Inteligencia de Negocio: Práctica 2
Visualización y Segmentación

David Cabezas Berrido

Grupo 2: Viernes

dxabezas@correo.ugr.es

25 de noviembre de 2020

Índice general

Parte I

Visualización

Sobre los resultados de la práctica 1, realizaremos representaciones de los resultados obtenidos por cada algoritmo y preprocesamiento, también de las relaciones entre los atributos. No nos centraremos en por qué unos modelos o preprocesamientos son mejores que otros, ni en detalles de los mismos (hiperparámetros configurados o por defecto. Eso ya lo hicimos en la práctica anterior. Nos centraremos cómo podemos interpretar las visualizaciones para ayudarnos a distinguir qué algoritmo o procesamiento es más adecuado en cada caso, o qué relaciones hay entre los atributos.

1. Visualización de medidas

Compararemos los scores de todos los modelos para cada preprocesamiento y viceversa. Las métricas que representaremos son la Accuracy y el F1-score.

1.1. Por procesamiento

El **primer procesamiento** consiste en **eliminar las instancias con valores perdidos**.

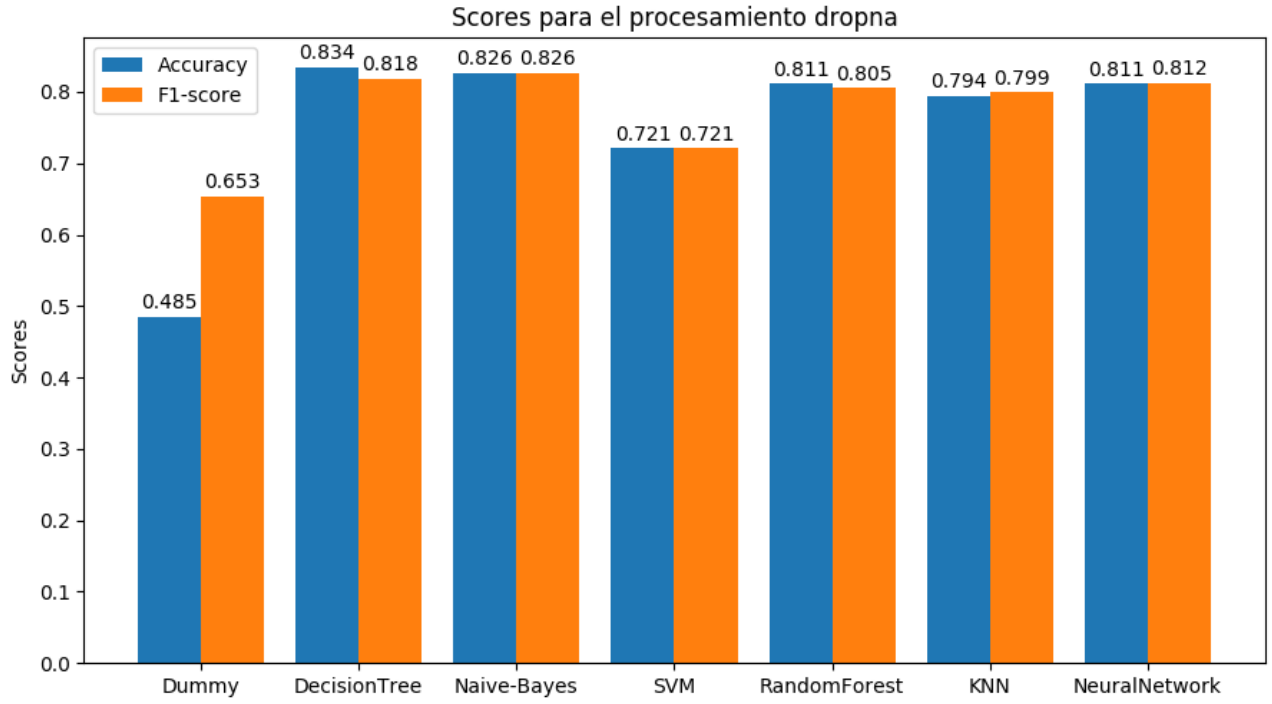


Figura 1: Desempeño de los algoritmos sobre el preprocesado 1: Eliminación de valores perdidos

La barra azul (Accuracy) representa el desempeño general del modelo, la proporción de instancias bien clasificadas en validación cruzada. Mientras que la barra naranja (F1-score) le da una mayor prioridad a la clasificación de instancias positivas, que tienen mayor importancia en este problema, puesto que un falso negativo es más grave que un falso positivo.

Los modelos que más score han obtenido para este preprocesamiento son el Árbol de Decisión y Naive-Bayes. El Árbol de Decisión tiene ligeramente mayor Accuracy, mientras que Naive-Bayes tiene más F1-score, esto significa que Decision Tree clasifica mejor las instancias negativas que Naive-Bayes, y éste último clasifica mejor las positivas.

SVM presenta resultados bastante peores al resto de algoritmos (obviando Dummy).

Generalmente, la Accuracy y la F1-score son similares. La excepción es Dummy, que clasifica bien todas las positivas pero ninguna negativa.

El **segundo procesamiento** consiste en **imputar los valores perdidos**, para las variables numéricas usamos la mediana y para las nominales, la moda.

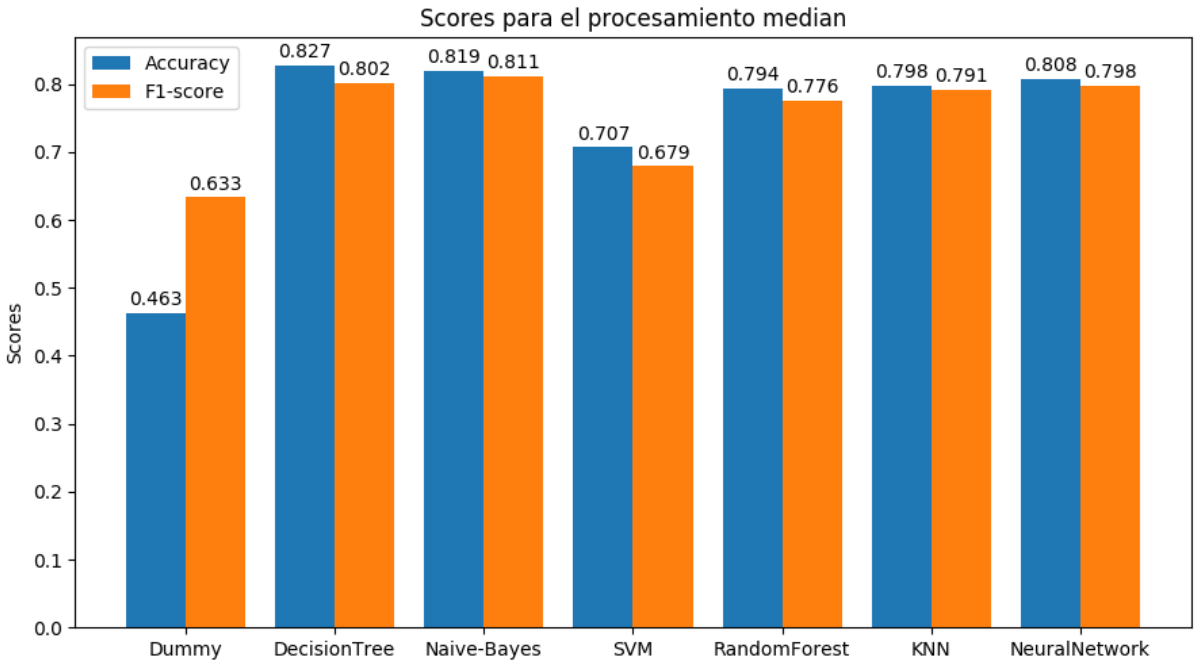


Figura 2: Desempeño de los algoritmos sobre el preprocesado 2: Imputación de valores perdidos

La comparación entre los modelos no ha cambiado mucho, salvo un descenso en el desempeño de todos los modelos. La gráfica nos ayuda a observar, comparando las barras azules con las naranjas, que el descenso ha sido mayor en la F1-score (la barra naranja ahora está más abajo que la azul en todos los algoritmos salvo Dummy), esto significa que, en general, ha empeorado la clasificación de instancias positivas.

El **tercer procesamiento** consiste en **simplificar el conjunto de atributos**. Concretamente, eliminamos Density y simplificamos la distribución de BI-Rads para que sólo tome los valores 4 y 5.

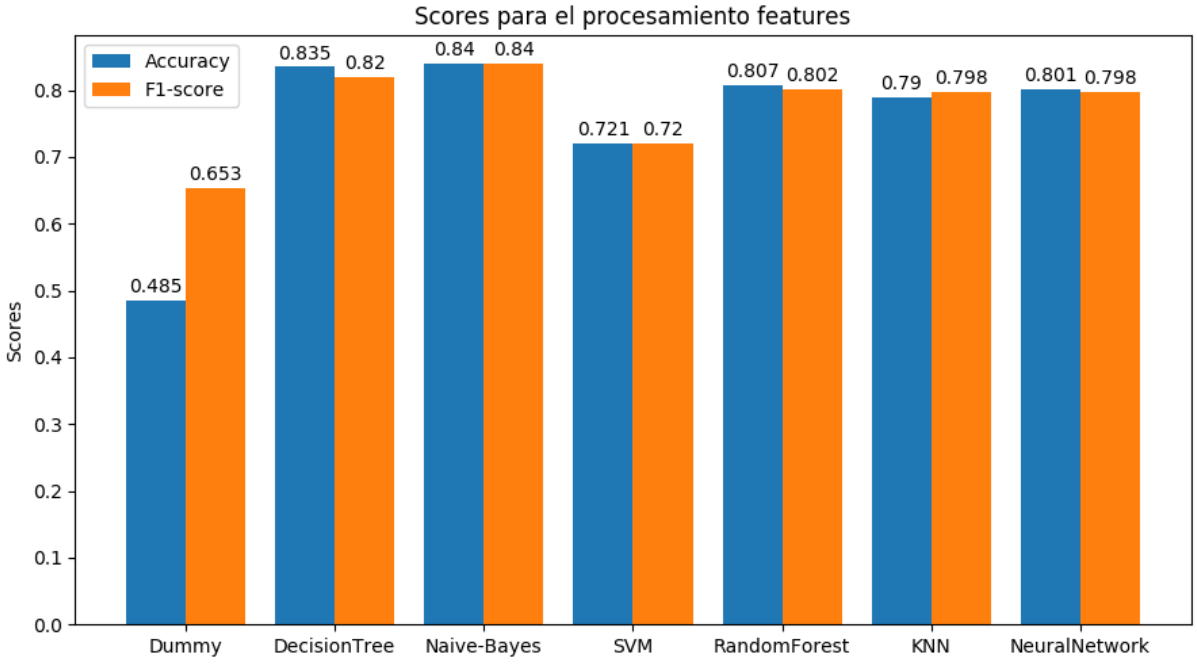


Figura 3: Desempeño de los algoritmos sobre el preprocesado 3: Eliminación de características innecesarias

Con este preprocesamiento, observamos que se ha recuperado la similitud entre los valores de F1-score y Accuracy, puesto que se hizo sobre los datos del preprocesado 1 y no sobre los del 2. También se ha producido una mejora general en la mayoría de modelos, sobre todo en Naive-Bayes por lo que comentamos en la práctica anterior de que supone los atributos independientes. Sin embargo, la comparación entre los modelos no cambia demasiado.

