Universitat de València

Machine Learning I

Trabajo Final A

Clasificación de eventos sonoros

Grupo A4:

Enrique Lozoya Lidón
José Manuel Sánchez Aquilué

marzo, 2022

Escuela Técnica Superior de Ingeniería (ETSE-UV)

Índice

1. Introducción

A lo largo de este documento se desarrollarán los resultados obtenidos en el trabajo final de la asignatura de machine learning. Trataremos de clasificar seis tipos de eventos sonoros (estornudos, llanto, aplausos, pisadas, tos y respiraciones) mediante todo tipo de técnicas de aprendizaje automático vistas durante el curso.

Las muestras de audio cuentan con multitud de dimensiones, muchas de ellas redundantes. Por tanto, será necesario reducir la dimensionalidad de los datos antes de comenzar. Se pondrán a prueba distintas maneras de reducir dimensionalidad y seleccionaremos aquella que resulte más óptima. A continuación se intentará clasificar por medio de técnicas de aprendizaje no supervisado, más concretamente métodos de agrupamiento. Por un lado, probaremos algoritmos de agrupamiento obviando que conocemos las etiquetas y veremos hasta qué punto los agrupamientos obtenidos se parecen al etiquetado real. Asimismo, con el fin de acelerar los modelos sin perder calidad, realizaremos una reducción del conjunto por medio de distintas técnicas de agrupamiento, que seleccionarán las muestras más representativas de cada clase.

En lo que se refiere a aprendizaje supervisado, se pondrán a prueba los tres principales modelos vistos en clase: árboles de decisión (DT), máquinas de vectores de soporte (SVM) y perceptrones multicapa (MLP). Se ajustarán de manera precisa para dar con las mejores puntuaciones. Asimismo se determinará cuales son las características más influyentes para cada modelo según criterios basados en importancia de Gini y valores de Shapley. De hecho, en base a esos rankings se probarán modelos entrenados con el subconjunto de las mejores características. Finalmente, se construirán distintos ensembles y se extraerán conclusiones en base a todos los resultados obtenidos.

2. Carga de datos

Nuestro conjunto de datos consta de 720 muestras de 6 tipos de eventos sonoros representados de manera alícuota en el dataset, es decir, contamos con 120 sonidos de cada clase. Cada audio tiene una duración de 3 segundos y lo componen 66150 dimensiones. Naturalmente, no resulta factible trabajar con tantas variables y tan pocas muestras debido a la archiconocida maldición de la dimensión. Es por ello que debemos reducir el número de dimensiones de manera que no se produzcan pérdidas de información.

Fueron planteados tres métodos para reducir la dimensionalidad: de dominio temporal, espectrogramas de Mel y Coeficientes Cepstrales en las Frecuencias de Mel (MFCC). Nos decantamos por MFCC, debido a que reducía la dimensionalidad a 479, lo que supone una disminución del 99.3 %, y, esencialmente, a que era el método que otorgaba una mayor precisión. El método de espectrogramas de Mel dejaba cada muestra con tan sólo 178 pero su precisión en el análisis discriminante

lineal era de tan solo 0.55, frente al 0.94 qué conseguíamos con MFCC. En la figura 1 se ilustra la proporción de varianza explicada de cada método en función del número de dimensiones.

