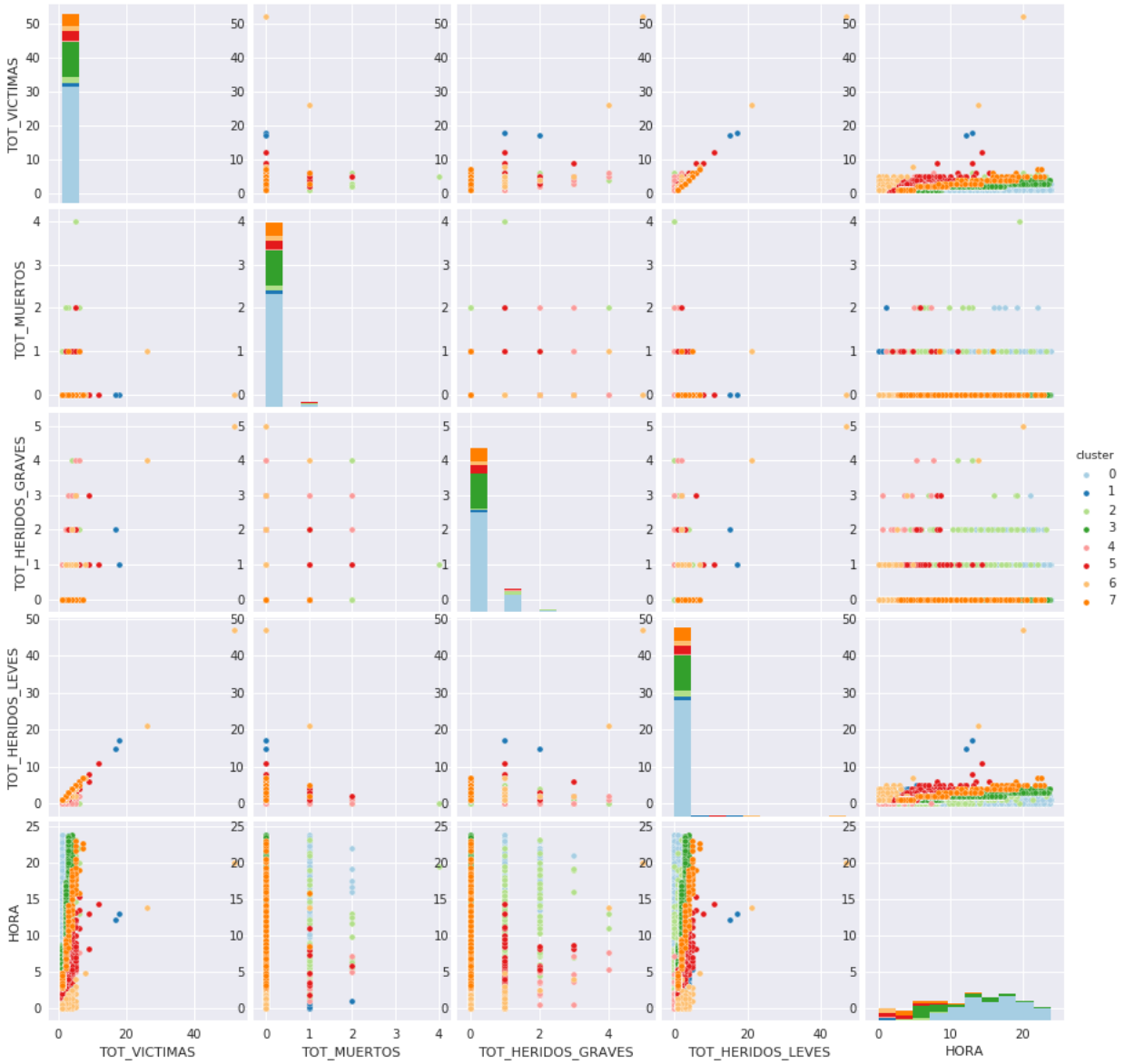

Figure 3.10: ScatterMatrix de Agglomerative Clustering, caso 2, número de clusters 8.