El **cuarto procesamiento** consiste en **binarizar las características nominales**, Shape y Margin. Esto lo hicimos porque no tenía sentido medir la distancia entre valores de una variable nominal si los numerábamos con números naturales, y algoritmos que no tratan bien las variables nominales como KNN y Neural Network podían verse afectados por esto.

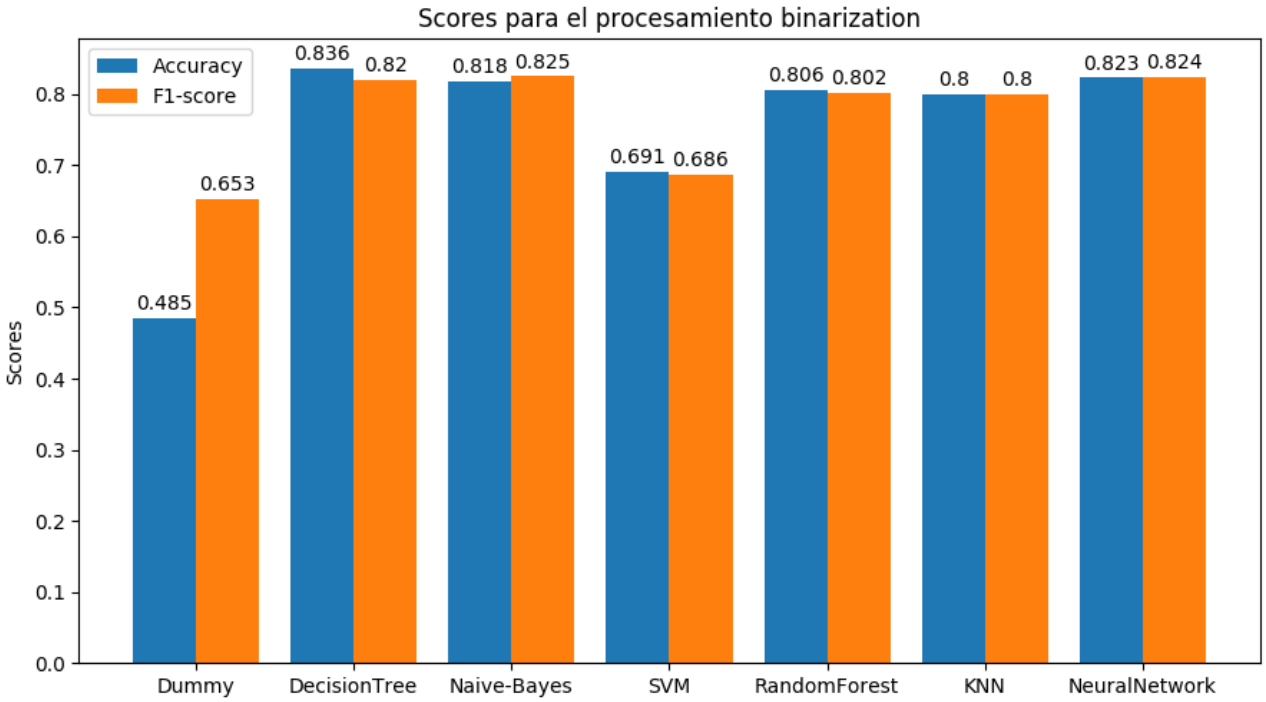


Figura 4: Desempeño de los algoritmos sobre el preprocesado 4: Binarización de características nominales

Se aprecia un descenso en Naive-Bayes, por haber introducido nuevas características que no son independientes unas de otras (cuando una vale 1, el resto de características binarizas asociadas a la misma variable nominal están forzadas a valer 0). La mejora en KNN es muy leve, pero sí destaca ahora una subida en la columnna de Neural Network.

Este procesamiento también perjudica aún más a SVM, aumenta más la diferencia entre su columna y el resto.

El **quinto procesamiento** consiste en **reescalar las variables**, para que tengan media 0 y varianza 1. Esto se hizo para solucionar la importancia exagerada que cobraba la variable Age en algoritmos como KNN.

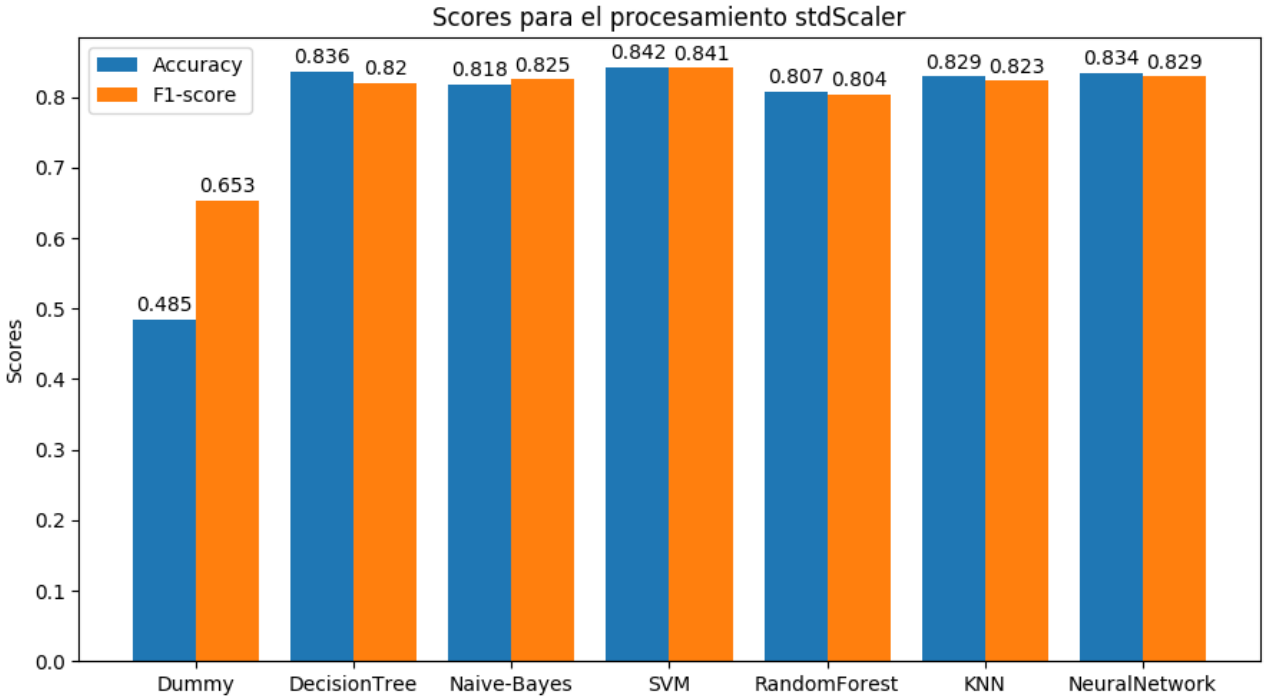


Figura 5: Desempeño de los algoritmos sobre el preprocesado 5: Estandarizado de los datos

Lo que más descata en esta gráfica es la importante subida de SVM, su columna sube hasta acabar ligeramente por encima del resto. También observamos una subida notable en el desempeño de KNN, cuya barra adelanta a la de Random Forest.

1.2. Por modelo

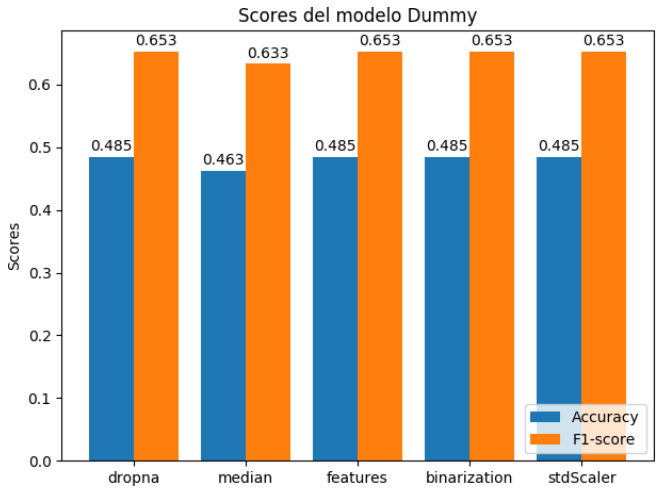


Figura 6: Desempeño de Dummy para cada preprocesamiento

No hay mucho que comentar de Dummy, consigue los mismos scores para todos los preprocesamientos, ya que ignora los datos de entrada y se limita a predecir siempre maligno. Cambia para el el segundo preprocesamiento, porque varía el número de instancias, ya que no eliminamos las instancias con valores perdidos.

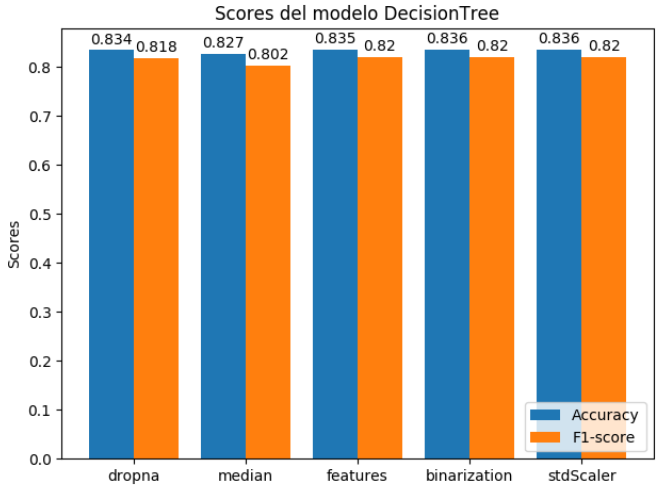


Figura 7: Desempeño de Decision Tree para cada preprocesamiento

La eficacia de Decision Tree no se vió afectada apenas por ninguno de los procesamientos que planteamos. Trabaja con las variables por separado, por lo que no le afecta el reescalado; trabaja bien con las variables nominales, por lo que no se ve beneficiado de la binarización de las mismas. Sí que observamos que este modelo acostumbra a clasificar mejor las instancias negativas, ya que su desempeño general (barra azul) está por encima de su F1-score (barra naranja), que da mayor importancia a los ejemplos positivos.

