Inteligencia de Negocio. Práctica 2: Visualización y Segmentación

Patricia Córdoba Hidalgo
patriciacorhid@correo.ugr.es
Grupo 2 (Viernes)

21 de noviembre de 2020

Índice

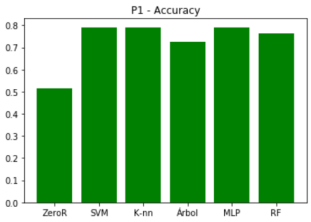
1. Visualización

1.1. Visualización de medidas

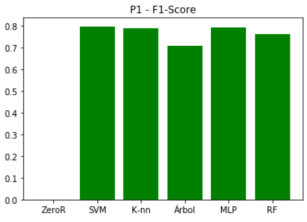
En la práctica 1 se mostraron los datos de cada una de las medidas en tablas, mostrando el valor numérico de éstas. Otra forma de mostrar estos datos es mediante gráficas. En esta práctica, se mostrarán en diagramas de barras los valores de las medidas más utilizadas para la toma de decisiones en la práctica anterior.

Veamos primero los resultados de los diferentes preprocesados de datos:

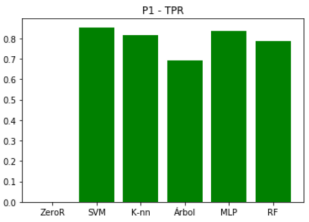
Procesado 1



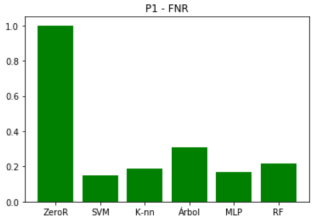
(a) Accuracy



(b) F1-Score

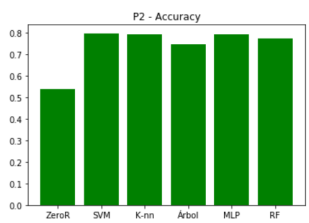


(c) TPR

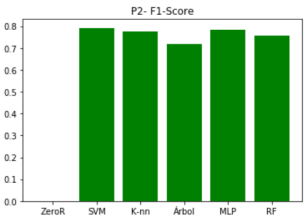


(d) FNR

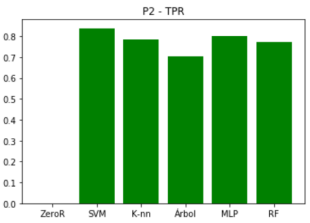
Procesado 2



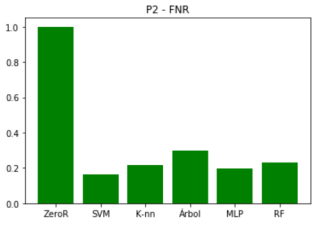
(e) Accuracy



(f) F1-Score

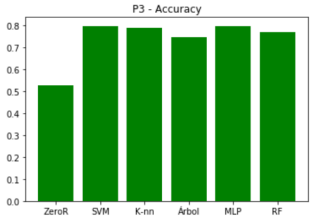


(g) TPR

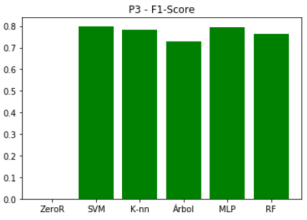


(h) FNR

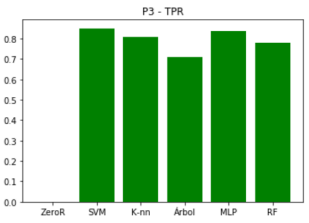
Procesado 3



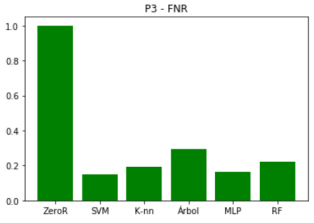
(i) Accuracy



(j) F1-Score



(k) TPR



(l) FNR

El código de cada gráfica es:

```
fig, ax = plt.subplots()
ax.bar(["ZeroR", "SVM", "K-nn", "Arbol", "MLP", "RF"], pX_metrica, color='green')
ax.set_title("PX-metrica")
```

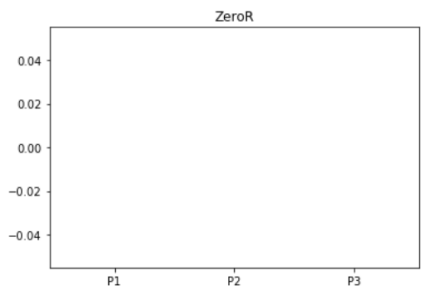
Donde "X" es denota el preprocesado que se ha aplicado, 1, 2 o 3, y " metrica" la métrica que se mide, pudiendo ser `accuracy`, `F1-Score`, `TPR` o `FNR`. El vector pX-metrica es un vector donde se guardan los valores de la métrica "metrica" con el preprocesado "X" de todos los modelos considerados.

Podemos observar que los tres preprocesados obtienen resultados muy similares, como ya comentamos en la práctica anterior. A primera vista, la estructura de los diagramas parece la misma, es decir, al ordenar los diferentes modelos

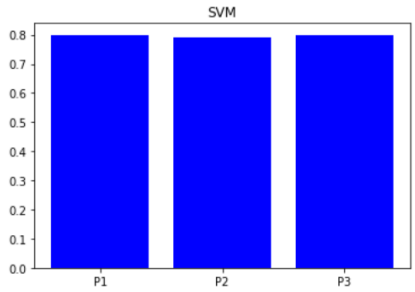
según el valor de la métrica correspondiente con cada procesamiento, este orden es muy parecido en todos ellos, no atreviéndome a decir el mismo por haber barras de alturas semejantes. Es por esto que decantarse por un procesamiento con estos gráficos resulta complicado.

En los tres preprocesamientos el modelo con menor `accuracy` es el ZeroR, seguido del árbol de decisión. El SVM, K-nn y MLP son los modelos con mayor `accuracy` en los tres casos. En el resto de métricas se obserba que el SVM y el MLP tienen un mejor desempeño que el K-nn, siendo estos dos modelos los que mejores resultados obtienen. El árbol de decisión y el ZeroR son los modelos que peor desempeño tienen. El Random Forest y el árbol de decisión no obtienen tan buenos resultados como los otros modelos inicialmente pero, como vimos en la práctica 1, aplicando la poda coste-complejidad obteniamos una gran mejora de su desempeño, ya que con esta poda se conseguía reducir el sobreajuste del modelo.

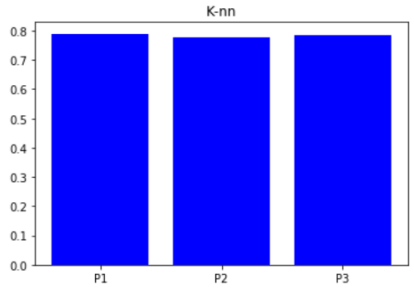
Para elegir que procesamiento utilizabamos en la práctica 1, nos decantamos por el procesamiento que en más modelos tuviese mayor `F1-Score`. En las siguientes gráficas mostramos el valor de esta métrica en los diferentes modelos para cada procesamiento:



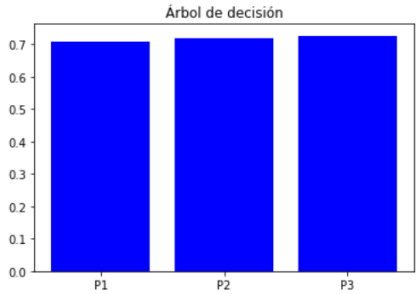
(m) ZeroR



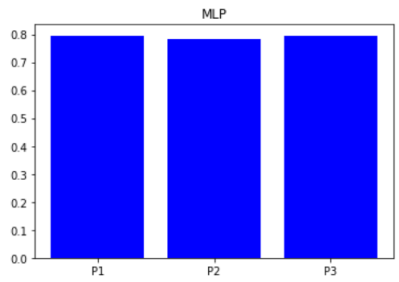
(n) SVM



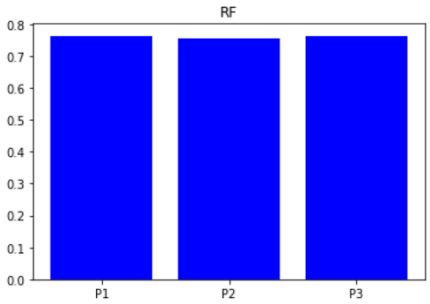
(ñ) k-nn



(o) Árbol de decisión



(p) MLP



(q) Random Forest

Como ya hemos visto antes, no hay gran diferencia entre unos y otros. En el modelo ZeroR la `F1-Score` es 0 en los tres casos, debido a que etiqueta todas las muestras como negaticas, luego no hay verdaderos positivos. En los modelos SVM, MLP, Random Forest y K-nn el procesamiento de datos 2 parece tener resultados ligeramente inferiores que los otros dos, que están muy igualados. En el árbol de decisión el procesamiento 2 tiene un mejor desempeño que el 1, pero peor que el 3. Este modelo es el que peores resultados ofrece sin considerar el ZeroR. Como podemos obsservar que el eje Y no llega a 0.8 como en los demás.