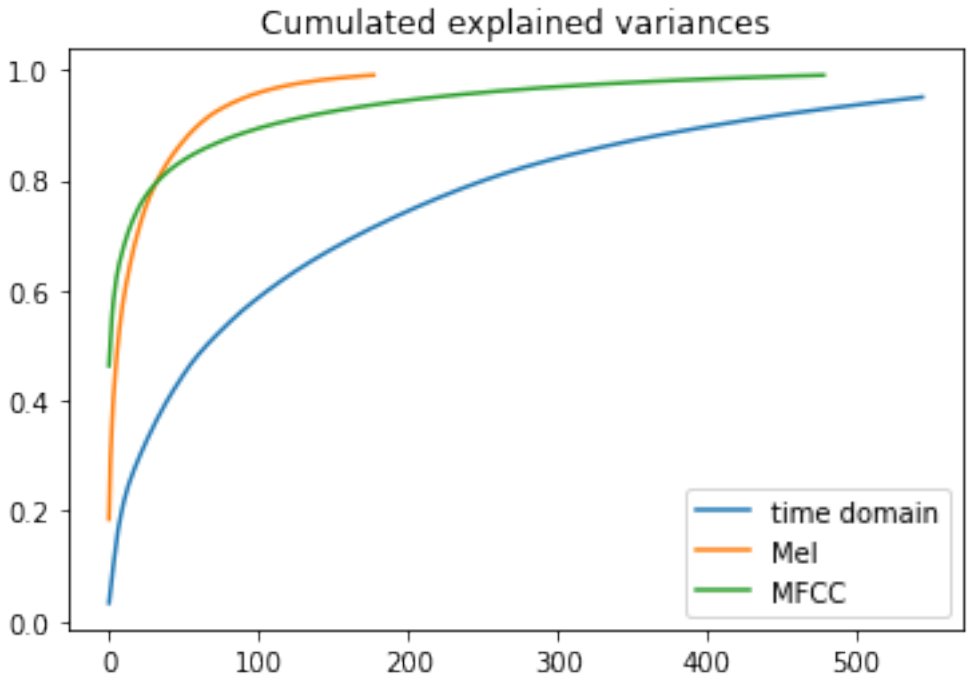

Figura 1: Proporción de varianza explicada por método por nº dimensiones.

Cabe matizar, sin entrar en excesivos detalles teóricos, lo que son los MFCC. Son unos coeficientes que en conjunto forman un MFC [1], un tipo de representación del espectro de potencia a corto plazo de un sonido, que se obtiene a través de la aplicación de la transformada de Fourier discreta, el banco de filtros de la Escala Mel y transformada de coseno, entre otras técnicas de procesado de señales. De esta manera se extraen las características importantes del audio y se evitan aquellas que no aportan información (como el ruido). Finalmente al conjunto de MFCC se le aplica análisis de componentes principales (PCA), con el fin de reducir todavía más la dimensionalidad.

3. Clustering

Tras reducir la dimensionalidad de los datos, comenzamos probando con ellos varios algoritmos de aprendizaje no supervisado. En concreto, aplicaremos distintos métodos de agrupamiento: el jerárquico aglomerativo, el c-medias, su variante c-medias difuso, el algoritmo EM de mezcla de Gaussianas y el mean-shift. Aunque el ámbito de aplicación natural de estos métodos es en conjuntos de datos no etiquetados, nos interesa comprobar hasta qué punto se parecen los grupos obtenidos por estos métodos al etiquetado real. Además, también nos serán de utilidad en el

apartado 4, donde los usaremos para reducir el conjunto de entrenamiento de los modelos supervisados.

Para comparar los agrupamientos con el agrupamiento real usaremos el índice de Fowlkes-Mallows [2], que se define como la media geométrica de precisión y sensibilidad (recall) por parejas. Este índice, acotado entre 0 y 1, compara un agrupamiento con otro que se toma como referencia. En nuestro caso, el clustering de referencia es el dado por las etiquetas. La interpretación es que cuanto mayor sea este índice, más se parecen los dos agrupamientos. Nuestros resultados se resumen en la tabla 2. Vemos que en general no hay un método que dé un clustering significativamente mejor que el resto.

Método		Índice Fowlkes-Mallows
Jerárquico aglomerativo	Single	0.404
	Complete	0.376
	Ward	0.312
	Average	0.396
	Weighted	0.392
	Centroid	0.402
	Median	0.402
K-means		0.318
Fuzzy k-means		0.285
Gaussian mixture		0.328
Mean-shift		0.382

Tabla 2: Resultado de las ejecuciones de los métodos de clustering.

En los métodos en los que se puede indicar el número de clusters a encontrar, todos a excepción de mean-shift, se ha fijado este parámetro a 6. En mean-shift, en cambio, el algoritmo encuentra 5 clusters estimando previamente un ancho de banda. Cambiando a mano el ancho de banda es posible obtener varios agrupamientos que sí tengan 6 grupos, pero en cualquier caso tampoco se obtiene una mejora en el índice Fowlkes-Mallow.

Podríamos haber seguido otra estrategia: encontrar el número óptimo de grupos con cada método usando alguna medida de bondad que no tome como referencia un etiquetado (una *ground truth*) como por ejemplo puede ser el coeficiente silhouette [5], para ver si se obtienen 6 grupos. No obstante, estos coeficientes asumen que los clusters son zonas de puntos densas separados entre sí por zonas de baja densidad, algo que no tiene porqué cumplirse con estos datos. De hecho, obsérvese que el coeficiente silhouette del etiquetado real es $-0{,}06$, incluso negativo, lo cual indica que no se cumple esta premisa. También se puede comprobar que hay un alto grado de solape entre las clases viendo la figura 3, donde se ha representado las dos primeras variables que se obtienen tras la PCA, i.e. aquellas que explican la mayor cantidad de varianza en los datos.