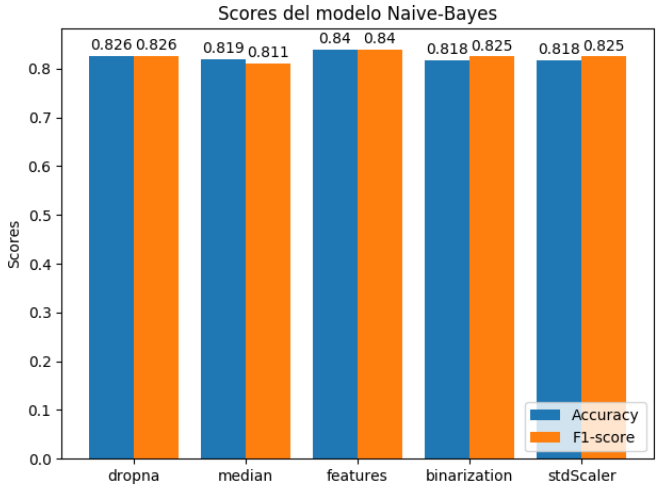


Figura 8: Desempeño de Naive-Bayes para cada preprocesamiento

Sobre este modelo percibimos que funciona mejor tras la simplificación del conjunto de características, pero empeora al introducir características nuevas que no son independientes cuando binarizamos las características nominales. Sobre algunos preprocesamientos ha tenido similar desempeño al clasificar instancias positivas y negativas (las barras azul y naranja están a la misma altura). Al igual que el resto de modelos, al imputar los valores perdidos pierde bastante efectividad sobre los ejemplos positivos (la barra naranja está más abajo que la azul). Al introducir las características dependientes en la binarización de variables nominales, decrece su eficacia sobre todo para los ejemplos negativos (la barra naranja no decrece tanto como la azul), y tampoco se ve afectado por el reescalado de los datos.

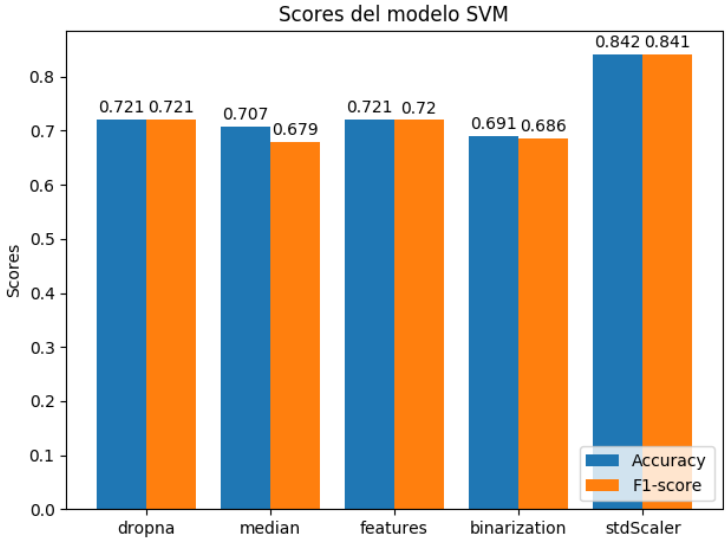


Figura 9: Desempeño de SVM para cada preprocesamiento

SVM presenta un desempeño bastante pobre hasta llegar al reescalado de las variables, cuando se convierte en el modelo que más score consigue. Se ve perjudicado por la binarización de características nominales y, exceptuando lo que les ocurre a todos los algoritmos con el segundo preprocesado (imputar valores perdidos), su desempeño es similar en las características positivas y en las negativas (las barras azul y naranja están muy igualadas).

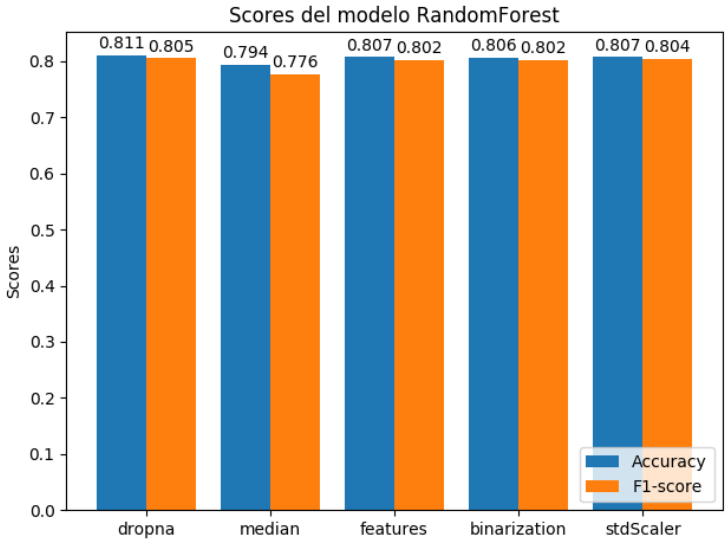


Figura 10: Desempeño de Random Forest para cada preprocesamiento

A Random Forest le ocurre lo mismo que a Decision Tree (Random Forest es un modelo que promedia varios Decision Tree), aunque su desempeño en general es peor. Hay un poco menos de diferencia entre las barras azul y naranja, por lo que no empeora tanto al clasificar instancias positivas.

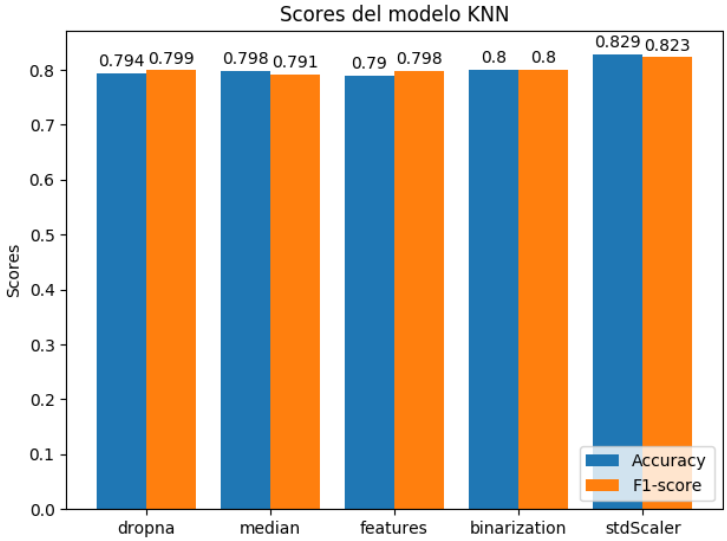


Figura 11: Desempeño de KNN para cada preprocesamiento

KNN se ve beneficiado por los dos últimos procesamientos, sobre todo con el último, el reescalado de las variables. Presenta resultados similares para todos los procesamientos, en comparación con los otros modelos, empeora poco con la imputación de valores perdidos. Su desempeño es similar para todos los preprocesamientos excepto para el último, donde la mejora es notable.

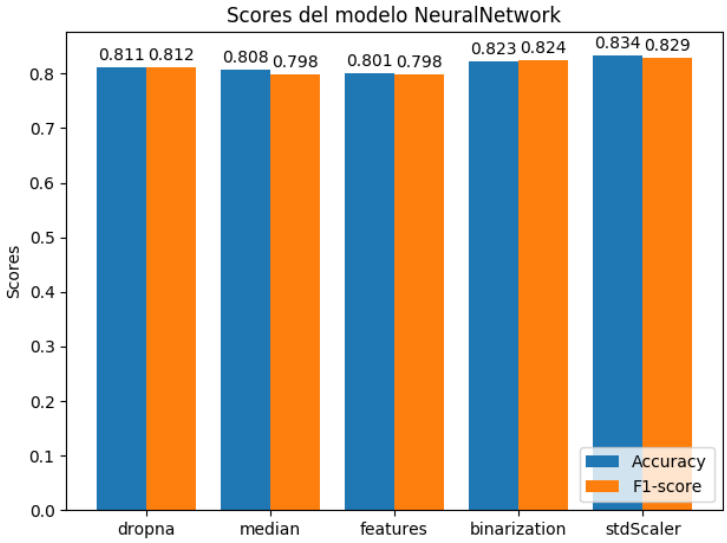


Figura 12: Desempeño de Neural Network para cada preprocesamiento

Neural Network se ve perjudicado por la simplificación del conjunto de características, probablemente porque es un modelo con una alta complejidad y capacidad explicativa y sea capaz de aprovechas la poca información que se pierde con esta simplificación. Los preprocesados de binarización y estandarización le benefician en gran medida. En general, su eficacia es similar en ejemplos positivos y negativos, las barras azul y naranja están prácticamente a la misma altura.

2. Gráficas de la curva ROC

Para cada preprocesamiento, compararemos los modelos representándolos en el espacio ROC. Los estamos tratando como modelos discretos (sólo atendemos a la clase que predigan, no la probabilidad que asignen de pertenencia a cada clase), por lo que no vamos a dibujar sus curva ROC como es habitual verlas, con forma de codo. Esto podríamos hacerlo con la función `roc_curve` de `sklearn.metrics`, pero tenemos que tener claro la interpretación continua que se hace del modelo para interpretar las curvas. Por ejemplo un árbol de decisión se puede interpretar como un modelo continuo de la siguiente manera: si una instancia cae sobre una hoja con p instancias de entrenamiento de la clase maligno y q de la clase benigno, puede asignar una probabilidad de $\frac{p}{p+q}$ para la clase maligno y una de $\frac{q}{p+q}$ para la clase benigno.

En su lugar, cada modelo aparecerá como un punto en el plano, concretamente en el intervalo $[0, 1] \times [0, 1]$.

El análisis gráfico por gráfico sería muy repetitivo, por lo que nos limitaremos a explicar cómo se interpretan estos gráficos. Lo haremos sobre el primero de ellos.

