Práctica 2

Inteligencia de Negocio

Amin Kasrou Aouam

UNIVERSIDAD
DE GRANADA

2020-12-13

Índice

Práctica 2

Apartado 1

Introducción

En este apartado, visualizaremos los resultados de la práctica anterior y los interpretaremos.

Procesamiento de datos

Mantenemos el mismo preprocesamiento de datos que en la práctica anterior, lo cual supone que para haremos una gráfica para cada tipo de preprocesamiento (eliminación de valores nulos e imputación con la media).

Ejecución

Para obtener las gráficas ejecutamos el siguiente comando, desde la raíz del proyecto:

```
python src/P1/processing.py drop
```

En el caso de que queramos los resultados al aplicar la imputación de la media, ejecutamos el siguiente comando:

```
python src/P1/processing.py fill
```

El conjunto de las gráficas se encontrará en el directorio **docs/assets**.

Gráfica de curva ROC

Una curva ROC nos permite medir el rendimiento de un clasificador según el umbral de discriminación en un problema de clasificación, i.e. nos permite saber como de bueno es nuestro modelo distinguiendo las diferentes clases.

Procedemos a mostrar cada uno de los modelos con un color distinto, además del *AUC*, dado que este parámetro nos permite asignar un valor numérico al rendimiento de cada modelo.

1. Eliminación de valores

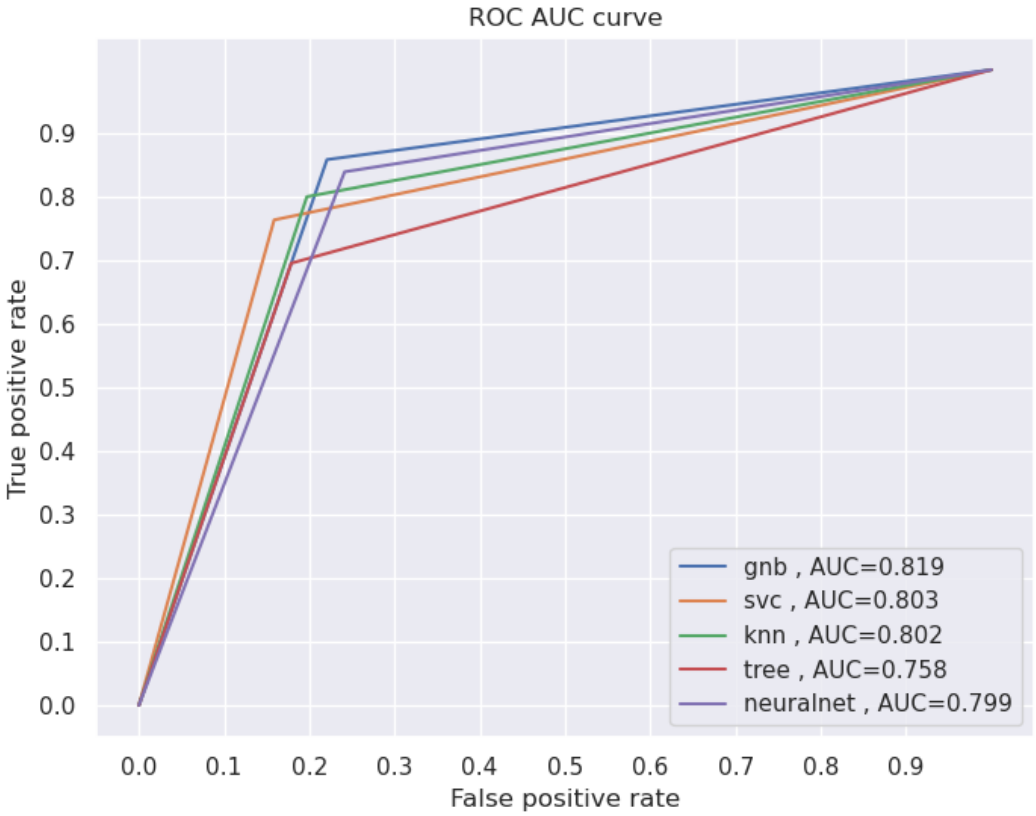


Figura 1: Curva ROC AUC con eliminación de valores

Observamos que el algoritmo *Naive Bayes* es el que obtiene una mejor puntuación, aunque no hay una gran diferencia de valores en el *AUC* entre los demás modelos.