3.4 Birch

Este algoritmo se ha definido mediante el siguiente código para los casos de número de clusters 2, 4 y 8:

$$birch = Birch(n_clusters = n_Clust, threshold = 0.2)$$

Analizaremos los resultados para los números de clusters iguales a 2, 4 y 8. Este algoritmo

se ha lanzado con el subconjunto de datos completo(7872 muestras), ya que es rápido. En las siguientes tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
Birch	14055.370	0.80467	0.11926	2
Birch	6165.079	0.79853	0.13812	4
Birch	4511.572	0.75844	0.11803	8

Table 3.9: Métricas Birch, Caso 2.

Cluster	Población
0	680
1	7192

Table 3.10: Pobalción clusters Birch, Caso 2, número de clusters 2.

Cluster	Población
0	80
1	591
2	9
3	7192

Table 3.11: Pobalción clusters Birch, Caso 2, número de clusters 4.

Cluster	Población
0	7
1	7192
2	397
3	61
4	19
5	14
6	2
7	180

Table 3.12: Pobalción clusters Birch, Caso 2, número de clusters 8.

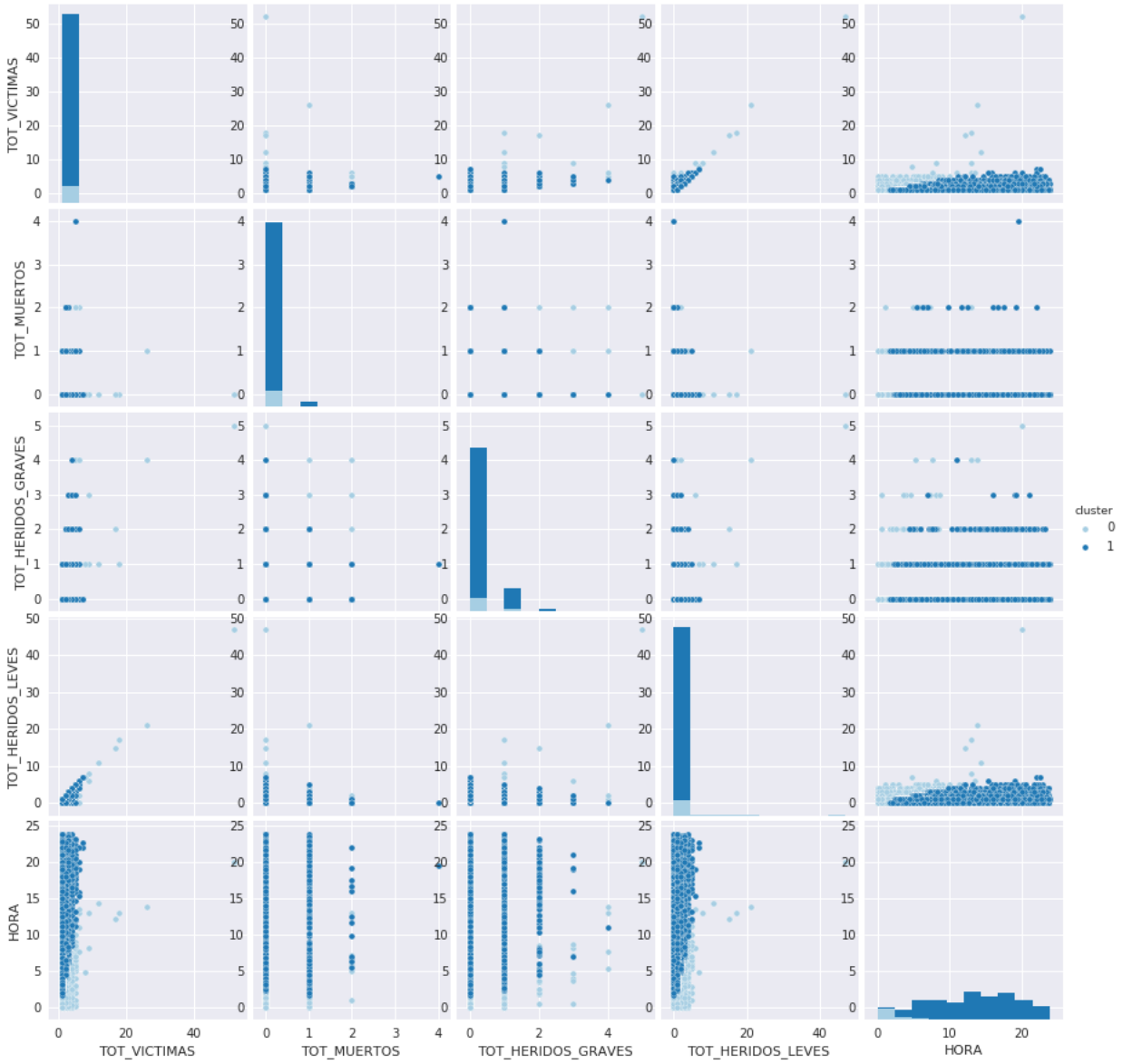

Figure 3.11: ScatterMatrix de Birch, caso 2, número de clusters 2.