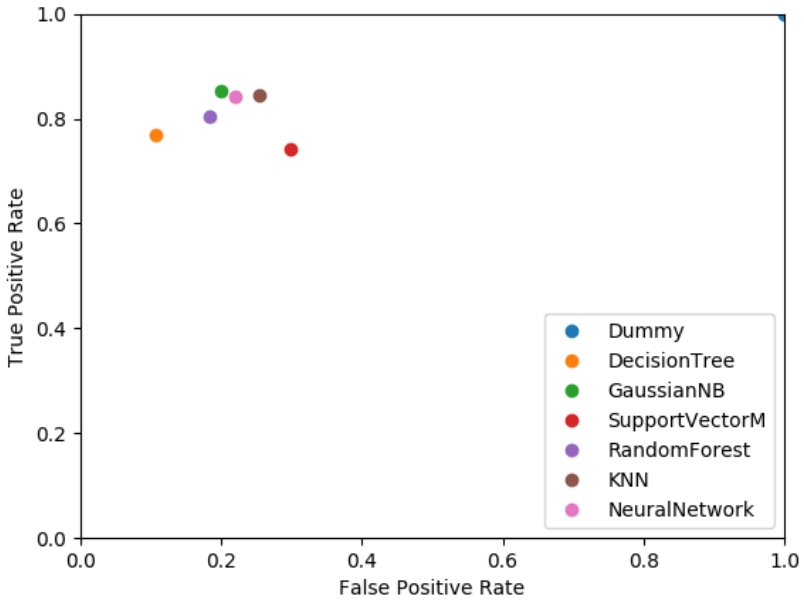


Figura 13: Algoritmos sobre el preprocesado 1 representados en el espacio ROC

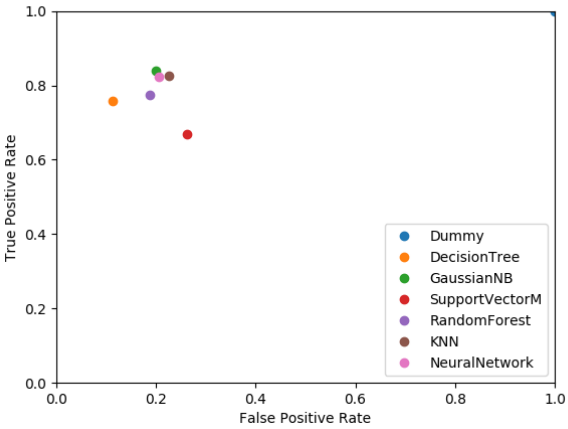
En el eje horizontal se representa el FPR, la tasa de falsos positivos (número de predicciones positivas incorrectas entre el número total de ejemplos negativos). En el eje vertical se representa el TPR, la tasa de verdaderos positivos (número de predicciones positivas correctas entre el número total de ejemplos positivos).

Cuanto más a la izquierda esté un punto, menor será la FPR del modelo, luego mejor clasificará ejemplos negativos. Cuanto más a arriba esté un punto, mayor será la TPR del modelo, luego mejor clasificará ejemplos positivos.

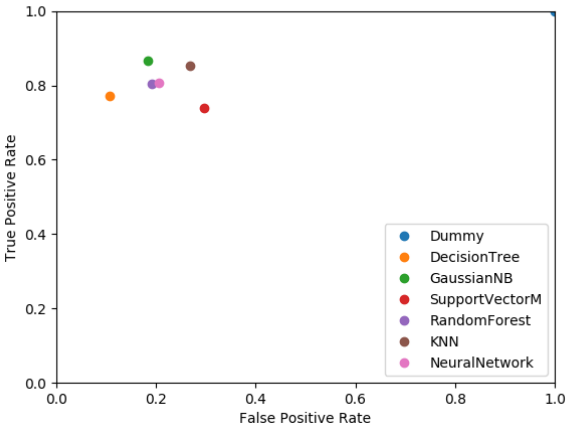
Si un modelo está situado arriba a la derecha de otro, significa que predice mejor tanto los ejemplos positivos como los negativos, por lo que podemos considerarlo mejor en cuanto a resultados, sin tener en cuenta la simplicidad ni la facilidad de interpretación de los modelos. De otra forma, dos modelos no son tan fácilmente comparables, por lo que tendríamos que atender a sus diferencias de FPR y TPR, a la importancia que le demos a cada una.

En este ejemplo concreto (Figura 13), Naive-Bayes es mejor que Neural Network, KNN y SVM. Obviamente Dummy supera a todos a la hora de clasificar ejemplos positivos, pero es el peor clasificando negativos. Naive-Bayes supera al árbol de decisión a la hora de clasificar instancias positivas, pero es peor clasificando instancias negativas. También apreciamos que SVM es el peor de todos sin contar Dummy.

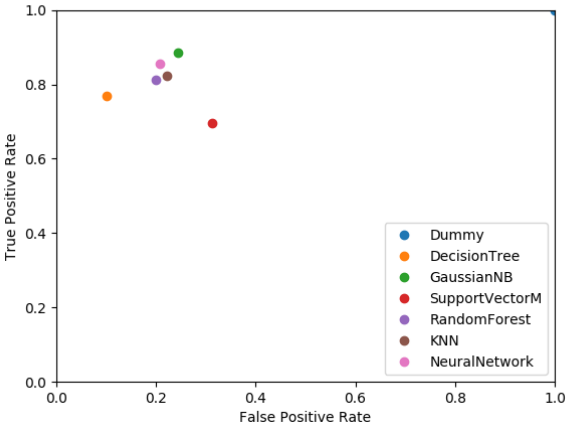
A continuación están las representaciones de los modelos en el espacio ROC para el resto de preprocesamientos.



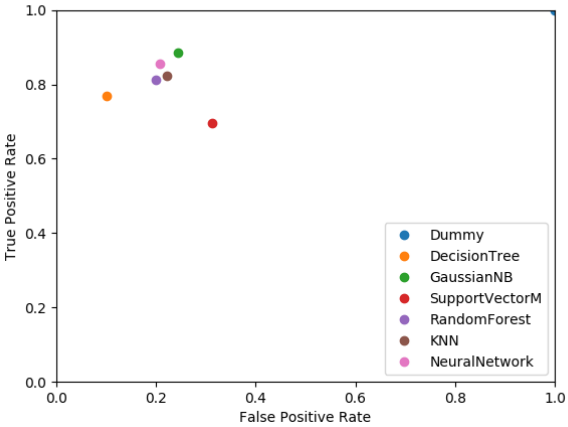
(a) Preprocesado 2: Imputación



(b) Preprocesado 3: Características



(c) Preprocesado 4: Binarización



(d) Preprocesado 5: Estandarización

Figura 14: Algoritmos sobre cada preprocesado representados en el espacio ROC

3. Análisis de atributos

Para cada atributo, representamos para cada posible valor el número de instancias positivas (en azul) y negativas (en naranja). Utilizamos un gráfico de barras apiladas, con el número de instancias malignas abajo. En este tipo de gráficos será fácil saber cuantas instancias malignas tienen determinado valor de una variable, y también cuantas instancias totales. Para ver el número de instancias benignas habrá que tener en cuenta donde empieza la barra.

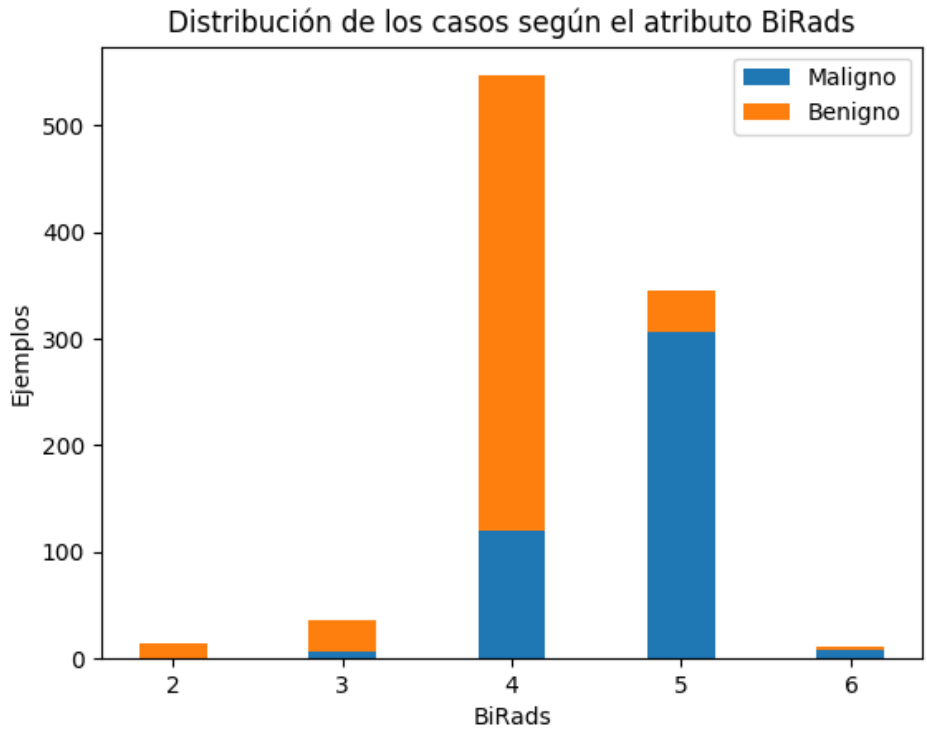


Figura 15: Distribución de ejemplos de cada clase para los distintos valores de BI-RADS

La mayoría de instancias tienen los valores 4 y 5. Al aumentar el valor de la variable, aumenta considerablemente la proporción de ejemplos malignos con ese valor. Por tanto, vemos que esta variable está altamente relacionada con la severidad del tumor.

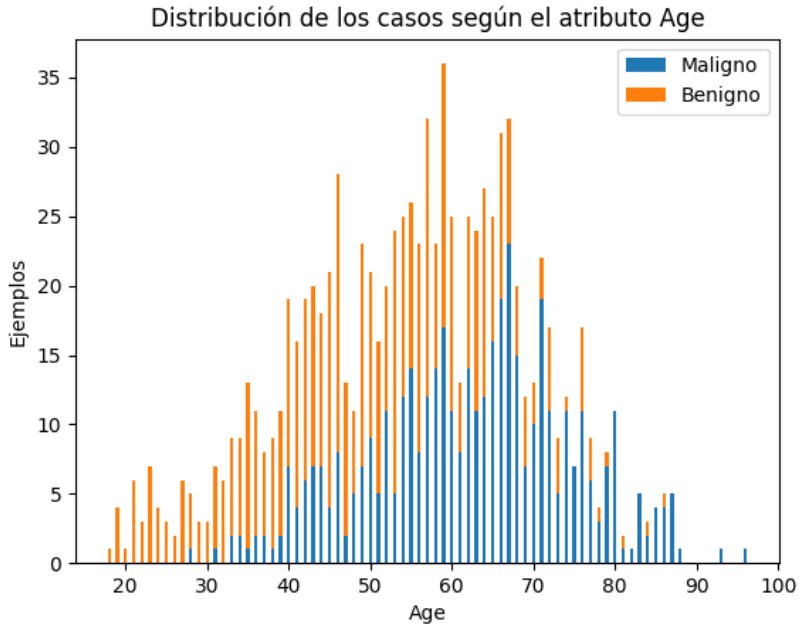


Figura 16: Distribución de ejemplos de cada clase para los distintos valores de Age

Al haber tantos valores es lógico que halla impurezas, pero al igual que antes, observamos un aumento en la proporción de ejemplos malignos para valores altos de la variable. Por lo que también concluimos que la edad está bastante relacionada con la severidad.