Al igual que en la práctica 1, el procesado de datos que usaría usando la información recogida en estas gráficas sería el procesado 3, porque en el árbol de decisión se ve que es el que mejores resultados ofrece y en el resto de modelos la diferencia con el procesado 1 no puedo apreciarla a simple vista.

Para crear las gráficas primero se crearon para cada modelo vector que contiene el valor de la métrica `F1-Score` de cada uno de los preprocesados:

```
v_zeror = [p1_f1[0], p2_f1[0], p3_f1[0]]
v_svm   = [p1_f1[1], p2_f1[1], p3_f1[1]]
v_knn   = [p1_f1[2], p2_f1[2], p3_f1[2]]
v_arbol = [p1_f1[3], p2_f1[3], p3_f1[3]]
v_mlp   = [p1_f1[4], p2_f1[4], p3_f1[4]]
v_rf    = [p1_f1[5], p2_f1[5], p3_f1[5]]
```

Tras esto, para cada uno de los modelos creamos la gráfica correspondiente así:

```
fig, ax = plt.subplots()
ax.bar(["P1", "P2", "P3"], v_clf, color='blue')
ax.set_title("clf")
```

donde "clf" denota el clasificador del que recogemos los datos en la gráfica.

1.2. Curva ROC

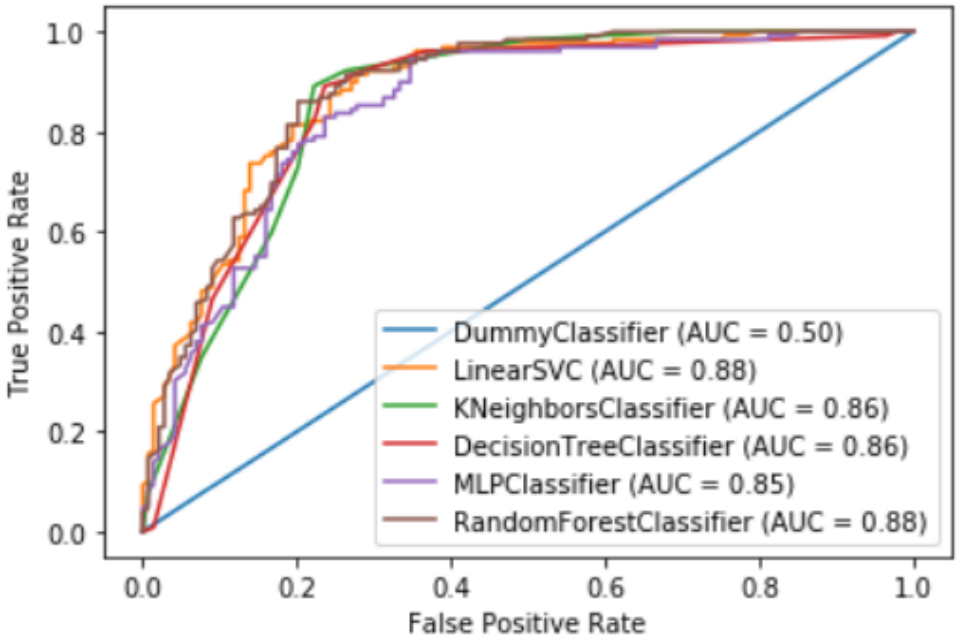
Para representar la curva ROC dividimos el conjunto de datos al que se le ha aplicado el procesado 3 salvo la normalización en comjunto de entrenamiento y conjunto de validación. La división se hace de manera que el 70 % de los datos formen el conjunto de entrenamiento y el otro 30 %, el de test, conservando la proporción de elementos en cada clase tanto en el conjunto de entrenamiento como en el de validación. Esto se hace con el código:

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(data, target,
test_size=0.3, stratify=target, random_state=0)
```

Tras esto,, completamos el procesado de datos 3 usando `MinMaxScaler()` para normalizar los datos de entrenamiento y se usa estas mismas transformaciones sobre los datos de validación. A continuación, entrené los modelos con los hiperparámetros seleccionados en la práctica anterior. Podemos representar en una gráfica la curva ROC de los diferentes modelos así:

```
ax = plt.gca()