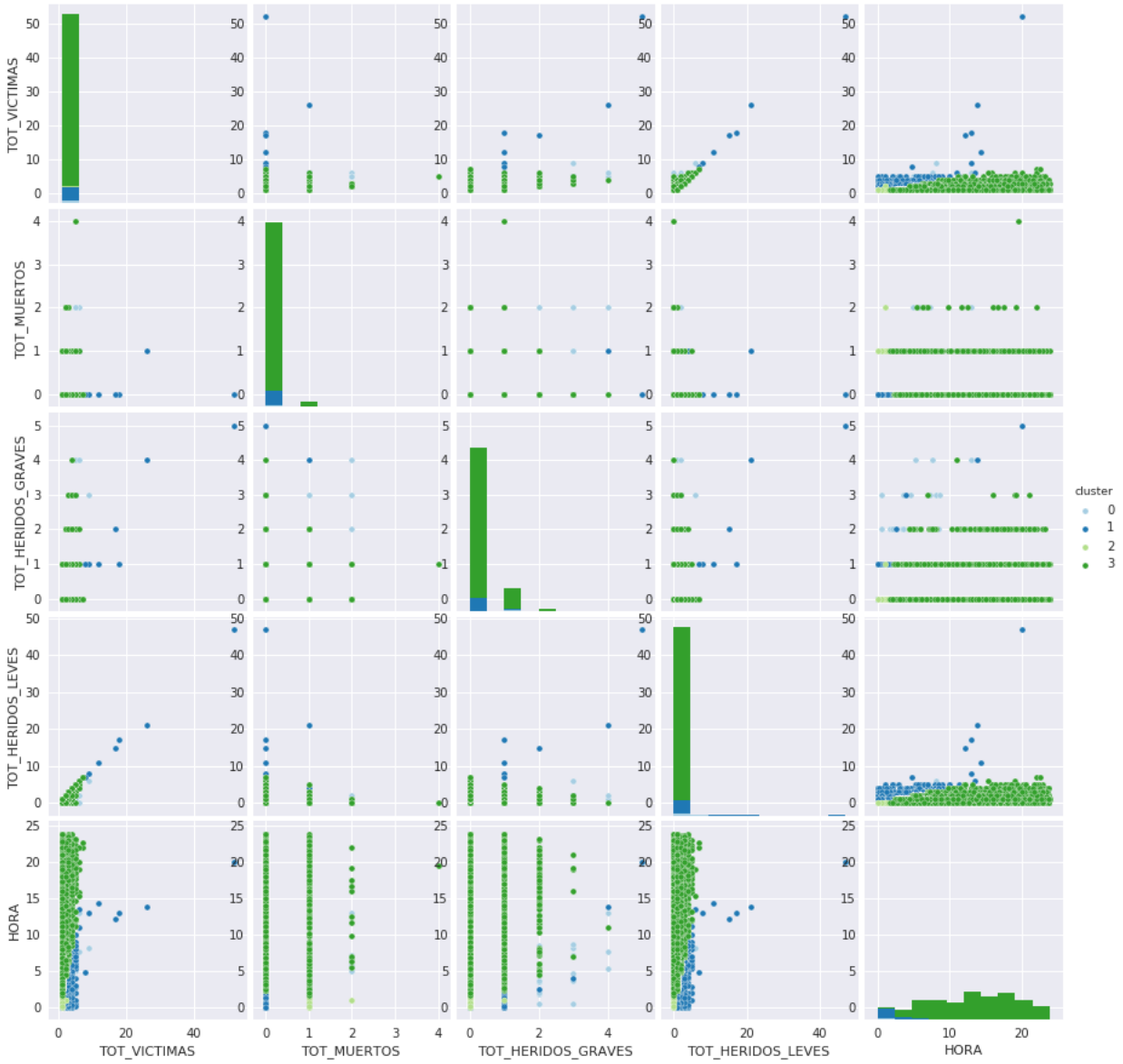

Figure 3.12: ScatterMatrix de Birch, caso 2, número de clusters 4.

Figure 3.13: ScatterMatrix de Birch, caso 2, número de clusters 8.

3.5 MeanShift

Este algoritmo se ha definido mediante el siguiente código:

$$meanshift = MeanShift(bin\_seeding = True)$$

Se ha realizado con la base de datos al completo(7872 muestras), para este algoritmo no

podemos establecer el número de clusters que obtendremos, los resultados obtenidos son los siguientes:

Algoritmo	CH	SC	Tiempo	nClusters
MeanShift	2545.901	0.635	3.57629	29

Table 3.13: Métricas MeanShift, Caso 2.

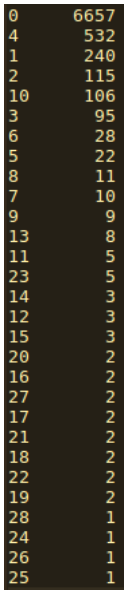

Figure 3.14: Pobalción clusters MeanShift, Caso 2, número de clusters 29.

Figure 3.15: ScatterMatrix de MeanShift, caso 2, número de clusters 29.

3.6 MiniBatchKmeans

Este algoritmo se ha definido mediante el siguiente código:

$$mbkm = MiniBatchKMeans(n_{clusters} = n_{Clust})$$

Analizaremos los resultados para los números de clusters iguales a 2, 4 y 8. Este algoritmo se ha lanzado con el subconjunto de datos completo(7872 muestras). En las siguientes

tablas podemos observar las estadísticas obtenidas:

Algoritmo	CH	SC	Tiempo	nClusters
MinBKMeans	14406.685	0.80533	0.01446	2
MinBKMeans	11374.752	0.70053	0.01636	4
MinBKMeans	12889.202	0.52962	0.03916	8

Table 3.14: Métricas MiniBatchKMeans, Caso 2.

Cluster	Población
0	7234
1	638

Table 3.15: Pobalción clusters MiniBatchKMeans, Caso 2, número de clusters 2.

Cluster	Población
0	6561
1	339
2	882
3	90

Table 3.16: Pobalción clusters MiniBatchKMeans, Caso 2, número de clusters 4.

Cluster	Población
0	898
1	187
2	512
3	4123
4	112
5	270
6	1727
7	43

Table 3.17: Pobalción clusters MiniBatchKMeans, Caso 2, número de clusters 8.

Figure 3.16: ScatterMatrix de MiniBatchKMeans, caso 2, número de clusters 2.