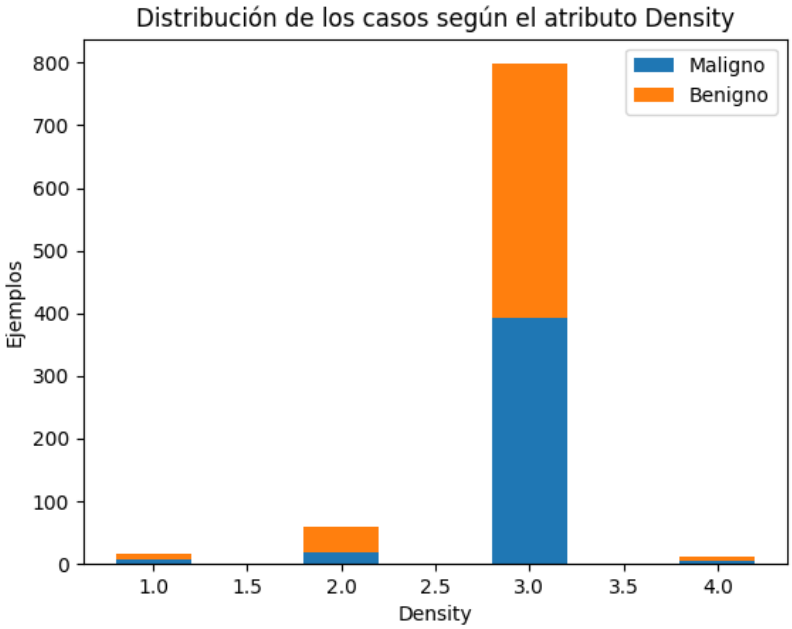


Figura 17: Distribución de ejemplos de cada clase para los distintos valores de Density

Observamos que la mayoría de las instancias presentan el valor 3 de esta variable. Además, para cada valor el número de instancias positivas y negativas que lo presentan es prácticamente el mismo. Por tanto, no parece que esta variable influya en la severidad del tumor.

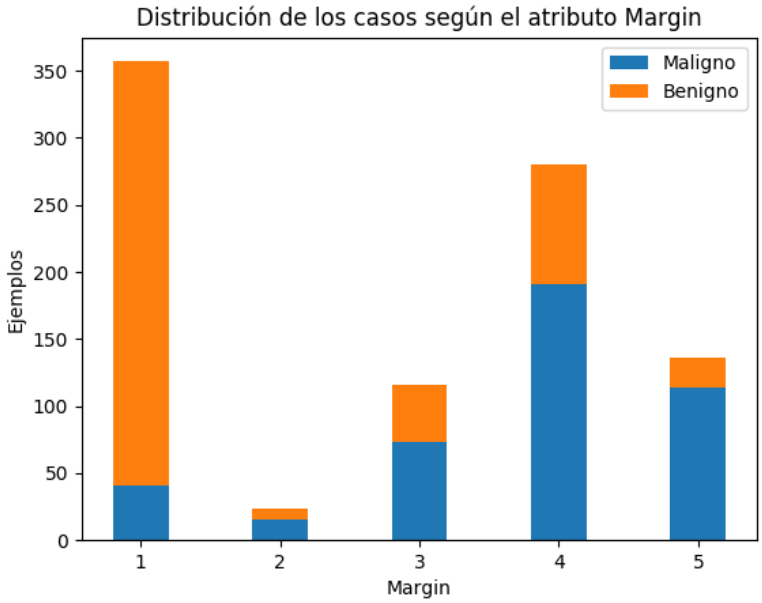


Figura 18: Distribución de ejemplos de cada clase para los distintos valores de Margin

La mayoría de las instancias que presentan el valor 1 corresponden a tumores benignos y la mayortía de las que presentan el valor 5, a malignos. Para el resto de valores (2, 3 y 4), la proporción de instancias malignas es mayor que la de benignas. Concluimos que este atributo sí puede aportarnos información sobre la severidad del tumor.

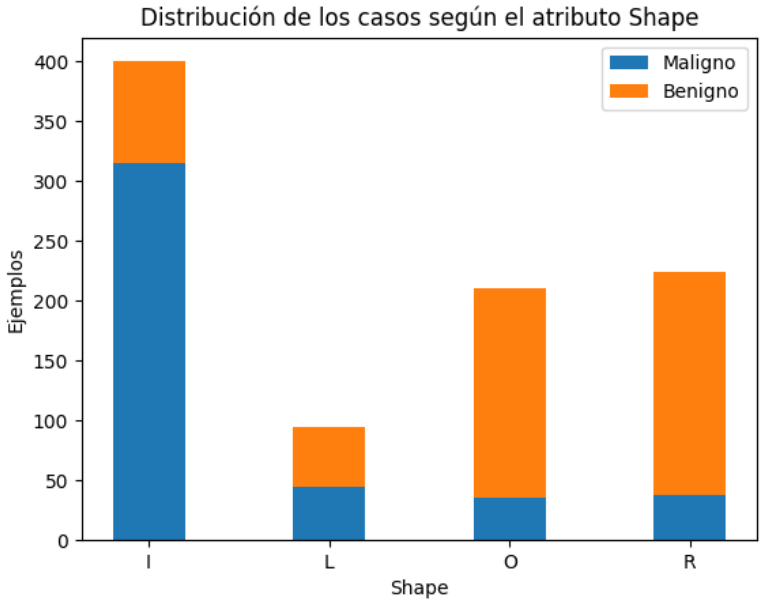


Figura 19: Distribución de ejemplos de cada clase para los distintos valores de Shape

La mayoría de instancias que presentan el valor I de esta variable, corresponden a tumores malignos. La mayoría de las instancias que presentan los valores O o R de esta variable, corresponden a benignos. Mientras que hay aproximadamente las mismas instancias benignas que malignas entre las que tienen valor L. Por tanto, esta variable también está relacionada con la severidad.

Parte II

Segmentación

1. Introducción

En esta práctica emplearemos técnicas de aprendizaje no supervisado, como es el clustering, para realizar un análisis relacional mediante segmentación.

Nuestro dataset consta de todos los accidentes ocurridos en España durante el año 2013. Los datos están publicados por la DGT. Disponemos de datos de 89519 accidentes, y 32 variables que miden circunstancias en las que ocurre cada accidente (día, hora, visibilidad, superficie de la calzada, factores atmosféricos, zona ...), el tipo (alcance, choque frontal, choque lateral, atropello a peatón o animal, colisión con obstáculo ...) y la gravedad (número de heridos leves y graves y número de muertos).

Los algoritmos de clustering que utilizaremos son:

- **K-means:** Un algoritmo iterativo basado en particionamiento. Es relativamente eficiente ($O(tkn)$ donde n es el número de datos, k el de clústers y t el de iteraciones) y suele alcanzar óptimos locales bastante buenos. Por otra parte, sólo trabaja bien cuando los datos son numéricos, necesita prefijar el número de clusters, sólo encuentra clusters esféricos y no lidia bien con datos ruidosos y outliers.
- **DBSCAN:** También basado en particionamiento. No necesita a priori el número de clusters, en su lugar requiere el radio máximo de cada clúster y el tamaño mínimo de los mismos. También es capaz de encontrar clusters con distintas formas y es robusto a datos ruidosos y outliers.
- **Ward:** Es un método aglomerativo, es decir, parte de un clúster para cada elemento y fusiona los clusters que generen en el agrupamiento con mínima varianza. Necesita el número de clúster o una distancia límite a partir de la cual no fusiona dos clusters.

Para estimar los hiperparámetros de los algoritmos atenderemos a dos métricas para interpretar la bondad de un agrupamiento:

- **Coeficiente Silhouette:** Compara la similitud de de los objetos de un mismo cluster (cohesión) con los de otros clusters (separación). Para cada elemento i, se calcula $a(i)$ como la distancia media entre i y el resto de elementos del cluster, mientras que que $b(i)$ representa el mínimo de las distancias entre i y cada cluster distinto del suyo (la distancia de i a un cluster se calcula como la media de las distancias de i a cada elemento del cluster). Tomando
$$s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)}$$
se tiene $s(i) \in [-1, 1]$. Si $s(i)$ es negativo $\big(b(i) < a(i)\big)$, claramente el elemento i pertenece al cluster equivocado (está más cerca de un cluster distinto al suyo); si $s(i)$ está cerca de 1, significa que i está mucho más cerca de los elementos de su cluster que del resto de clusters. El coeficiente silhoutte se calcula como la media de los $s(i)$ para todos los ejemplos i. Usualmente se encuentra entre 0 y 1.
- **Razón de Calinski-Harabasz:** Razón entre la dispersión intra-clusters (dentro de cada cluster) cantidad que y la dispersión inter-clusters (entre clusters). Su cálculo es más complejo que el anterior y su valor no está entre 0 y 1, en nuestro caso toma valores del orden de miles.

Otro factor muy importante a tener en cuanta será el número de clusters generados, ya un elevado número de clusters dificulta las tareas de interpretación y análisis de los resultados. El hecho de generar un número excesivo de clusters con el fin de maximizar las métricas al precio de dificultar el análisis y la interpretabilidad sería el equivalente a lo que en aprendizaje supervisado conocíamos como sobreajuste, podemos llamarle también de ese modo. Representaremos los scores frente al número de cluster en gráficas y elegiremos los hiperparámetros por un criterio similar al método del codo, a partir de la gráfica intentaremos determinar cuando la ganancia de score deja de compensar el aumento en el número de clusters.

Debido al elevado número de datos, los algoritmos de clustering (que suelen presentar un elevado orden de complejidad) podrían tardar un tiempo excesivo en ejecutarse, por lo que realizaremos dos casos de estudio en cada uno de los cuales seleccionaremos un subconjunto de las instancias. Para cada caso de estudio, restringiremos el número de datos fijando los valores de una o más variables de carácter circunstancial o de tipo, y realizaremos el agrupamiento basándonos en las variables que miden la gravedad de los accidentes, pues son numéricas y nos evitan problemas como la dificultad de K-means para trabajar con variables nominales.

Existe una amplia variedad de casos de estudio con sentido e interés que podríamos analizar: accidentes bajo condiciones climatológicas adversas frente a buen tiempo, en zonas urbanas frente a interurbanas, atropellos, colisiones,

etc. Nosotros analizaremos los accidentes en condiciones óptimas para la conducción y los accidentes que ocurren a altas horas de la madrugada.

Es importante normalizar los datos (reescalarlos por columnas para que estén en el intervalo $[0, 1]$ antes de realizar la segmentación para que los algoritmos funcionen correctamente. De lo contrario, algunas dimensiones cobrarían más importancia que otras. Para ello, utilizamos `MinMaxScaler` de *sklearn*.

Para ciertas tareas como el cálulo de las métricas y la visualización de los centroides y las distribuciones de los clusters, utilizamos las funciones que se nos proporcionan en el fichero `pract2_utils.py`, que internamente utiliza *skleran* para las métricas y *seaborn* para las visualizaciones. Hemos relizado pequeñas modificaciones en este fichero: la posibilidad de introducir los centroides ya desnormalizados para su visualización y cambios en las paletas de colores para hacer las visualizaciones más fácilmente interpretables.