2. Imputación de valores

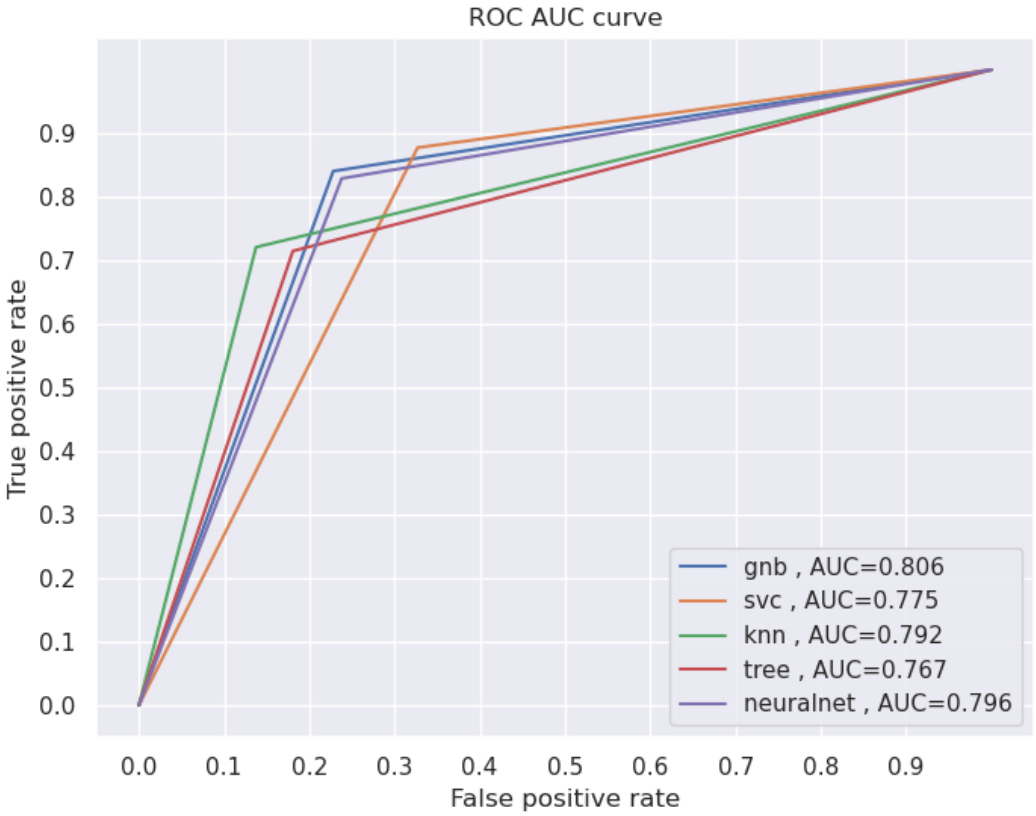


Figura 2: Curva ROC AUC con imputación de valores

Observamos que el algoritmo *Naive Bayes* sigue obteniendo la mejor puntuación, aunque vemos que las curvas han sido alteradas debido al proceso de imputación.

En la práctica anterior llegamos a la conclusión de que no era un buen método de preprocesamiento en nuestro caso particular.

Matriz de confusión

Una matriz de confusión nos permite visualizar el rendimiento de un algoritmo, al incluir el número de:

- Verdaderos positivos
- Falsos positivos
- Verdaderos negativos
- Falsos negativos

Procedemos a generar un *heatmap* por cada algoritmo, para comparar su rendimiento.