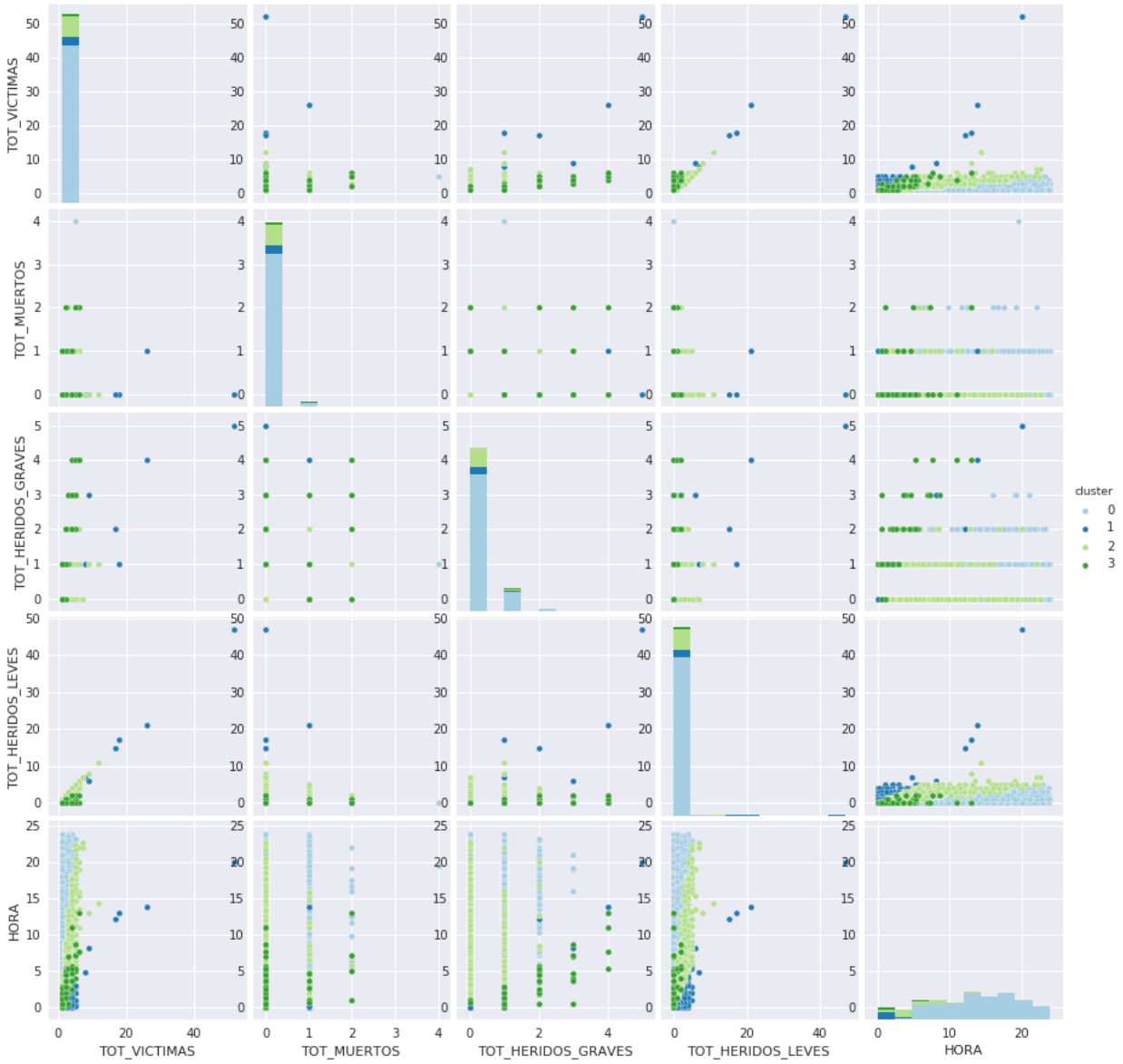

Figure 3.17: ScatterMatrix de MiniBatchKMeans, caso 2, número de clusters 4.