2. Caso de estudio 1: Análisis de los accidentes en condiciones óptimas para la conducción

Para este primer caso de estudio, seleccionamos los accidentes que ocurren en condiciones óptimas para la conducción. Estos accidentes pueden ocurrir bien por factor vehículo o bien por factor humano, ya que eliminaremos el factor vía seleccionando ejemplos de accidentes que ocurren en calzadas secas y limpias. No disponemos de datos relativos al factor vehículo, pero dentro del factor humano, eliminaremos también las condiciones que favorecen los errores humanos como la falta de luminosidad, visibilidad o la circulación densa. Teniendo en cuenta que el factor vehículo es causa de un porcentaje relativamente bajo de accidentes (menos del 10 %), principalmente tenemos accidentes debidos al factor humano que han ocurrido en condiciones óptimas para la conducción: buen tiempo, buena visibilidad, calzada limpia y seca, tráfico fluido, etc. Es decir, accidentes debidos a factor humano que se podrían haber evitado. Aquí radica el interés de este caso de estudio.

En concreto, estos son los valores que fijamos para las variables:

```
LUMINOSIDAD=='NOCHE: ILUMINACIÓN SUFICIENTE' | LUMINOSIDAD=='PLENO DÍA'
DENSIDAD_CIRCULACION=='FLUIDA'
DIASEMANA==6
OTRA_CIRCUNSTANCIA=='NINGUNA'
FACTORES_ATMOSFERICOS=='BUEN TIEMPO'
MEDIDAS_ESPECIALES=='NINGUNA MEDIDA'
SUPERFICIE_CALZADA=='SECA Y LIMPIA'
VISIBILIDAD_RESTRINGIDA=='SIN RESTRICCIÓN'
```

Con `OTRA_CIRCUNSTANCIA=='NINGUNA'` descartamos accidentes ocurridos con obras, inundaciones, baches, badenes, cambios de rasante, estrechamientos, pasos a nivel y demás circunstancias que podrían dificultar la conducción. Con `MEDIDAS_ESPECIALES=='NINGUNA MEDIDA'` descartamos accidentes ocurridos en carriles reversibles o con habilitación del arcén, que podríamos considerar fuera de la conducción "normal".

Además, para quedarnos con menos datos nos restringimos a los accidentes ocurridos en sábado con `DIASEMANA==6`. Los días del lunes al jueves son más propensos a presentar accidentes con una sola víctima (como apreciamos en la Figura 20), esto ocurre porque va mucha más gente a trabajar, mientras que en fin de semana es más usual que viajen varias personas en el coche. Así que elegimos este día para obtener un mayor número de accidentes con múltiples víctimas. El hecho de que viajen más personas en el coche a parte del conductor, por una parte puede aumentar la responsabilidad y por otra puede aumentar las distracciones o crear una falsa sensación de confianza. Esto hace los accidentes ocurridos en fin de semana más interesantes, en mi opinión. Motivo por el cuál elegimos un día del fin de semana.

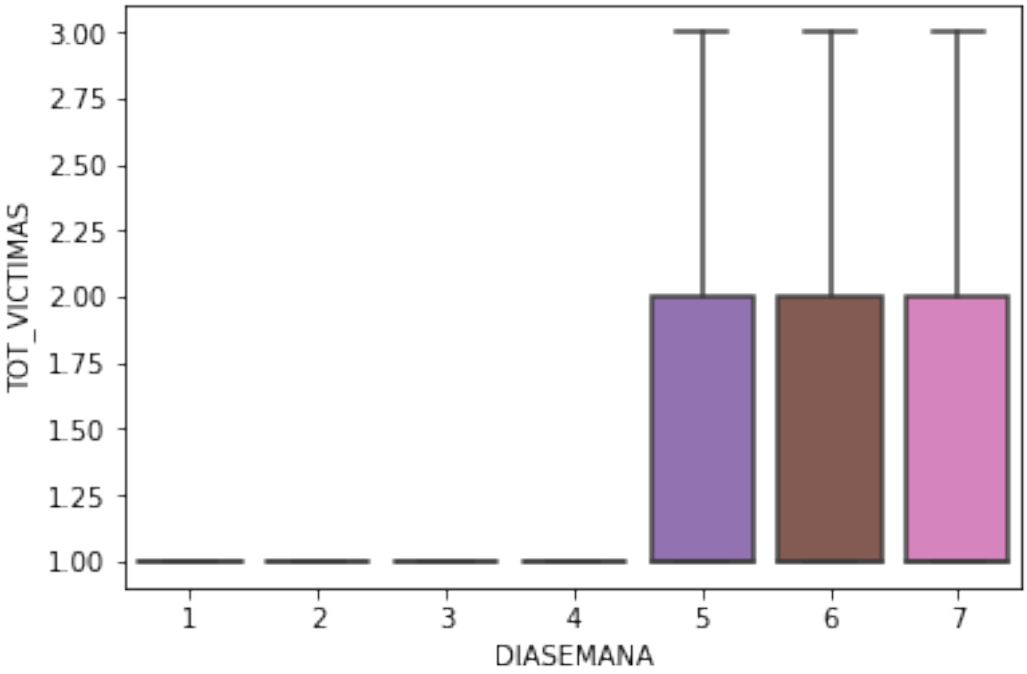


Figura 20: De lunes a jueves la mayoría de accidentes ocasionan una sola víctima (considera los accidentes con más víctimas como outliers). En fin de semana hay más accidentes con múltiples víctimas.

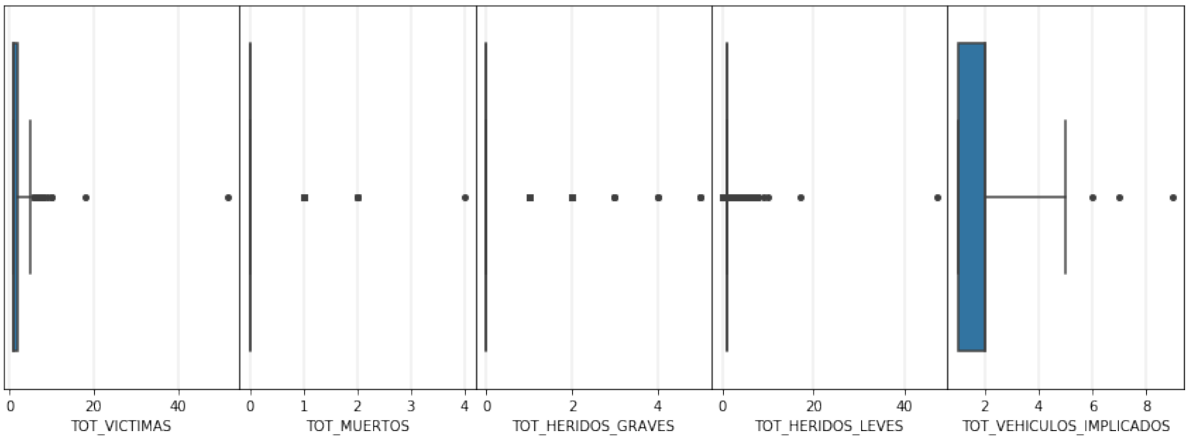
Fijando estos valores, nos quedamos con 4143 instancias, por lo que nuestros algoritmos se ejecutarán bastante rápido.

Anticipando problemas posteriores, debemos eliminar los outliers para poder visualizar la segmentación correctamente. Utilizando `Counter` observamos que hay un accidente con 52 víctimas, esto distorsiona las escalas de los ejes en las gráficas de víctimas y heridos, lo que dificulta su interpretación.

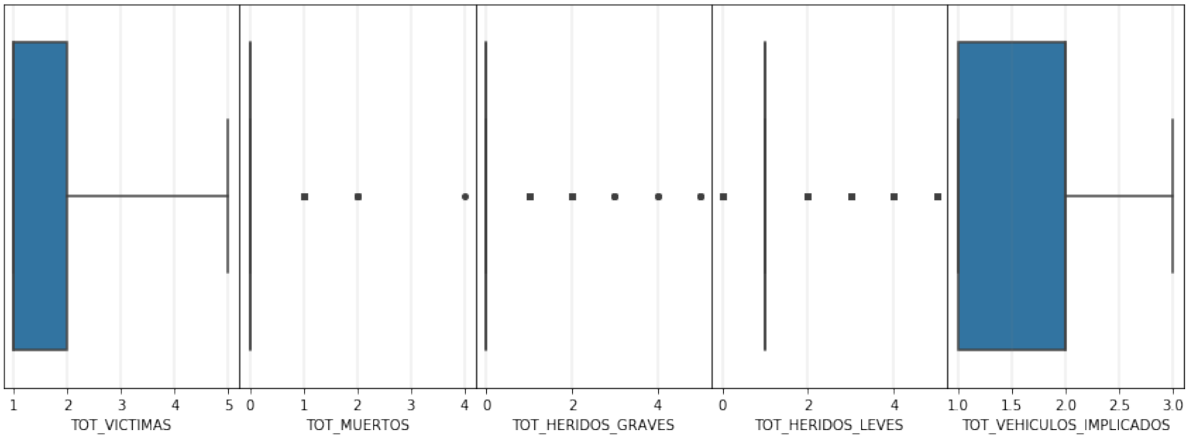
Por tanto, eliminamos los outliers con el siguiente código, que colapsa los valores por encima del máximo de la whisker box, que admite hasta la media más 3 por la desviación típica ($\mu + 3 \cdot \sigma$), en dicho máximo.

```
# Eliminar outliers, ya que distorsionan las gráficas
for a in atributos:
    if a in {'TOT_MUERTOS','TOT_HERIDOS_GRAVES'}:
        continue
    d=data[a][abs(zscore(data[a]))<3]
    data[a][zscore(data[a])<-3]=d.min()
    data[a][zscore(data[a])>3]=d.max()
```

La gran mayoría de accidentes no presenta muertos ni heridos graves, lo que provoca que los muertos y heridos graves se consideren outliers, por eso los dejamos tal y como están. En la siguiente gráfica podemos comparar las distribuciones de las variables, se elminan la mayoría de outliers y se regula la escala de los ejes, salvo en los dos atributos que no hemos modificado.



(a) Antes de eliminar outliers.



(b) Tras eliminar outliers.

Figura 21: Distribución de los valores de cada variable.