for model in [dummy_clf, svm_clf, knn_clf, tree_clf, mlp_clf, rf_clf]:
    metrics.plot_roc_curve(model, data, target, ax=ax)
```

El resultado es:



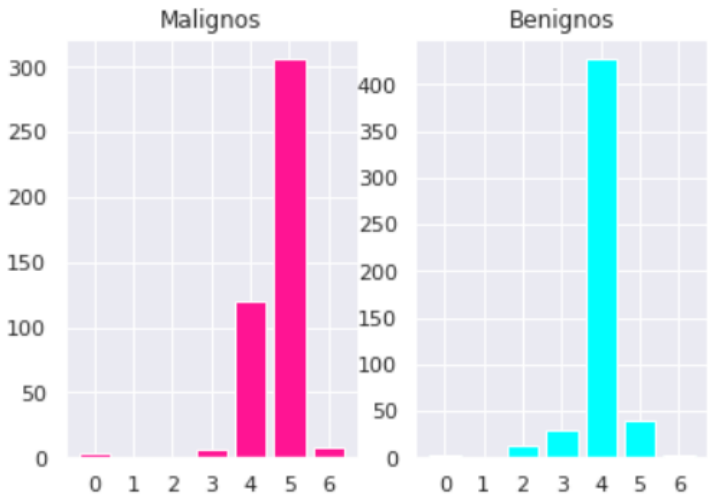
Curva ROC

Los modelos que presentan mayor métrica `AUC` son el Random Forest y el SVM, que son aquellos con mayor área bajo la curva ROC y los que tienen una mayor pendiente en los valores cercanos al cero. Esto implica que estos clasificadores son capaces de incrementar el número de verdaderos positivos a un ritmo mayor que el número de falsos positivos. Si usásemos esta métrica para sacar conclusiones, éstos serían los mejores modelos, mientras que el MLP es el modelo que peor comportamiento muestra, excluyendo al ZeroR. A pesar de esto, no hay excesivas diferencias entre los modelos considerados, a excepción del ZeroR.

1.3. Análisis de los atributos

En esta sección estudiaremos la importancia de los diferentes atributos en la clasificación. Para ello se visualizarán gráficos de barras para cada uno de los diferentes atributos y para cada una de las etiquetas, de manera que se muestre la distribución de los valores que toma dicho atributo en función de su etiqueta. También visualizaremos los diagramas de cajas, "boxplots", de aquellos atributos donde tenga sentido.

Empezamos mostrando las gráficas correspondientes al atributo `BI-RADS`:



Cantidad de datos con cada etiqueta

Este atributo representa la opinión de un médico experto sobre la gravedad del tumor. Si tiene el valor "1" o "2", el tumor es benigno, del mismo modo, si toma el valor "6" es maligno. Los valores "3", "4" y "5" designan casos en los que no se está seguro de la severidad del tumor, pero hay cierta probabilidad de que sea maligno o benigno. Esta información la podemos comprobar en https://es.wikipedia.org/wiki/BI-RADS. Este atributo da mucha información sobre la naturaleza del tumor, pero dado que necesitamos la opinión de un experto para obtenerlo, no es apropiado usarlo para el aprendizaje.

Según la página https://bigml.com/user/TotyB/gallery/dataset/509a98c6035d0706dd0001dd, de donde hemos obtenido los datos, el 94.46 % de estos tienen `BI-RADS` "4" o "5". De estos, los tumores con valor "5" son probablemente malignos. Los tumores malignos tienen más variabilidad que los benignos, dado que hay tumores malignos con valor de `BI-RADS` "4" o "5", predominando el valor "5". En el caso de los benignos, la mayoría de estos tienen la categoría de `BI-RADS` "4".

Esto se ve en el diagrama de cajas de este atributo:

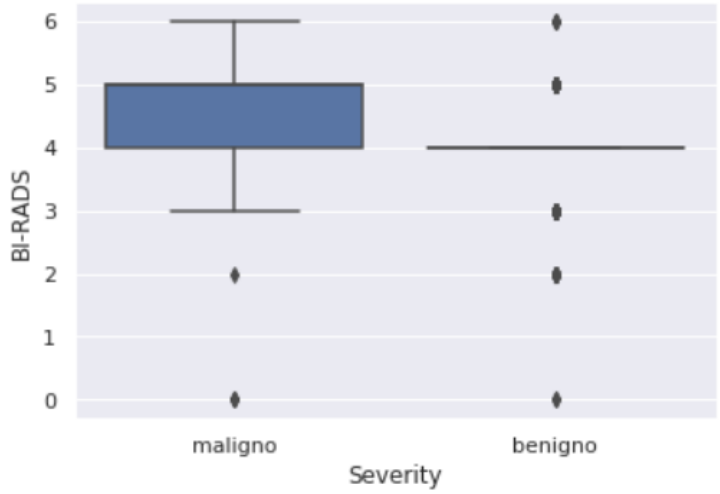
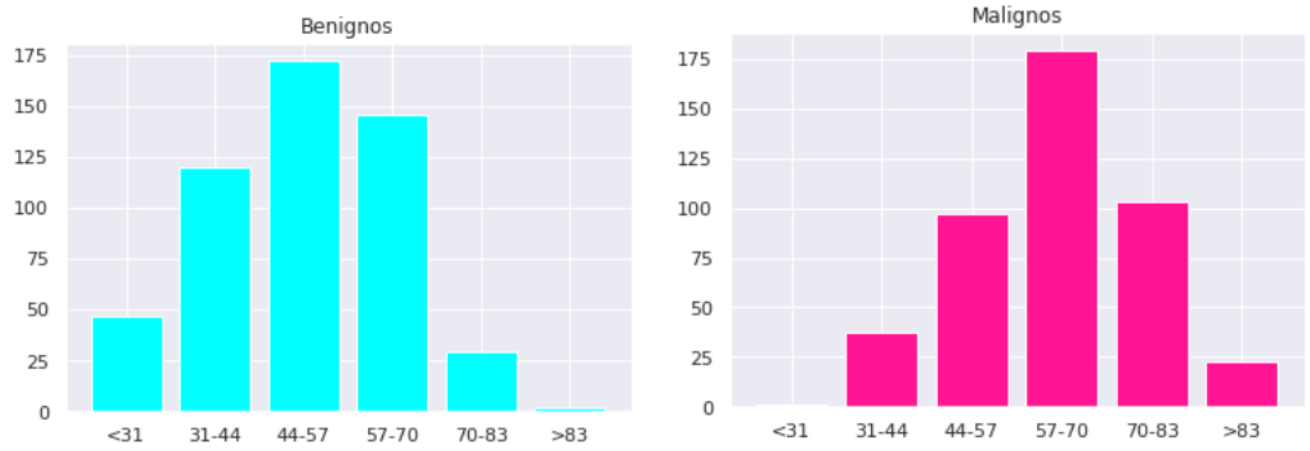


Diagrama de cajas de BI-RADS

Los tumores etiquetados como malignos se mueven entre el valor “4” o “5” mientras que la mayoría de los benignos toman el valor “4”.

La gráfica del atributo edad nos proporciona los siguientes resultados:



Cantidad de datos con cada etiqueta

Vemos que cuanto menor edad tiene una persona, más probable es que su tumor sea benigno. En particular, la mayoría de las personas menores de 31 años de la muestra tienen tumores benignos y gran parte de las que tienen más de 83 tienen tumores malignos. En el diagrama de cajas podemos observar que la mediana de edad de los pacientes con tumores malignos de la muestra es superior a los 60 años, mientras que la de los pacientes con tumores benignos está rondando los 50. Aunque la edad de los pacientes con tumores benignos varía entre los 18 y algo más de los 80 años, los datos entre el primer y el tercer cuartil se encuentran concentrados entre los 40 y 60. Los datos entre el primer y el tercer cuartil de los tumores malignos se encuentran entre algo más de los 50 y algo más de los 70 años.

Por consiguiente, la edad de una persona sí es un atributo relevamte en la clasificación, porque aunque no podríamos estimar la severidad del tumor sabiendo sólo la edad de ésta, si una persona es muy joven podríamos esperar que su tumor sea benigno.

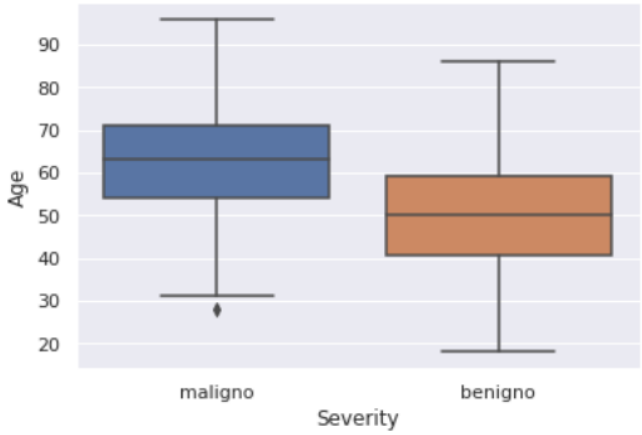
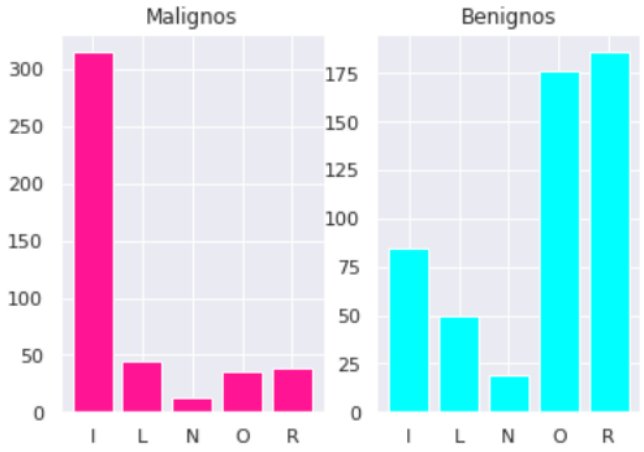


Diagrama de cajas de la edad

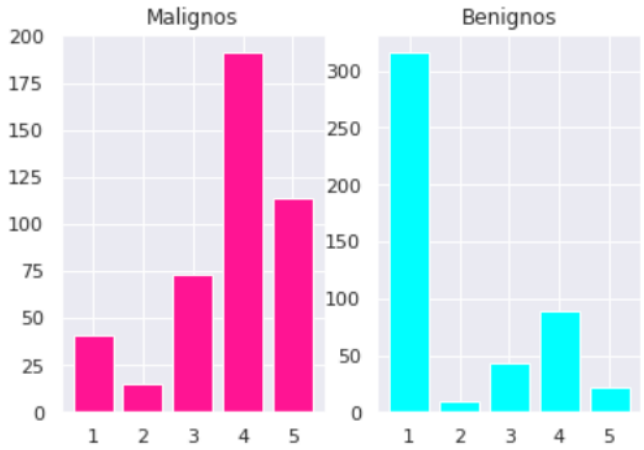
A continuación, analizaremos las gráficas del atributo Shape. La mayoría de los tumores malignos tienen una forma irregular, mientras que los benignos suelen tener una forma ovalada o redondeada. Los tumores benignos presentan mayor variabilidad en los valores que toman en este atributo que los malignos. Resulta interesante considerar este atributo para nuestro aprendizaje, ya que la distribución de los valores que toman los tumores malignos difiere bastante de la de los benignos.



Cantidad de datos con cada etiqueta

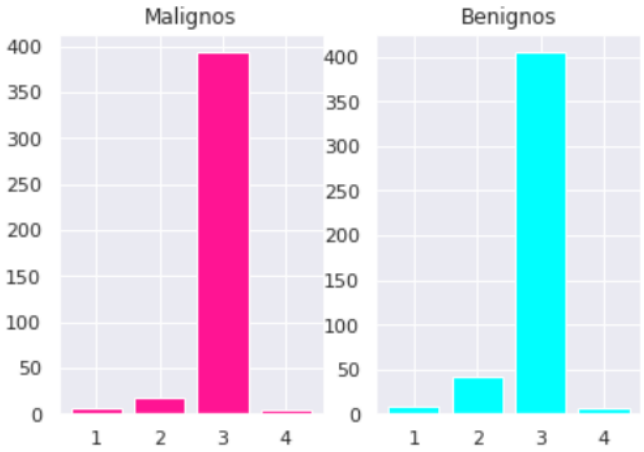
Como el atributo Shape es cualitativo y no tiene orden, no tiene sentido dibujar un diagrama de cajas en este caso, ya que este variaría para cada asignación diferente de valores numéricos a cada posible valor del atributo, y esta asignación es totalmente aleatoria, ya que estos carecen de orden.

El siguiente atributo a analizar es Margin. Al igual que en Shape, los distintos valores que toma este atributo no tienen un orden determinado, por lo tanto no representé el diagrama de cajas de este atributo. Los tumores malignos tienen mayor variabilidad que los benignos, siendo el valor de Margin más frecuente entre estos "ill-defined" seguido del "spiculated". Los benignos, sin embargo, suelen tener margin "circumscribed". En la práctica 1, una vez eliminados los atributos `BI-RADS` y Density, vimos que este atributo era el que mayor información nos aportaba en la clasificación, dado que era el elegido en el nodo raíz del árbol de decisión. Aquí podemos comprobar que efectivamente hay una gran diferencia entre la distribución de los valores de Margin que toman los tumores malignos de los benignos, cosa que afecta favorablemente a la clasificación, permitiéndonos diferenciar una muestra maligna de una benigna con mayor facilidad.