1. Eliminación de valores

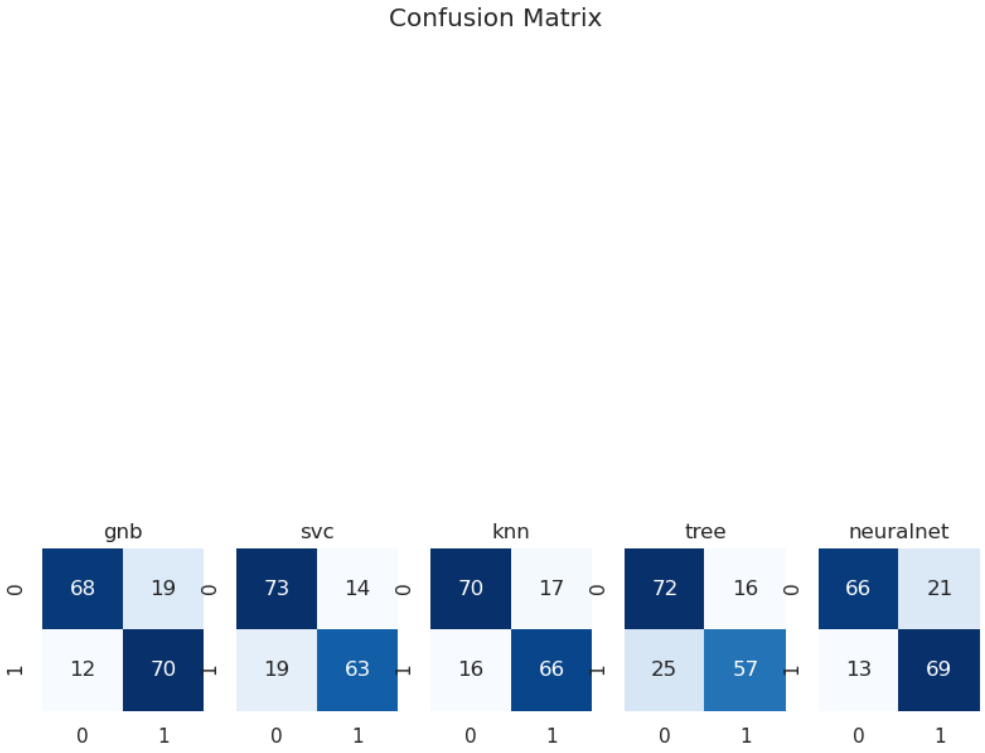


Figura 3: Matriz de confusión con eliminación de valores

Observamos que el algoritmo *Linear SVC* es el que obtiene una mejor puntuación, dado que nos presenta el menos número de falsos negativos y falsos positivos.

2. Imputación de valores

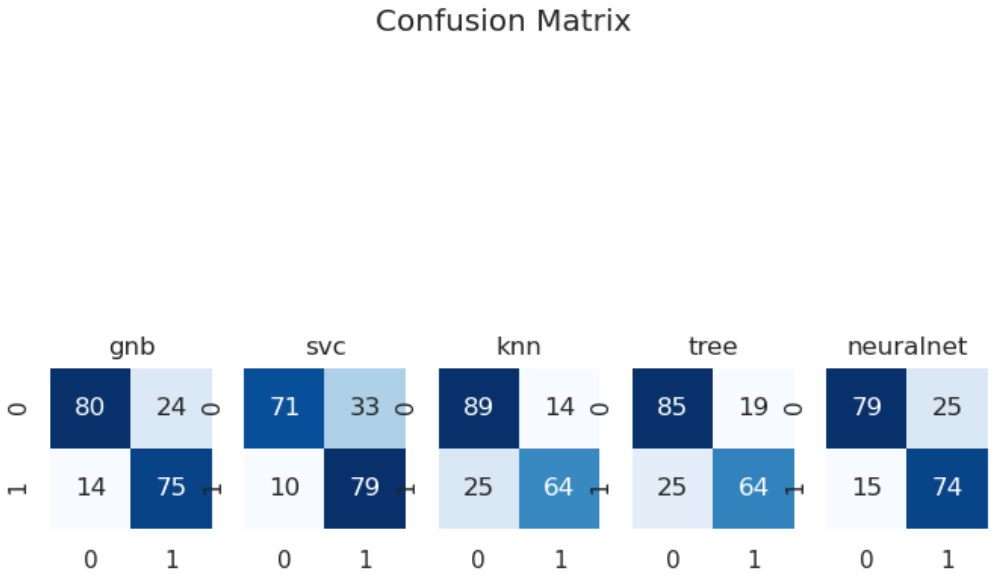


Figura 4: Matriz de confusión con imputación de valores

Observamos que el rendimiento de ciertos algoritmos se degrada en la detección de falsos positivo o falsos negativos. El *Linear SVC* se ve afectado en la detección de falsos positivos, y en este apartado lo supera el *K-NN*.