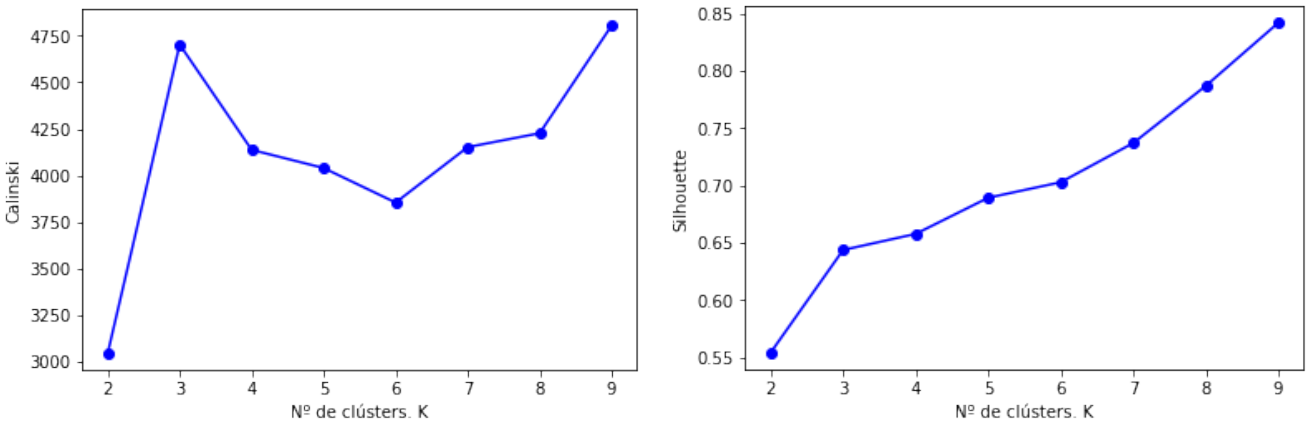
2.1. Algoritmos de clustering y resultados

El primer algoritmo que probaremos para abordar este problema es **K-means**, utilizaremos la implementación del módulo de clusters de *sklearn*. En primer lugar debemos escoger un valor de K adecuado, para ello nos basamos en las métricas Silhouette y Calinski(-Harabasz). Probando diferentes valores de K, obtenemos la siguiente tabla:

K	Calinski	Silhouette
2	3042.33	0.5541
3	4703.28	0.6437
4	4136.87	0.6577
5	4038.97	0.6892
6	3853.22	0.7027
7	4151.42	0.7369
8	4226.28	0.7867
9	4805.5	0.8417

Tabla 1: Scores de K-means para distinto número de clusters.

Nos ayudamos de representaciones gráficas para elegir un valor de K adecuado. Buscamos un valor bajo para K y valores altos para las métricas.



(a) Razón de Calinski-Harabasz

(b) Coeficiente Silhouette

Figura 22: Representaciones de los scores de K-means para distinto número de clusters.

La gráfica de Calinski claramente sugiere tomar $K = 3$, ya que presenta un máximo local bastante alto, sólo superado por $K = 9$ y no por mucho. En cambio, la gráfica de Silhouette no experimenta una ganancia tan abrupta para este valor, pero la ganancia es aun más suave para valores posteriores, por lo que de no tomar $K = 3$, sugeriría tomar $K = 8$ o $K = 9$. Esto complica bastante la posterior interpretación de la segmentación, por lo que nos decantamos por $K = 3$.

El siguiente código realiza el agrupamiento y obtiene las etiquetas (el cluster al que pertenece cada ejemplo) y los centroides de cada cluster.

```
K=3
results = KMeans(n_clusters=K, random_state=0).fit(data_norm)
labels=results.labels_
centroids=results.cluster_centers_
```

Con `Counter`, consultamos el tamaño (número de elementos) de cada cluster. Observamos que el cluster 0 es bastante más pequeño que los otro dos.

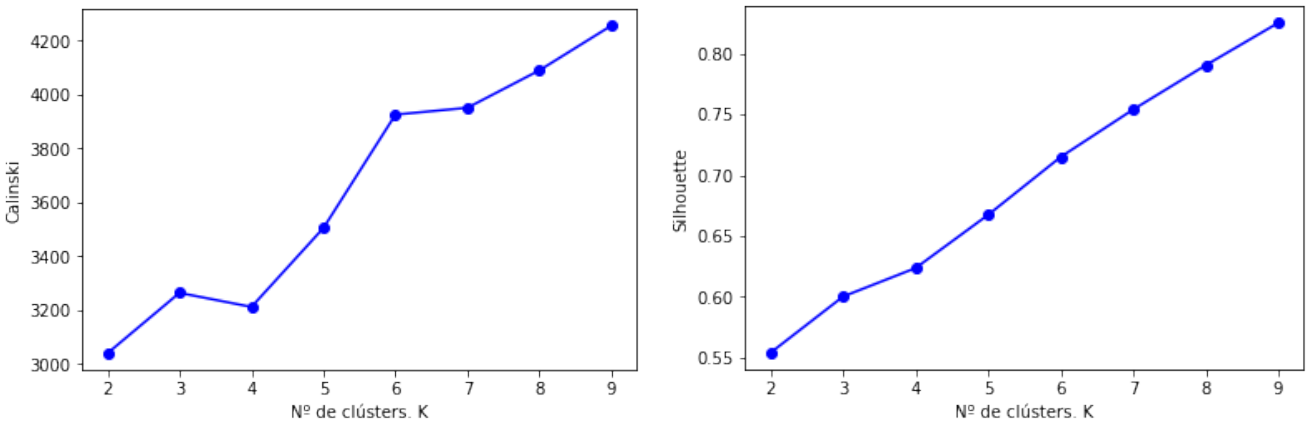
```
Counter(labels)
> Counter({0: 419, 1: 1673, 2: 2051})
```

Dejaremos la visualización de los centroides y de las distribuciones de los clusters para la interpretación, en la Sección 2.2. Relizaremos ahora el agrupamiento con el algoritmo **Ward**, un método acumulativo. La implementación que usamos es la de clustering aglomerativo de *sklearn*, indicando Ward (mínima varianza) como criterio para fusionar clusters (parámetro `linkage`).

Para seleccionar un número adecuado de clusters, realizamos un experimento análogo al de la Tabla 1. Obteniendo:

K	Calinski	Silhouette
2	3042.33	0.5541
3	4703.28	0.6437
4	4136.87	0.6577
5	4038.97	0.6892
6	3853.22	0.7027
7	4151.42	0.7369
8	4226.28	0.7867
9	4805.5	0.8417

Tabla 2: Scores de Ward para distinto número de clusters.



(a) Razón de Calinski-Harabasz

(b) Coeficiente Silhouette

Figura 23: Representaciones de los scores de Ward para distinto número de clusters.

La métrica de Calinski sugiere los valores $K = 3$ y $K = 6$, mientras que Silhouette crece prácticamente de forma lineal. Tomaremos $K = 6$, ya que la ganancia en la métrica de Calinski es significativa y así tendremos la oportunidad de analizar un agrupamiento con un número de clusters algo mayor.

El siguiente código realiza el agrupamiento y obtiene las etiquetas (el cluster al que pertenece cada ejemplo).

```
K=6
results = AgglomerativeClustering(distance_threshold=None, n_clusters=K).fit(data_norm)
labels=results.labels_
```

Este algoritmo no devuelve los centroides, para calcularlos agrupamos los datos por etiqueta y hacemos la media de cada atributo.

```
dataC=data.copy()
dataC['cluster']=labels
centroids = dataC.groupby('cluster').mean()
```

Vemos que hay diferentes tamaños de cluster.

```
Counter(labels)
> Counter({0: 1493, 1: 549, 2: 1535, 3: 140, 4: 264, 5: 162})
```

2.2. Interpretación de la segmentación

Comenzaremos por el agrupamiento generado por el algoritmo K-means. Visualizamos los centroides con la función `visualize_centroids` del fichero `pract2_utils.py`. Visualizamos también cómo se distribuyen los valores de las variables en cada cluster.

Visualizar los centroides nos da una idea del tipo de accidentes en cada cluster, pero visualizar la distribución de las variables en un diagrama de cajas y bigotes nos proporciona más información. Porque podemos saber cómo de cerca están los elementos del cluster de ese centroide. Si vemos una sola línea, significa que la mayoría de elementos del cluster tienen ese valor, los suficientes para considerar el resto como outliers.

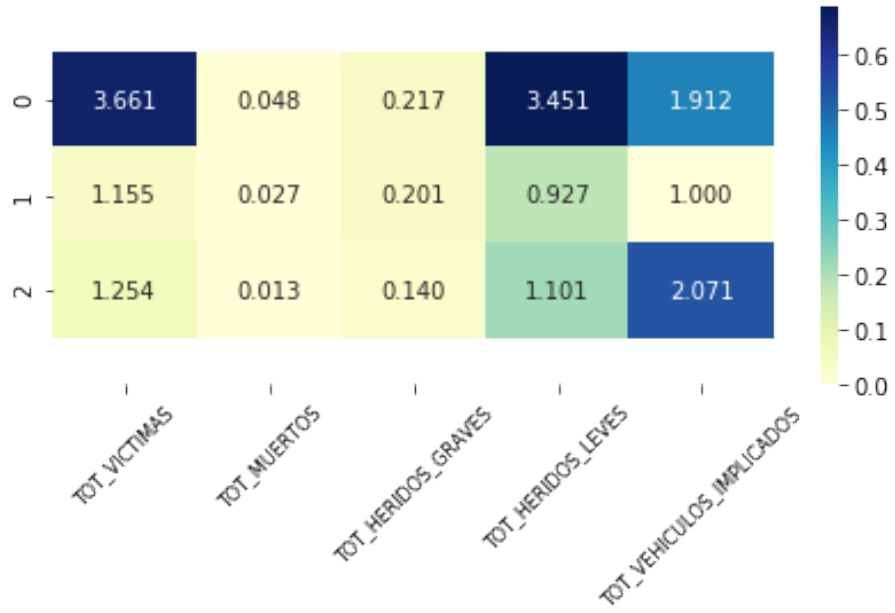


Figura 24: Centroides de los clusters generados por K-means.