Cantidad de datos con cada etiqueta

Por último, analizamos el atributo Density. Como ya comentamos, este atributo tiene muy poca variabilidad en la muestra y podemos comprobar que los dos diagramas de barras son muy parecidos, lo que nos incita a pensar que este atributo no aporta apenas información a la clasificación.



Cantidad de datos con cada etiqueta

En el diagrama de cajas volvemos a apreciar la poca variabilidad de este atributo, ya que cualquier valor distinto de 3 lo interpreta como outlayer. Esta razón fue la que me llevó a considerar eliminar este atributo durante el procesado de datos 3.

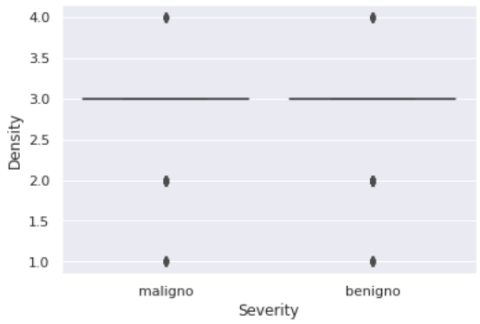


Diagrama de cajas de Density

2. Segmentación

2.1. Introducción

El problema que abordaremos en esta sección es agrupar accidentes obtenidos de los datos publicados por la Dirección General de Tráfico en conjuntos de accidentes con características similares. Para ello utilizaremos técnicas de clustering estudiadas en la asignatura, concretamente `K-means` y `DBSCAN`. Ambos métodos son métodos de particionamiento.

El algortimo `K-means` necesita como parámetro el número de clústers deseado y va clasificando las diferentes instancias en el clúster cuyo centroide sea el más cercano a ésta. Es un método iterativo que recalcula el centroide en cada iteración y repite el proceso de clasificación de las instancias. Termina cuando en una iteración ninguna instancia cambia de clúster. El algoritmo siempre converge.

El algoritmo `DBSCAN` depende de los parámetros *epsilon* y *min_samples*. El primero indica la distancia máxima a la que se considera que dos puntos son directamente alcanzables. Para que un punto sea un punto núcleo necesita que haya al menos *min_samples* a distancia menor que *epsilon* de él. A partir de ahí, un punto será alcanzable si existe una secuencia de puntos $p_1, \ldots, p_n$ tales que p_{i+1} es directamente alcanzable desde p_i. Cada punto núcleo forma un clúster formado por todos aquellos puntos alcanzables desde él. A los puntos que no pertenecen a ningún clúster se les asigna el clister -1.

Los atributos en los que nos basamos para hacer la segmentación son:

- `TOT_VICTIMAS`: Total de víctimas
- `TOT_MUERTOS`: Total de muertos
- `TOT_HERIDOS_GRAVES`: Total de heridos graves.
- `TOT_HERIDOS_LEVES`: Total de heridos leves.
- `TOT_VEHICULOS_IMPLICADOS`: Total de vehículos involucrados en el accidente.

Las métricas que usaremos para la estimar la bondad del ajuste son el coeficiente Silhouette y el índice Calinski-Harabasz.

El coeficiente Silhouette mide como de similares son los elementos del mismo clúster comparado con los elementos de otro clúster. Este coeficiente es la media de los $s(i)$, con

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

donde $b(i)$ es la mínima distancia del elemento i a otro clúster y $a(i)$ es la distancia media del elemento al resto de instancias de su clúster. Cuanto mayor sea este coeficiente, mejor es el rendimiento del modelo.

El índice Calinski-Harabasz mide la razón entre la dispersión interclústers y la dispersión intraclústers. El agrupamiento será mejor cuanto mayor sea este índice.

Para la elección de hiperparámetros, elegiremos aquellos que maximicen el índice Calinski-Harabasz, que utiliza todas las muestras para su cálculo y dado que usamos los mismos datos en todas las iteraciones, que su valor no esté normalizado no afecta en la interpretación del resultado.

Antes de realizar el análisis ejecutando los algoritmos de clusterig normalizamos los datos con la función `MinMaxScaler`, cuyo comportamiento es el mismo que la función `norm` de `pract2_utils`, por lo que es correcto usar `denorm` con los datos normalizados para deshacer esta operación (como hacemos para representar los centroides).

2.2. Caso de estudio 1: Choque frontal en carreteras convencionales

Estudiaremos las propiedades de los choques frontal en carreteras convencionales que, según el libro Manual del alumno. Permiso B es el caso de accidente con más víctimas mortales.

2.2.1. K-means

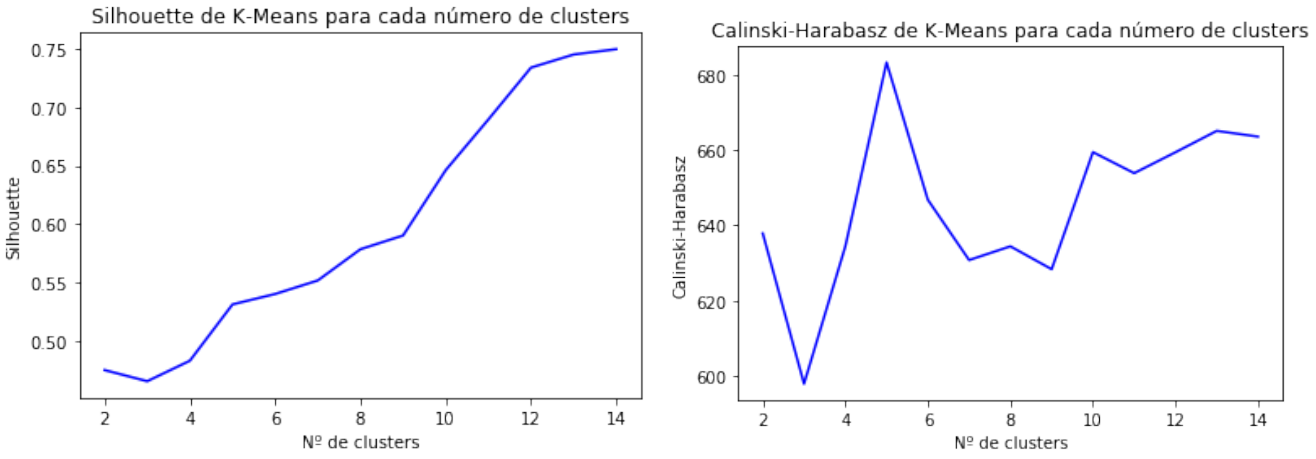
El algoritmo `K-means` tiene como parámetro el número de clúster a considerar. Para elegir el mejor valor de éste, ejecuté el algoritmo con un número de clústers comprendido entre 2 y 14. Los resultados de las métricas son:

Nº de clústers	Silhouette	Calinski-Harabasz
2	0.4755151052183165	637.8831140088855
3	0.46599969633063965	597.9798532808658
4	0.48348908673333896	634.4016355078425
5	0.531569055229591	683.2263367508311
6	0.5404576589330072	646.8004817570682
7	0.5519986909803548	630.8036708742871
8	0.5786471085948179	634.4445453990445
9	0.5903779353773022	628.3756258948337
10	0.6461326222918197	659.4342926638305
11	0.6893747693555886	653.852874658656
12	0.7336714808160643	659.3965542394021
13	0.744841516853055	665.0690582901761
14	0.749410469848095	663.5502144820157

Para facilitar la comprensión de la tabla, visualizamos la información que contiene en las siguientes gráficas, que se realizan utilizando la función `grafica`, cuyo código es:

```
def grafica(data, label, title, xlab, ylab):
    plt.plot(data,label, c='b')
    plt.title(title)
    plt.xlabel(xlab)
    plt.ylabel(ylab)

    plt.show()
```

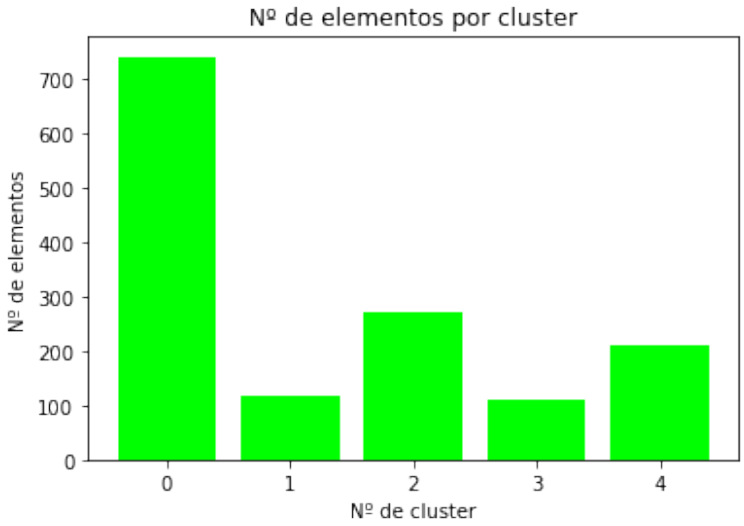
Obtenemos el siguiente resultado:



Gráficas con el valor de las métricas en función del número de clústers

Vemos que el coeficiente Calinski-Harabasz alcanza un máximo cuando se consideran 5 clústers, por lo que elegiremos este valor para dicho hiperparámetro.

El resultado de la segmentación es la división del conjunto de accidentes considerados en 5 conjuntos con las siguiente distribución:



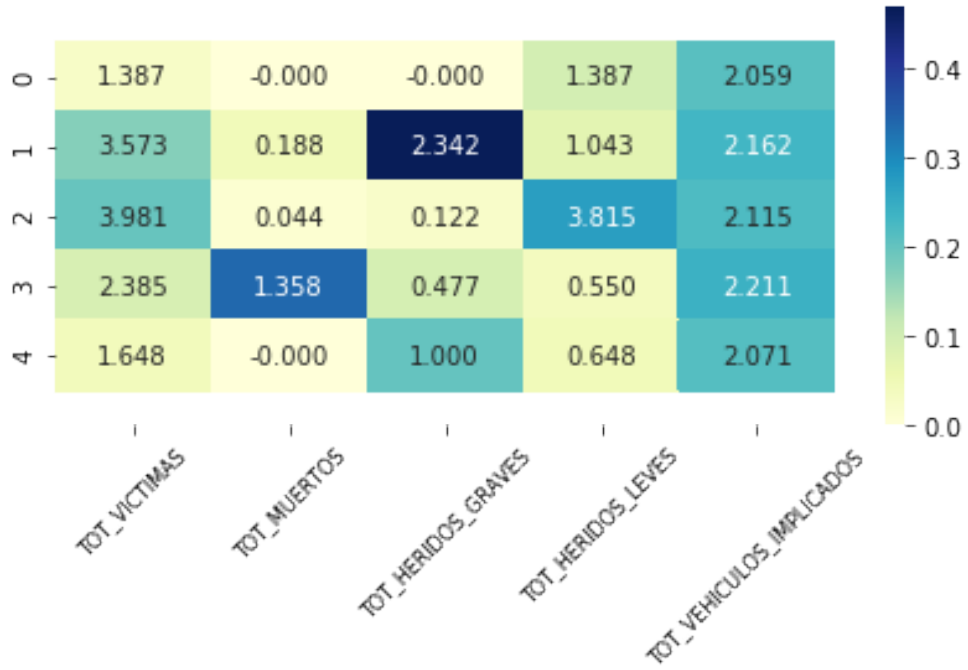
Clústers

Vemos que el 51.24 % de los accidentes pertenecen al clúster 0. El siguiente clúster más grande, el 2, posee el 18.65 % de éstos y el 4 posee el 14.51 %. Los clústers con menos elementos son el 1 y el 3 con el 8.08 % y el 7.53 %, respectivamente.

Los valores de las métricas para este modelo son:

Silhouette	**Calinski-Harabasz**
0.531569055229591	683.2263367508311

Los centroides de los diferentes clusters son:



Centroides

El clúster 0, que es aquel con más instancias, tiene como centro un accidente con 2.059 vehículos implicados, del que 1.387 personas han sido heridos leves, pero sin muertos ni heridos graves. Es el clúster cuyo centro tiene un menor número de víctimas.

El clúster 1 tiene como centro un accidente con $3,573$ víctimas, en el que $2,342$ son heridos graves, 1.043 son heridos leves y 0.188 son muertos. Hubo 2.162 vehículos implicados. Es el clúster con mayor número de heridos graves de media.

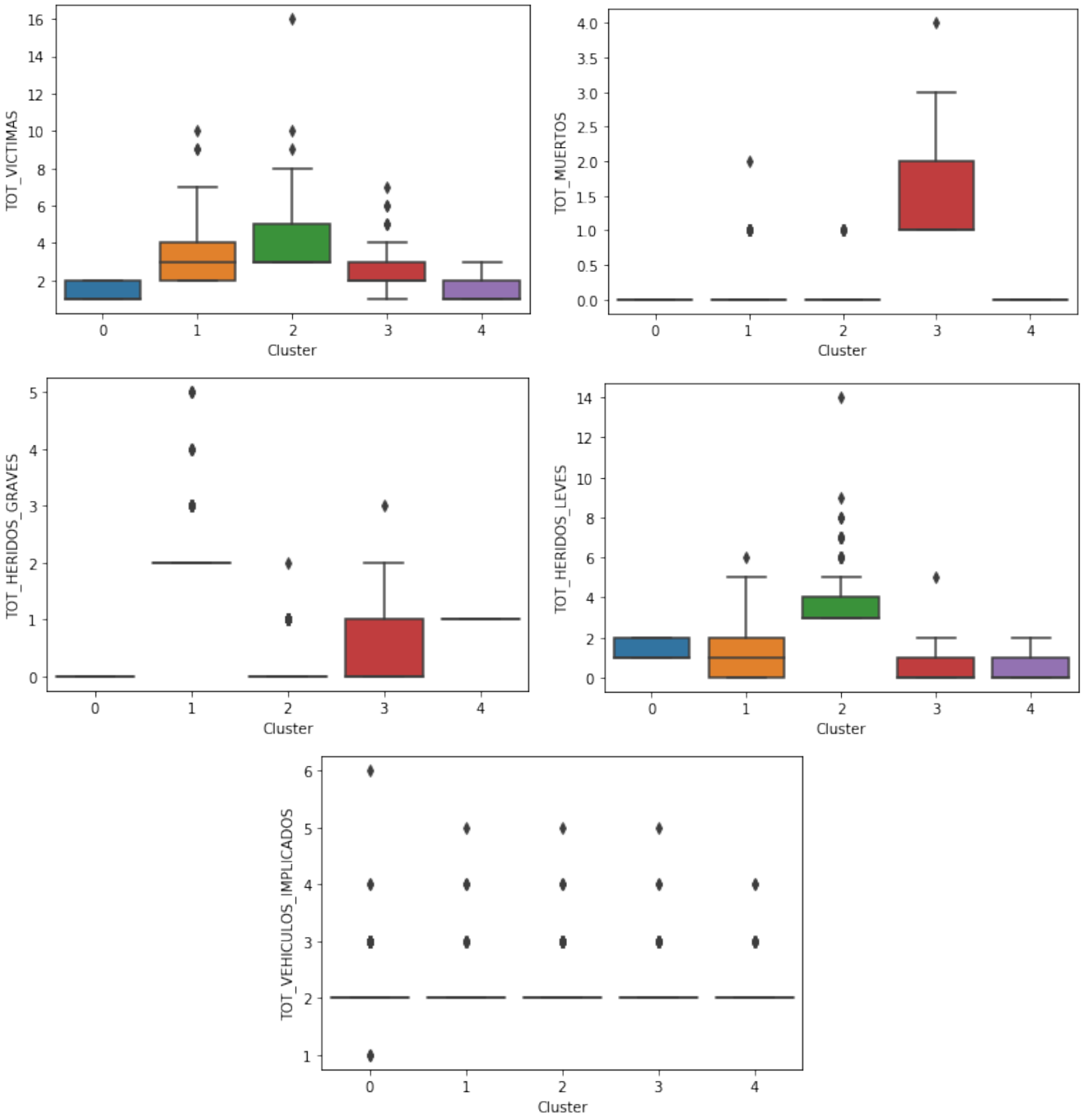
El clúster 2 tiene como centro un accidente con 3.981 víctimas, la mayoría heridos leves. Tiene de media más heridos leves que el resto, pero es el sugundo con menos heridos graves. Como en el resto de clústers, los vehículos involucrados están alrededor de 2.

El clúster 3 tiene como centro un accidente con 2.3850 víctimas. Tiene de media más muertos que el resto, 1.358 muertos. También tiene 0.477 heridos graves y 0.55 heridos leves.

El cúster 4 tiene como centro un accidente con 1.648 víctimas, pero ningún muerto.

Las diferencias entre los distintos clústers pueden visualizarse con `pairplot`:

Podemos corroborar lo que vimos con los centroides, que todos los clústers suelen tener entorno a 2 vehículos implicados. En el atributo `TOT_HERIDOS_GRAVES` nos ayuda a discernir el clúster 1 de los clusters 2 y 3. El atributo `TOT_HERIDOS_LEVES` diferencia los clústers 4, 0 y 2. Para ver estas diferencias de manera más detallada podemos usar `boxplot`:



Diagramas de cajas

En la siguiente tabla mostramos el rango de valores más representativo de cada cluster:

Nº del clúster	Víctimas	Muertos	Heridos graves	Heridos leves	Vehículos implicados
0	1-2	0	0	1-2	2
1	2-7	0	2	0-5	2
2	3-8	0	0	3-5	2
3	0-4	1-3	0-2	1-2	2
4	0-3	0	1	1-2	2

Salvo outlayers, todos los clúster están compuestos por accidentes con dos vehículos implicados.

El clúster 0 está compuesto por accidentes sin muertos ni heridos graves, y entre uno y dos heridos leves. Es aquel con más datos y cuyos accidentes son menos graves.

El clúster 1 está formado principalmente por accidentes con dos heridos graves y entre cero y cinco heridos leves. Es aquel con mayor variabilidad de heridos leves.

El clúster 2 está compuesto mayoritariamente por accidentes sin muertos ni heridos graves y entre tres y cinco heridos leves. Es aquel con más víctimas, pero todas ellas son heridos leves.

El clúster 3 es el único cluster, salvo oultayers, en cuyos accidentes hay muertos, entre uno y cinco concretamente. hay entre cero y dos heridos graves y entre uno y dos leves.

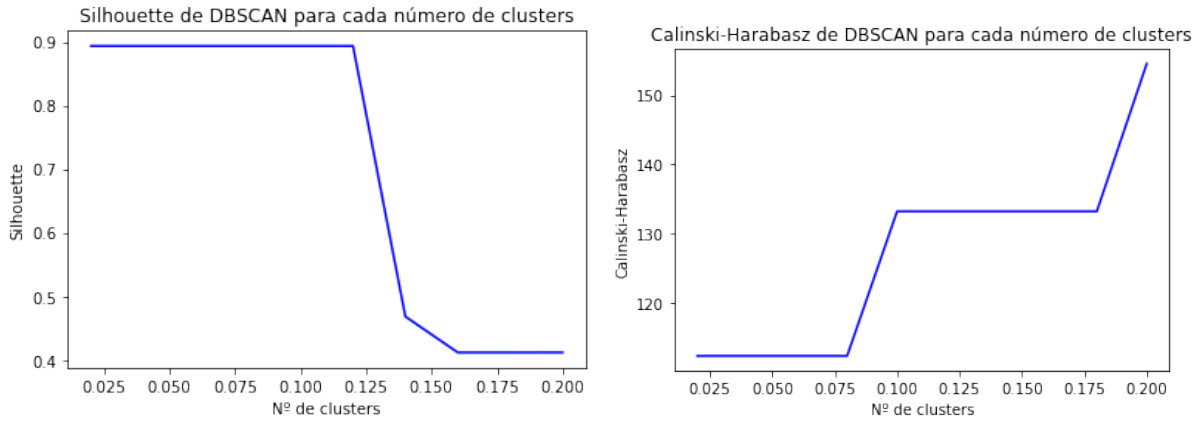
El cúster 4 se compone de accidentes con un herido grave y entre uno y dos leves.

2.2.2. DBSCAN

El algoritmo DBSCAN tiene como parámetro la distancia a la que consideramos que dos puntos son directamente alcanzables, *epsilon*. Consideré que un clúster tenía que estar constituido por 5 puntos mínimo. Para elegir el mejor valor de *epsilon*, ejecuté el algoritmo con diez valores de *epsilon* equidistantes entre 0.02 y 0.2, ambos incluidos. Los resultados de las métricas son:

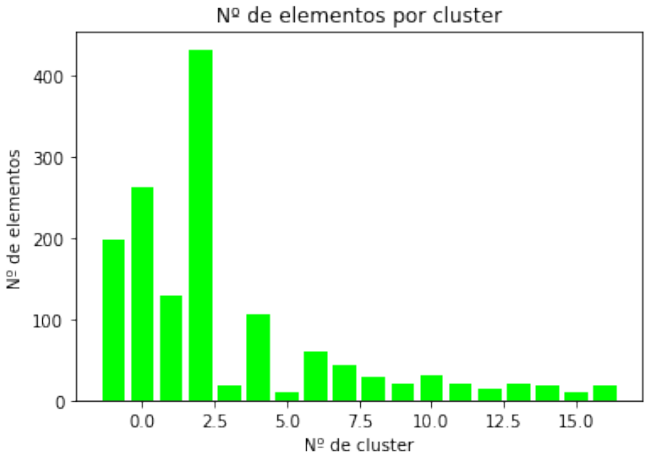
Nº de clústers	Silhouette	Calinski-Harabasz
0.02	0.8625282937916661	112.37177969144075
0.04	0.8625282937916661	112.37177969144075
0.06	0.8625282937916661	112.37177969144075
0.08	0.8625282937916661	112.37177969144075
0.1	0.5025984766300052	133.23030691481816
0.12	0.5025984766300052	133.23030691481816
0.14	0.5025984766300052	133.23030691481816
0.16	0.5025984766300052	133.23030691481816
0.18	0.5025984766300052	133.23030691481816
0.2	0.5221811203009754	154.55497718151375

Para facilitar la comprensión de la tabla, visualizamos la información que contiene en las siguientes gráficas:



Gráficas con el valor de las métricas en función del número de clústers

Elegimos el valor de *epsilon*= 0.08, ya que ambas medidas alcanzan un máximo con este valor. Los clúster formados quedan así:



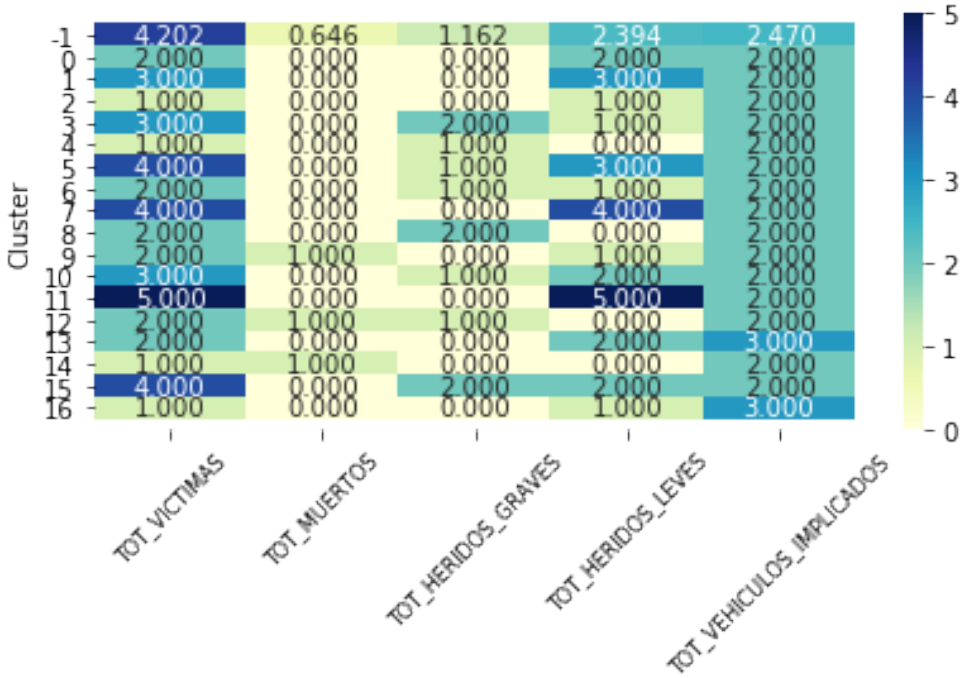
Gráficas con el valor de las métricas en función del número de clústers

Vemos que se han formado 18 cluster, siendo el clúster con identificador -1 aquel compuesto por todos los datos que no encajaban en ningúno de los otros 17. Hay 198 datos pertenecientes a este clúster, es decir, el 13.67 % de los datos no se clasifica correctamente. El clúster 2 tiene más instancias que el resto, seguido del clúster 0. Los clústers 1 y 4 también tienen más de 100 instancias. El resto de clúster no superan las 60.