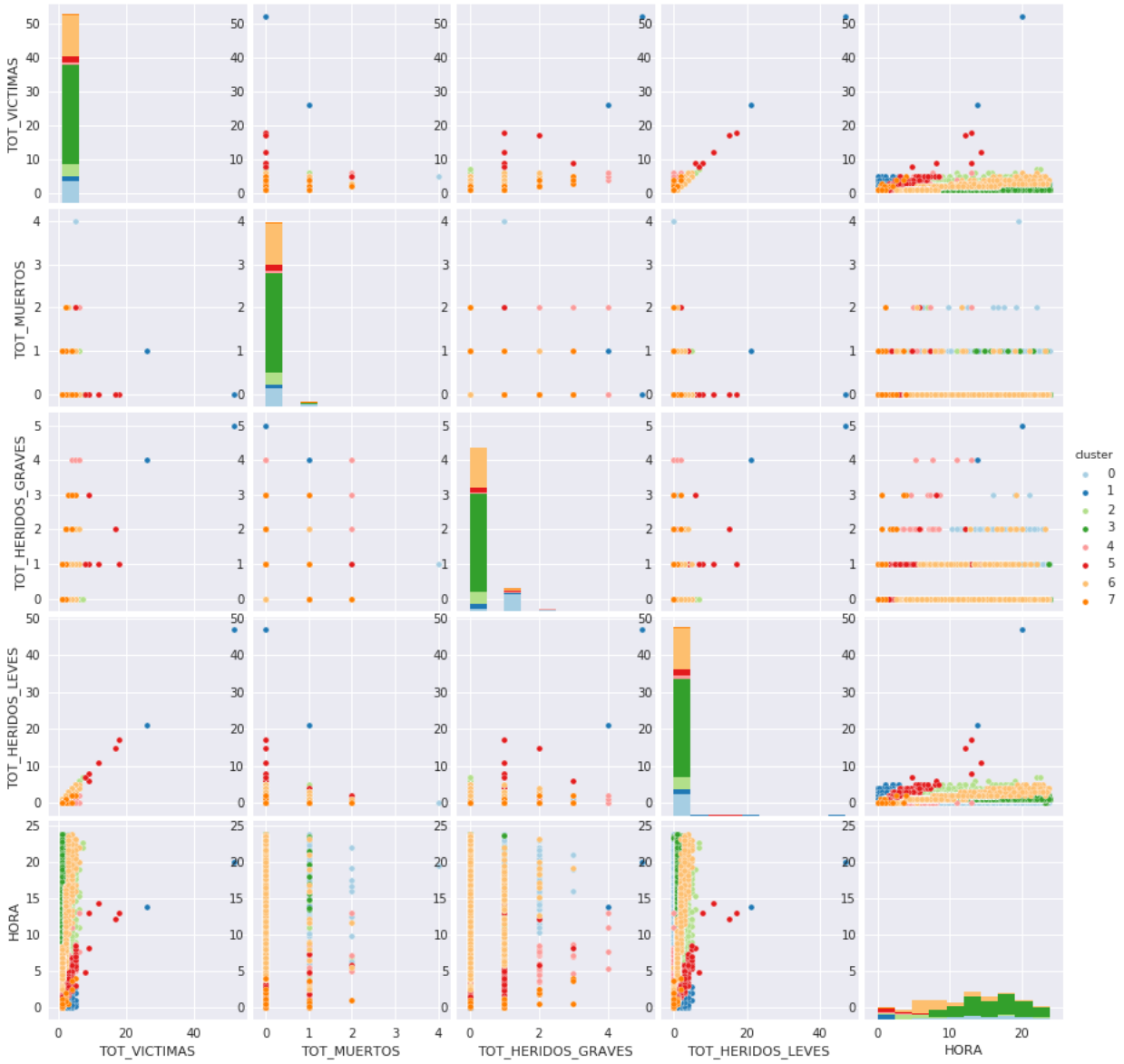

Figure 3.18: ScatterMatrix de MiniBatchKMeans, caso 2, número de clusters 8.

3.7 Interpretación de la Segmentación

Primero comentaremos el rendimiento general de los algoritmos, agruparemos los estadísticos en las siguientes tablas:

Algoritmo	CH	SC	Tiempo
KMeans	14408.581	0.805	0.01829
AgglomerativeClustering	13663.904	0.798	1.09614
Birch	14055.370	0.805	0.11926
MeanShift	2545.901	0.635	3.57629
MiniBatchKMeans	14406.685	0.805	0.01446

Table 3.18: Estadísticos generales del caso 2, para un número de clusters 2.