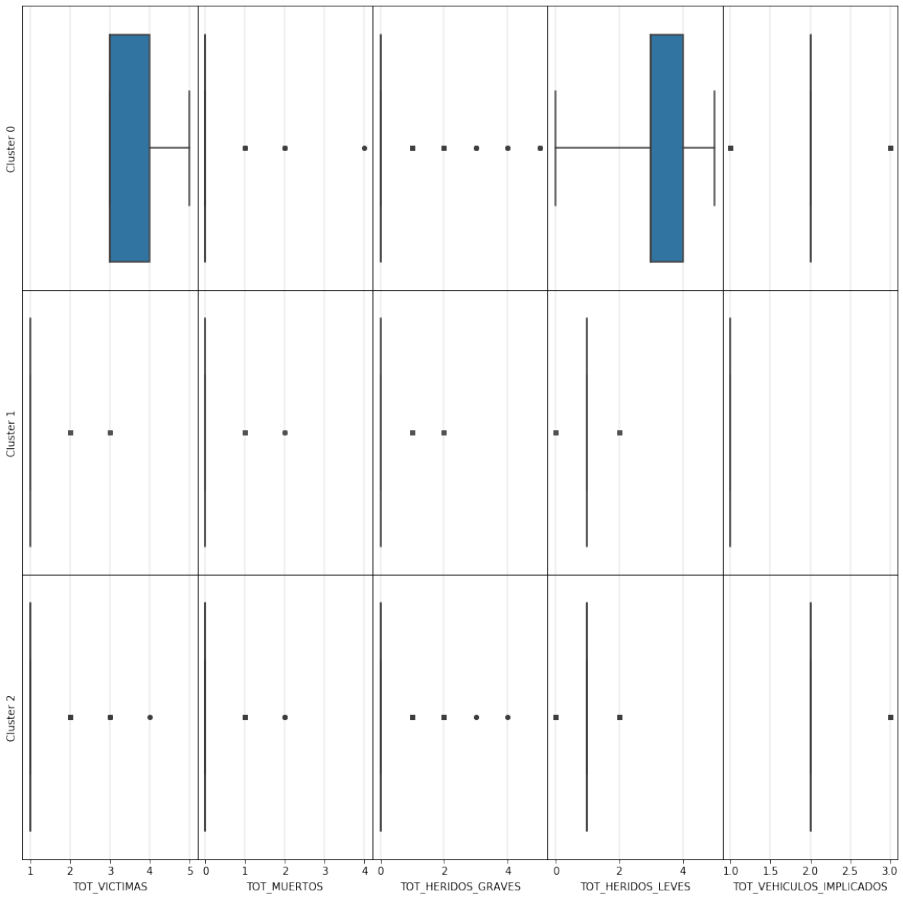


Figura 25: Distribución de las variables en los clusters generados por K-means.

Tenemos el cluster 0, que agrupa los accidentes con 3 o más víctimas, la mayoría de las cuales son heridos leves.

Generalmente son accidentes con dos vehículos implicados, aunque en la Figura 24 apreciamos que debe haber algunos accidentes con un sólo vehículo implicado (la media es ligeramente menor a 2), pero la Figura 25 vemos que estos casos constituyen un porcentaje tan bajo que los considera outliers. Los elementos de este cluster probablemente representen en su mayoría choques frontales/laterales de vehículos, que suelen ser los que más víctimas ocasionan.

En el cluster 1, tenemos accidentes con generalmente una vícitma (que acostrumbra a ser leve) y un vehículo implicado. Podrían tratarse, por ejemplo, de choques de un solo vehículo con un obstáculo, generalmente con un sólo conductor y leves. En la Figura 25 apreciamos que en algunas ocasiones (outliers) hay más heridos o de mayor gravedad, pero en el caso de los vehículos implicados únicamente hay accidentes con un vehículo implicado.

Por último, el cluster 2 agrupa accidentes con dos vehículos implicados pero con bajo número de víctimas y generalmente leves. Pueden tratarse en su mayoría de colisiones por alcance, que acostumbran a ser más leves que el resto de colisiones. Aunque la Figura 24 nos dice que deben existir en este cluster accidentes don más de dos vehículos implicados (la media es ligeramente mayor que 2), en la Figura 25 observamos que son una minoría (los considera outliers).

Debemos resaltar que la proporción de muertos y heridos leves es demasiado baja como para que se consideren a la hora de agrupar por los datos, en la Figura 25 observamos que los valores distintos de 0 para estas dos variables los considera outliers. Nos surge aquí la siguiente duda que tendremos la oportunidad de responder más adelante: ¿Al aumentar el número de clusters tendrá en cuenta los valores de estas variables para agrupar? En caso afirmativo, ¿a partir de qué número de clusters? ¿merecerá la pena aumentar el número de cluster o esto provocará también que se "fuerce la separación" de clústers que tiene perfecto sentido fusionar? Lo que sería sobreajuste.

Podemos también visualizar los cluster en un *pairplot* para observar la relación entre las variables dos a dos, así como la distribución de los valores de las variables en cada cluster.

Figura 26: Distribución de las variables en los clusters generados por K-means, por pares.

En las diagonales, observamos las distribuciones para cada cluster de los valores del atributo correspondiente. Por ejemplo en los heridos leves y en el núemero de víctimas observamos que el cluster 0 llega a tomar valores más altos (la distribución está más a la derecha), mientras que en los otros dos clústers hay poca diferencia, el cluster 2 (verde) toma el valor 2 con ligeramente más frecuencia. Si observamos los muertos, concluimos que esta variable no nos ayuda a diferenciar los clusters, porque todos tienen su distribución prácticamente degenerada en 0. Si observamos en la diagonal los vehículos implicados, vemos que no aparece el cluster 1 por tomar todos sus elementos el valor 1 para esta variable, lo que hace que la "línea" tenga grosor 0. Esto nos lo advierte un warning (Dataset has 0 variance; skipping density estimate) al generar el gráfico. El principal problema de este tipo de gráfico es que se representan cantidades absolutas, por lo que al existir algunos cluster más grandes que otros, se distorsionan las gráficas, por ejemplo el cluster 0 (azul) es menos numeroso y cuesta distinguir su distribución. Otro problema de este tipo de gráficos es que acostumbrarn a solaparse las gráficas de varias distribuciones, por lo que son muy difíciles de interpretar si el número de clusters es elevado.

Fuera de la diagonal, representamos en la posición (i, j) la relación entre los valores de los atributos i y j (de ahí el nombre de *pairplot*), por lo que el gráfico es simétrico salvo cambiar los ejes. Los problemas de estas representaciones son dos. El primero es que los puntos se tapan unos a otros: por ejemplo en la esquina inferior izquierda vemos un punto naranja (cluster 1) representando un accidente con 1 víctima y un vehículo implicado, pero esto no quiere decir que en los cluster 0 y 2 no puedan existir dichos casos, sólo que son más frecuentes en el cluster 1 (en proporción dentro del propio cluster, si no los cluster más pequeños no aparecerían nunca), pero no sabemos si mucho más frecuentes o sólo un poco más frecuentes que en el resto de clusters, el gráfico no nos dice si el punto es "claramente naranja" o ha quedado a unos pocos ejemplos de ser dibujado en otro color. El segundo problema es que no tiene en cuenta las densidades en cada punto: por ejemplo en la comparación entre los heridos leves y las víctimas aparece un punto azul indicando que al menos hay un accidente con 5 víctimas y 0 heridos leves, y un punto verde indicando que hay accidentes con una víctima y un herido leve. Este último caso es mucho más frecuente, pero esto el gráfico no lo refleja, sólo refleja que al menos hay un accidente con esos valores. Por tanto, estos gráficos no reflejan correctamente las relaciones entre los atributos.

Tras explicar cómo se interpretan este tipo de gráficos y exponer sus inconvenientes, basaremos los futuros análisis en gráficos como las figuras 24 y 25, más fácilmente interpretables.

Ahora pasemos a la segmentación obtenida con el algoritmo aglomerativo **Ward**, hemos seleccionado 6 clusters.

Figura 27: Centroides de los clusters generados por Ward.

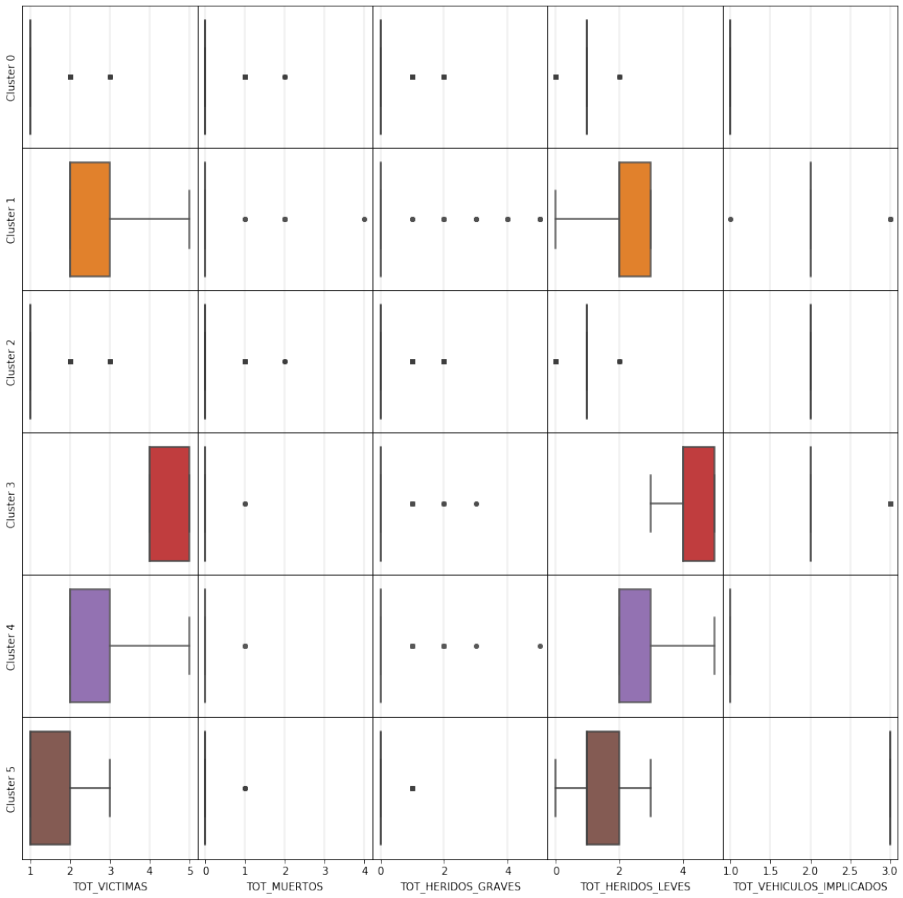


Figura 28: Distribución de las variables en los clusters generados por Ward.

Comprobamos que tampoco utiliza los muertos ni los heridos graves para agrupar los clusters. Sí que hay algunas diferencias entre los valores de estas variables en cada cluster, pero la inmensa mayoría presentan el valor 0 y los considera como outliers (dentro del cluster).

Cada par de clusters se diferencian significativamente en el valor de al menos una variable. Por ejemplo, los clusters 0 y 2 son muy similares en cuanto a la distribución de valores de los atributos en todas las variables salvo el total de vehículos implicados. Ambos podrían costar de choques leves, pero los accidentes del cluster 2 (2 vehículos implicados) serían choques entre vehículos y los del cluster 0 (1 vehículo implicado) serían choques con obstáculos.

En la Figura 25 vimos que, exceptuando las víctimas y heridos leves del cluster 0, cada atributo de cada cluster presenta una distribución muy concentrada (sólo vemos una línea y algunos puntos sueltos que considera outliers). En esta ocasión, los atributos total de víctimas y total de heridos leves en los clusters 1, 3, 4 y 5 presentan distribuciones con rangos de valores más amplios, ya que estos clusters son más pequeños y concentran accidentes con diferentes rangos de valores de estas variables, no centradas en el valor 1.

3. Caso de estudio 2: Análisis de los accidentes a altas horas de la madrugada

3.1. Algoritmos de clustering y resultados

DBSCAN:

3.2. Interpretación de la segmentación