Los valores de las métricas para este modelo son:

Silhouette	**Calinski-Harabasz**
0.8625282937916661	112.37177969144075

Los centroides de los diferentes clusters son:



Centroides

El clúster 0 tiene como centro un accidente con 2 vehículos implicados, en el que 2 personas han sido heridos leves, pero sin muertos ni heridos graves.

El clúster 1 tiene como centro un accidente con 3 víctimas, todas ellas heridos leves, sin muertos ni heridos graves. Dos vehículos se han visto involucrados.

El clúster 2 tiene como centro un accidente con 1 víctima, herida levemente, y dos vehículos implicados.

El clúster 3 tiene como centro un accidente con 3 víctimas, en el que 2 son heridos graves, y un herido leve. Hubo dos vehículos implicados.

El clúster 4 tiene como centro un accidente con dos vehículos implicados con un herido grave.

El clúster 5 tiene como centro un accidente con 4 víctimas, en el que una ha sido herida gravemente y hay tres herido leve. Hubo dos vehículos implicados.

El clúster 6 tiene como centro un accidente con 2 víctimas, una herida grave y otra leve. Hubo dos vehículos implicados.

El clúster 7 tiene como centro un accidente con 4 víctimas, las cuatro heridas leves. Hubo dos vehículos implicados.

El clúster 8 tiene como centro un accidente con 2 víctimas, las dos heridas graves. Hubo dos vehículos implicados.

El clúster 9 tiene como centro un accidente con 2 víctimas, un muerto y un herido leve. Hubo dos vehículos implicados.

El clúster 10 tiene como centro un accidente con 3 víctimas, en el que uno es un herido grave y los otros dos heridos leves. Hubo dos vehículos implicados.

El clúster 11 tiene como centro un accidente con 5 víctimas, todas ella heridos leves. Hubo dos vehículos implicados.

El clúster 12 tiene como centro un accidente con 2 víctimas, un muerto y un herido grave. Hubo dos vehículos implicados.

El clúster 13 tiene como centro un accidente con 3 vehículos implicados, donde las 2 víctimas son heridos leves.

El clúster 14 tiene como centro un accidente con una víctima, un muerto. Hubo dos vehículos implicados.

El clúster 15 tiene como centro un accidente con 4 víctimas, dos heridas gravemente y dos heridos leves. Hubo dos vehículos implicados.

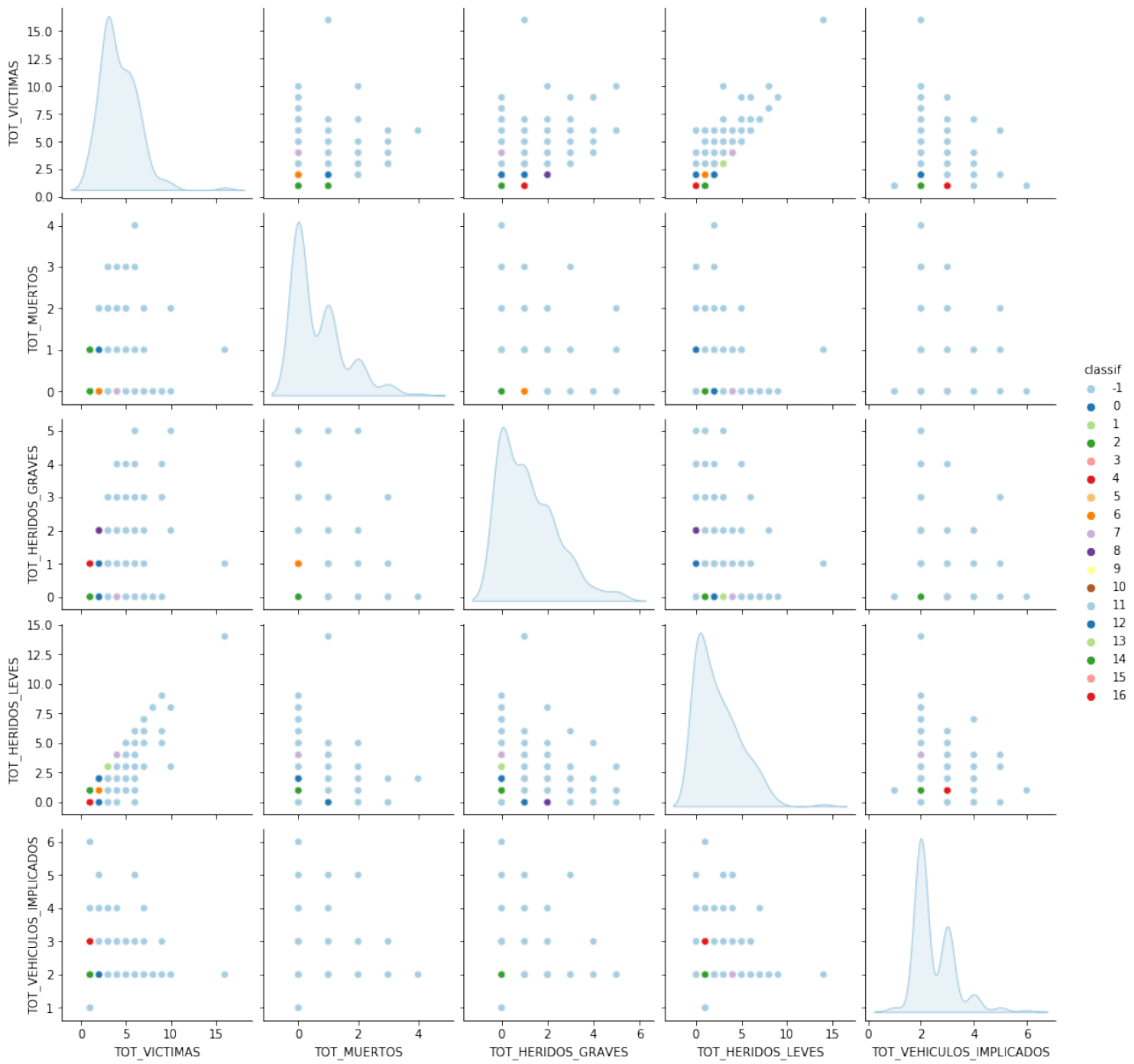
El clúster 16 tiene como centro un accidente con una víctima, un herido leve. Hubo tres vehículos implicados.

Para visualizar los centroides se usó el siguiente codigo:

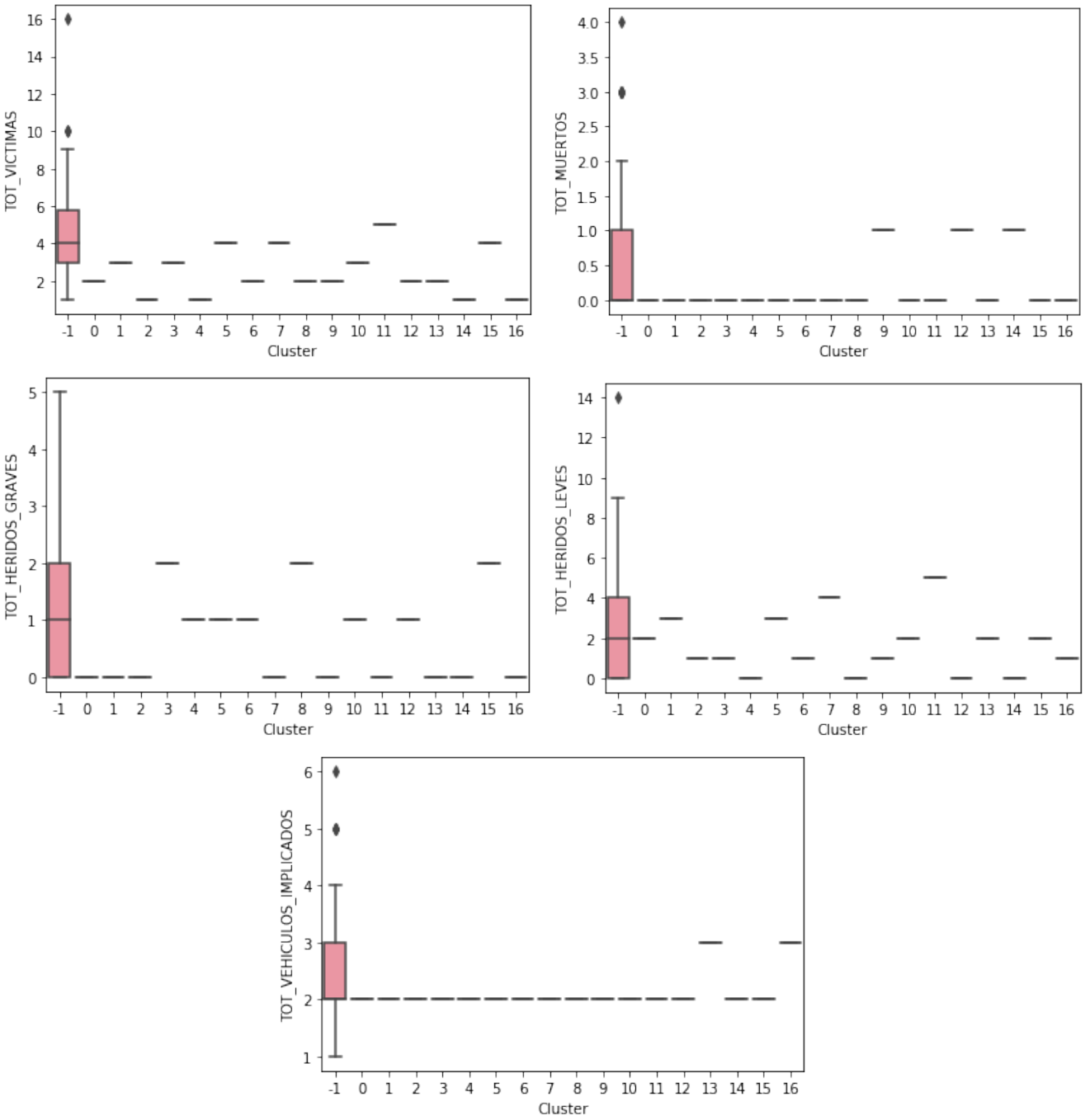
```
datos_cen = caso1.copy()
datos_cen.columns = atributos
datos_cen['Cluster'] = results.labels_