Correlación entre atributos

Vamos a tratar de observar qué atributos están más relacionados con el resultado del diagnóstico, para determinar cual de ellos es más discriminativo.

Procedemos a generar un histograma para cada uno de los atributos, para ello discretizamos el atributo de la edad.

1. Eliminación de valores

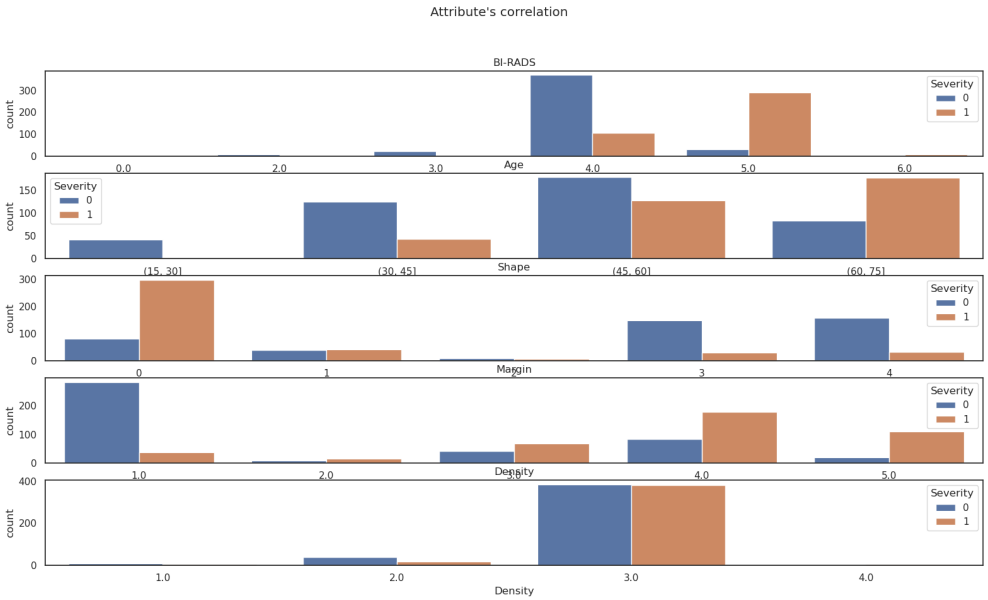


Figura 5: Correlación de atributos con eliminación de valores

Observamos que los atributos que más peso tienen son:

- *Margin*
- *Shape*
- *Age*

2. Imputación de valores

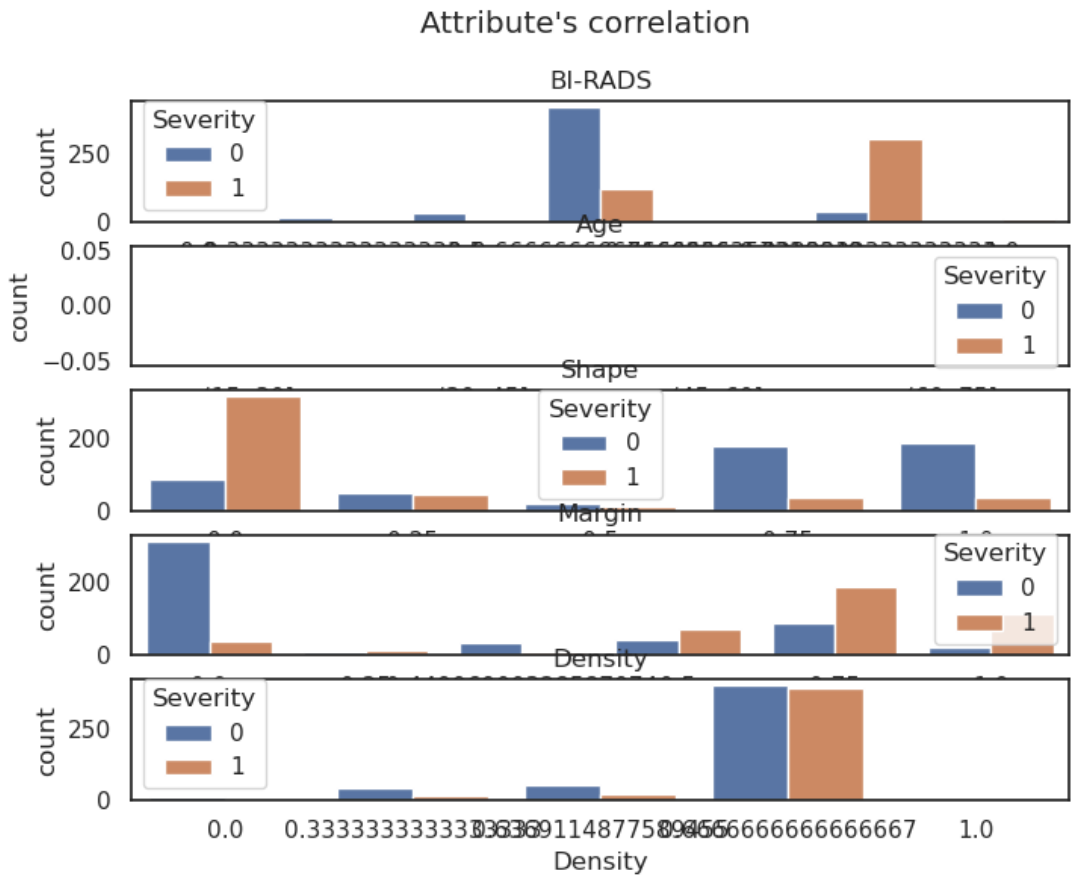


Figura 6: Correlación de atributos con imputación de valores

Observamos que los atributos son menos discriminativos al realizar la imputación, lo cual nos confirma que esta técnica de preprocesamiento es nociva para nuestro caso de estudio.

Apartado 2

Introducción

En este apartado, usaremos distintos algoritmos de *clustering* para resolver un problema de agrupación.

El problema en cuestión trata de agrupar los accidentes que ocurrieron en el año 2013 según la DGT, evaluando variables similares y relaciones de causalidad para determinar la gravedad y el tipo del accidente.

Estudiaremos el rendimiento de los siguientes algoritmos:

- K-means
- Birch
- Spectral clustering
- Mean shift clustering
- DBSCAN

En el caso del *K-means* y de *Birch*, estudiaremos el rendimiento según un diferente número de clusters.

Casos de estudio

1. Caso de estudio 1: Accidentes ocurridos por la noche

   La visibilidad es un factor importante en la conducción, por lo tanto vamos a agrupar los distintos accidentes ocurridos de noche.

   a) 5 clusters

   Ejecutamos el siguiente comando:

   ```
   1 python src/P2/processing.py case1 5
   ```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	5	[1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0.676187	4753.226068	0.061349
birch	5	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.590490	1011.652950	0.098661
spectral	8	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.690485	153.023986	2.732878
meanshift	79	[0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.852581	2710.750238	9.662226
dbscan	27	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.768946	996.405388	0.131728

Figura 7: Caso 1 con 5 clusters

   Como podemos observar, según las diferentes métricas ciertos algoritmos tienen mejor o peor rendimiento:

- *Silhouette Coefficient*: Mean Shift clustering
- *Calinski-Harabasz*: K-means
- *Time*: K-means

b) 10 clusters

Ejecutamos el siguiente comando:

```
python src/P2/processing.py case1 10
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	10	[9, 7, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, ...	0.745974	4820.176476	0.090424
birch	10	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	0.384681	633.758394	0.098308
spectral	8	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.690485	153.023986	2.768155
meanshift	79	[0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.852581	2710.750238	9.699414
dbscan	27	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.768946	996.405388	0.130580

Figura 8: Caso 1 con 10 clusters

Debido a la variación del número de clusters, vemos como el rendimiento de *Birch* se degrada de forma considerable, mientras que el *k-means* obtiene una mejor puntuación

2. Caso de estudio 2: Accidentes en islas con lluvia

Estudiamos los accidentes que han ocurrido en el territorio español no peninsular, en condiciones lluviosas.

a) 5 clusters

Ejecutamos el siguiente comando:

```
python src/P2/processing.py case2 5
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	5	[0, 3, 0, 3, 3, 4, 0, 0, 0, 0, 3, 0, 0, 3, 3, ...	0.661937	146.167681	0.025141
birch	5	[3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.636028	70.283936	0.007813
spectral	8	[4, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.607824	157.130072	0.048306
meanshift	7	[6, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.723650	140.720785	0.159004
dbscan	4	[-1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...	0.654457	65.579559	0.002411

Figura 9: Caso 2 con 5 clusters

Como podemos observar, según las diferentes métricas ciertos algoritmos tienen mejor o peor rendimiento:

- *Silhouette Coefficient*: Mean Shift clustering
- *Calinski-Harabasz*: Spectral clustering
- *Time*: DBSCAN

b) 10 clusters

Ejecutamos el siguiente comando:

```
python src/P2/processing.py case2 10
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	10	[8, 7, 0, 7, 5, 4, 0, 0, 0, 0, 7, 0, 0, 5, 7, ...	0.884567	789.703386	0.042207
birch	6	[0, 1, 1, 1, 1, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0.674235	108.686547	0.005430
spectral	8	[5, 1, 1, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0.607824	157.130072	0.065682
meanshift	7	[6, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.723650	140.720785	0.149201
dbscan	4	[-1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	0.654457	65.579559	0.002284

Figura 10: Caso 2 con 10 clusters

Debido a la variación del número de clusters, vemos como el rendimiento del *k-means* supera a todos los demás algoritmos, excepto en la métrica del tiempo de ejecución.

3. Caso de estudio 3: Accidentes en autopista después de las 19

   Estudiamos los accidentes que han ocurrido en autopistas, después de las 19:00.

   a) 5 clusters

   Ejecutamos el siguiente comando:

```
python src/P2/processing.py case3 5
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	5	[4, 4, 4, 4, 4, 3, 0, 0, 0, 0, 4, 2, 0, 0, 0, ...	0.435276	49.075312	0.024536
birch	5	[0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 3, 0, 0, 0, ...	0.395290	34.490745	0.012562
spectral	8	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 4, 0, 0, 0, ...	0.207974	14.166581	0.043937
meanshift	8	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 4, 6, 0, 0, 0, ...	0.433497	35.904919	0.151134
dbscan	4	[0, 0, 0, 0, 0, -1, 1, -1, 2, 2, -1, -1, 2, -1...	0.338141	7.661519	0.003088

Figura 11: Caso 3 con 5 clusters

Observamos como el *K-means* es el que obtiene mejores resultados, con un tiempo de ejecución marginalmente superior al del *DBSCAN*.

b) 10 clusters

Ejecutamos el siguiente comando:

```
python src/P2/processing.py case3 10
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	10	[4, 4, 4, 4, 4, 5, 0, 0, 1, 1, 8, 2, 1, 0, 1, ...	0.634518	77.020351	0.038275
birch	10	[0, 0, 0, 0, 0, 5, 5, 0, 2, 2, 9, 7, 2, 0, 2, ...	0.528897	53.384881	0.008883
spectral	8	[1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 5, 2, 1, 1, 1, ...	0.288635	19.920989	0.045778
meanshift	8	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 4, 6, 0, 0, 0, ...	0.433497	35.904919	0.136926
dbscan	4	[0, 0, 0, 0, 0, -1, 1, -1, 2, 2, -1, -1, 2, -1...	0.338141	7.661519	0.002136

Figura 12: Caso 3 con 10 clusters

Debido a la variación del número de clusters, vemos como el rendimiento del *K-means* y de *Birch* mejora considerablemente.

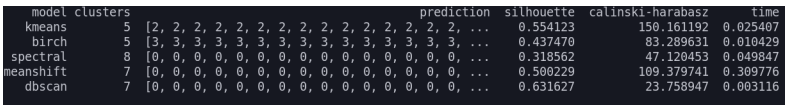
4. Caso de estudio 4: Accidentes en Andalucía sin iluminación

   Estudiamos los accidentes que han ocurrido en la comunidad autónoma de Andalucía, en particular, cuando no había iluminación.

   a) 5 clusters

   Ejecutamos el siguiente comando:

```
python src/P2/processing.py case4 5
```



model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	5	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	0.554123	150.161192	0.025407
birch	5	[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ...	0.437470	83.289631	0.010429
spectral	8	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.318562	47.120453	0.049847
meanshift	7	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.500229	109.379741	0.309776
dbscan	7	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.631627	23.758947	0.003116

Figura 13: Caso 4 con 5 clusters

Como podemos observar, según las diferentes métricas ciertos algoritmos tienen mejor o peor rendimiento:

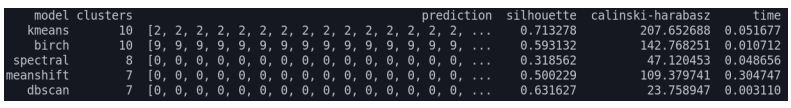
- *Silhouette Coefficient*: DBSCAN
- *Calinski-Harabasz*: K-means
- *Time*: DBSCAN

Observamos como el algoritmo con mejor rendimiento es el *DBSCAN*, excepto en la métrica del *Calinski-Harabasz* donde predomina el *K-means*.

b) 10 clusters

Ejecutamos el siguiente comando:

```
python src/P2/processing.py case4 10
```



model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	10	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	0.713278	207.652688	0.051677
birch	10	[9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, ...	0.593132	142.768251	0.010712
spectral	8	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.318562	47.120453	0.048656
meanshift	7	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.500229	109.379741	0.304747
dbscan	7	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.631627	23.758947	0.003110

Figura 14: Caso 4 con 10 clusters

Debido a la variación del número de clusters, vemos como el rendimiento del *K-means* y de *Birch* mejora considerablemente. En el caso particular del *K-means*, éste supera al *DBSCAN* en todas las métricas, excepto la del tiempo de ejecución.

5. Caso de estudio 5: Accidentes ocurridos un domingo en Madrid

   Estudiamos los accidentes que han ocurrido en domingo, en la comunidad de Madrid.

   a) 5 clusters

   Ejecutamos el siguiente comando:

```
python src/P2/processing.py case5 5
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	5	[0, 1, 1, 4, 2, 4, 1, 4, 1, 1, 1, 1, 0, 2, 0, ...	0.731629	172.471612	0.024995
birch	5	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.421421	39.782565	0.011128
spectral	8	[0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 7, 4, ...	0.475337	50.316079	0.048466
meanshift	17	[1, 0, 0, 2, 10, 2, 0, 2, 0, 0, 0, 0, 1, 3, 4,...	0.964706	1.000000	0.120523
dbscan	8	[0, 1, 1, 2, -1, 2, 1, 2, 1, 1, 1, 1, 0, 3, 4,...	0.845252	39.528474	0.003105

Figura 15: Caso 5 con 5 clusters

   Como podemos observar, según las diferentes métricas ciertos algoritmos tienen mejor o peor rendimiento:

   - *Silhouette Coefficient*: Mean Shift
   - *Calinski-Harabasz*: K-means
   - *Time*: DBSCAN

   Observamos como el algoritmo *Mean Shift* consigue un *Silhouette Coefficient* cerca de 1.

   b) 10 clusters

   Ejecutamos el siguiente comando:

```
python src/P2/processing.py case5 10
```

model	clusters	prediction	silhouette	calinski-harabasz	time
kmeans	10	[2, 1, 1, 0, 6, 0, 1, 0, 1, 1, 1, 1, 2, 6, 8, ...	0.905809	333.474708	0.040625
birch	10	[2, 2, 2, 0, 0, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, ...	0.553973	56.361403	0.012076
spectral	8	[0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 7, 6, ...	0.475337	50.316079	0.048228
meanshift	17	[1, 0, 0, 2, 10, 2, 0, 2, 0, 0, 0, 0, 1, 3, 4,...	0.964706	1.000000	0.111703
dbscan	8	[0, 1, 1, 2, -1, 2, 1, 2, 1, 1, 1, 1, 0, 3, 4,...	0.845252	39.528474	0.003129

Figura 16: Caso 5 con 10 clusters

   Debido a la variación del número de clusters, vemos como el rendimiento del *K-means* mejora considerablemente y *Birch* mejora de forma marginal. En el caso particular del *K-means*, el valor de su *Silhouette Coefficient* se acerca mucho al valor del *Mean Shift*.