Algoritmo	CH	SC	Tiempo
KMeans	11903.052	0.590	0.05829
AgglomerativeClustering	10809.871	0.525	1.07347
Birch	6165.079	0.799	0.13812
MeanShift	2545.901	0.635	3.59711
MiniBatchKMeans	11374.752	0.701	0.01636

Table 3.19: Estadísticos generales del caso 2, para un número de clusters 2.

Algoritmo	CH	SC	Tiempo
KMeans	13082.849	0.519	0.08467
AgglomerativeClustering	11068.809	0.507	1.06837
Birch	4511.572	0.758	0.11803
MeanShift	2545.901	0.635	3.70664
MiniBatchKMeans	12889.202	0.530	0.03916

Table 3.20: Estadísticos generales del caso 2, para un número de clusters 2.

Podemos observar que el *Calinski-Harabaz Index* en este caso se comporta mejor en todos los algoritmos para un número de clusters igual a 2, mientras que para 4 clusters obtienen peor resultados los algoritmos. Esto se debe a que en este caso podemos diferenciar un grupo grande, con algunos grupos medianos y muchos casos aislados, esto también lo confirma los resultados obtenidos por el algoritmo MeanShift, que genera 29 clusters y no se ve muy penalizado en el *Silhouette Coefficient*. Se han realizado mediciones para un número de clusters de 12 y los indices seguían cercanos a los obtenidos por 8 clusters por lo mencionado anteriormente. En este caso observamos que el algoritmo Birch mantiene estable el *Silhouette Coefficient* en todas las pruebas por lo que podemos concluir que tiene una gran capacidad de clusterización en bases de datos con outliers. Todos los algoritmos han sido bastante más rapidos que en el caso anterior ya que este tiene un menor número de muestras. El algoritmo que mejor se ha comportado ha sido por lo tanto Birch.

En cuanto al análisis de los ScatterMatrix se han detectado los siguientes patrones:

- Dos grandes grupos, uno por la noche, de 00:00 hasta 07:00 aproximádamente, y otro durante el día, debido a que por el día la circulación es mayor y por lo tanto hay un mayor número de accidentes, esto lo podemos observar en cualquier gráfico de los expuestos para un número de clusters igual 2, fijandonos en la población de los clusters.

- Observamos que los accidentes con mayor número de víctimas totales ocurre al medio día, probablemente porque sea cuando mayor circulación hay, nos falta información para corroborarlo.

- Se pueden volver a observar los patrones de heridos graves, heridos leves, víctimas totales y mortalidad, al margen de la hora y el hecho de que sea fin de semana.

Cabe destacar que un número de clusters igual a 4 se obtiene poca información, ya que no se tiene la suficiente especialización como para detectar los casos más excepcionales. Realménte no podemos sacar datos concisos sobre que los fines de semana se produzcan mas o menos accidentes por desvíos en función de la hora, más allá de las probabilidades por una circulación mayor, este caso se eligió para observar si por la noche/madrugada los fines de semana se producían un mayor número de desvios o los accidentes resultaban con mayor fatalidad.

References

[1] http://datamining.rutgers.edu/publication/internalmeasures.pdf.

[2] http://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/InteligenciaDeNegocio//Curso17-18/accidentes_2013.csv.zip.

[3] http://scikit-learn.org/stable/.

[4] http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster.

[5] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering.

[6] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html#sklearn.cluster.Birch.

[7] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.

[8] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html#sklearn.cluster.MeanShift.

[9] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans.

[10] http://seaborn.pydata.org/generated/seaborn.clustermap.html.

[11] http://seaborn.pydata.org/generated/seaborn.pairplot.html.

[12] https://en.wikipedia.org/wiki/Silhouette_(clustering).

[13] https://matplotlib.org/.

[14] https://pandas.pydata.org/.

[15] https://seaborn.pydata.org/.

[16] https://sedeapl.dgt.gob.es/WEB_IEST_CONSULTA/subcategoria.faces.

[17] https://www.python.org/.