centroids = datos_cen.groupby('Cluster').mean()
visualize_centroids_dbscan(centroids, np.array(caso1), atributos)
```

Donde la función `visualize_centroids_dbscan` es análoga a la función `visualize_centroids`, pero no aplica `denorm`, ya que los centroides no se le pasan normalizados.

Como con K-Mean, utilizé `pairplot` con la esperanza de ver diferencias entre los distintos clústers, pero los resultados no son los esperados y no puedo deducir nada de este gráfico.



Representamos pues diagramas de cajas para poder ver mejor la variabilidad de los diferentes atributos en cada clúster y diferenciarlos así.



Diagramas de cajas

El clúster -1 es aquel con más variabilidad en todos los atributos por estar compuesto por todos aquellos accidentes que no encajaban en el resto de clústers. El resto de clústers tienen poca variabilidad en sus atributos, lo que nos puede indicar que hay demasiados clústers para el problema, y que el modelo está sobreajustado, ya que hay poca generalización.

En la siguiente tabla mostramos el rango de valores más representativo de cada cluster:

Nº del clúster	Víctimas	Muertos	Heridos graves	Heridos leves	Vehículos implicados
0	2	0	0	2	2
1	3	0	0	3	2
2	1	0	0	1	2
3	3	0	2	1	2
4	1	0	1	0	2
5	4	0	1	3	2
6	2	0	1	1	2
7	4	0	0	4	2
8	2	0	2	0	2
9	2	1	0	1	2
10	3	0	1	2	2
11	5	0	0	5	2
12	2	1	1	0	2
13	2	0	0	2	3
14	1	1	0	0	2
15	4	0	2	2	2
16	1	0	0	1	3

Debido a la poca variabilidad de los atributos en cada clúster, todos los elementos de cada clúster coinciden con el centro de éste, que fue expuesto anteriormente. A cualquier otro elemento que no encaje en las descripciones se le asigna el clúster -1.

2.2.3. Comparación de los algoritmos

Los resultados de los algoritmos considerados son:

Algoritmo	Silhouette	Calinski-Harabasz	Nº de clústers
K-Means	0.531569055229591	683.2263367508311	5
DBSCAN	0.8625282937916661	112.37177969144075	18

Vemos que el algoritmo K-Means obtiene un valor de la métrica Calinski-Harabasz más de seis veces superior del valor que obtiene DBSCAN con esa misma métrica. Sin embargo, DBSCAN obtiene un mayor valor en la métrica Silhouettte. Esto puede deberse a la poca variabilidad de los atributos en cada clúster, por lo que la ditancia entre ellos es muy reducida. El algoritmo Kmeans tiene por tanto mejor comportamiento que el DBSCAN, consiguiendo mejor valor en la métrica Calinski-Harabasz, y consigue agrupar los accidentes en cinco clusters claramente diferenciados, aumentando ligeramente la variabilidad de cada atributo en la cluster, pero disminuyendo el sobreajuste.

2.2.4. Interpretación de la segmentación

En la mayoría de accidentes hay dos vehículos implicados, por lo que el atributo `TOT_VEHICULOS_IMPLICADOS` no aporta mucha información. El atributo `TOT_VICTIMAS` tampoco es muy relevante, puesto que puede calcularse a partir de `TOT_MUERTOS`, `TOT_HERIDOS_GRAVES` y `TOT_HERIDOS_LEVES`. El atributo `TOT_MUERTOS` sirve para diferenciar uno o tres clúster en cada uno de los modelos considerados del resto, y el resto de clústers se diferencian entre sí por los atributos `TOT_HERIDOS_GRAVES` y `TOT_HERIDOS_LEVES`, por lo que estos son los atributos dos más relevantes para el agrupamiento.

2.3. Caso 2: Fin de semana

Nº del clúster	Víctimas	Muertos	Heridos graves	Heridos leves	Vehículos implicados
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

3. Contenido adicional

4. Bibliografía

Visualización

- https://es.wikipedia.org/wiki/BI-RADS
- Datos: https://bigml.com/user/TotyB/gallery/dataset/509a98c6035d0706dd0001dd

Segmentación

- Transparencias Tema 5: Clustering
- Manual del alumno. Permiso B
- https://github.com/SofiaAlmeida/IN/blob/master/P2/memoria.pdf
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN
- https://es.wikipedia.org/wiki/DBSCAN