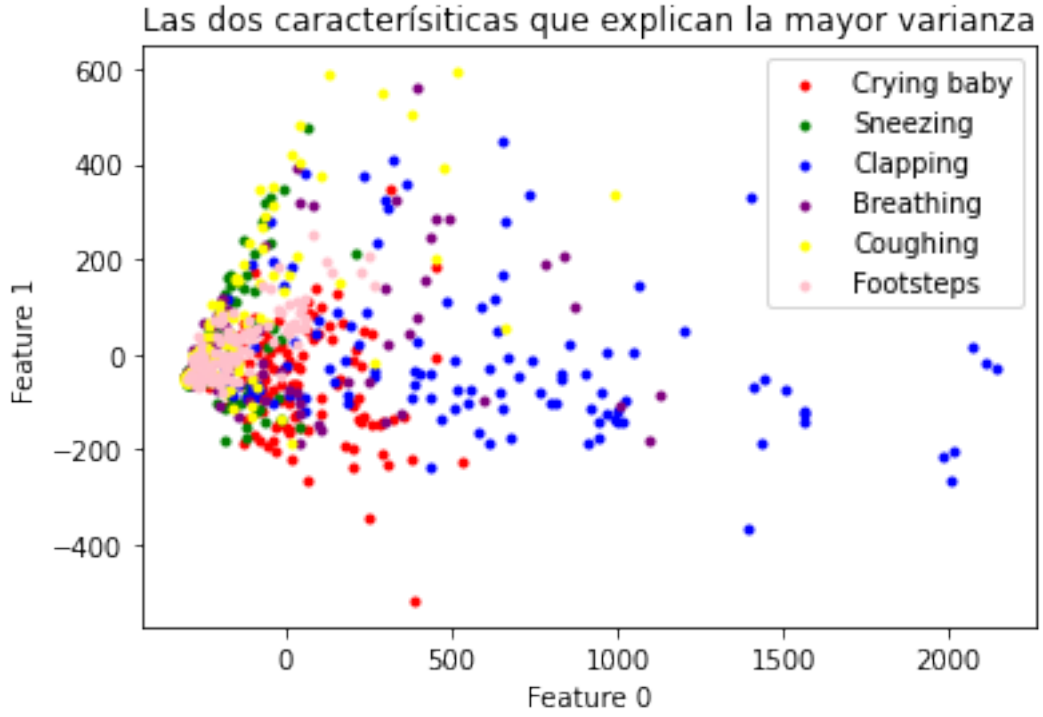

Figura 3: Dos primeras variables obtenidas tras hacer la PCA.

4. Reducción del conjunto

Para entrenar los modelos de aprendizaje supervisado usaremos conjuntos de entrenamiento de tamaño reducido. Esta reducción se hará mediante técnicas de clustering; en concreto, con k-medoids, k-medias y mezcla de Gaussianas. La metodología se detalla a continuación:

- **K-medoids**: Para cada clase, se ajusta un modelo k-medoids fijando tantos clusters como puntos queramos obtener en la reducción. Por ejemplo, dado que tenemos 120 datos por clase, si quisiéramos una reducción al 60 % se buscarían 72 clusters. Los puntos de la reducción serían los 72 medoides. El algoritmo k-medoids está implementado en la librería scikit-learn-extra. Es similar al k-means, con la salvedad de que las medias han de ser puntos del conjunto.
- **K-means**: El procedimiento con k-medias es el mismo que con k-medoids, es decir, buscar tantos grupos como datos queramos obtener en la reducción. Los puntos que forman parte de la reducción serán los más cercanos a los centroides de entre el conjunto original. El tamaño de la reducción, al igual que con los otros dos métodos, se puede elegir como parámetro en la función.
- **Mezcla de Gaussianas**: Se ajusta a cada clase una mezcla de Gaussianas con 3 componentes y matrices de covarianzas general. El número de componentes es variable, pero encontramos que 3 componentes supone un buen compromiso

entre tiempo de cómputo y un mejor ajuste. Posteriormente se extraen muestras aleatorias de la distribución y se busca el punto del conjunto original más cercano a la muestra. Este punto se añade al conjunto reducido y se repite hasta encontrar tantos puntos como haga falta.

Todos los métodos anteriores tienen una cierta componente aleatoria, que se puede fijar mediante un parámetro de tipo "estado aleatorio". Para valorar qué conjunto o conjuntos proporcionarán mejores resultados en los siguientes apartados se han hecho 18 reducciones, que corresponden a los tres métodos, con tres estados aleatorios distintos y dos tamaños de reducción.

Una forma de validar estos conjuntos es entrenar y testear un árbol de decisión para cada uno de estos conjuntos y ver cuál tiene mejor rendimiento en test. A propósito de esto último, los conjuntos de test serán sacados del conjunto reducido, dedicando entorno a un 70 %-30 % del conjunto reducido para entrenamiento y test respectivamente. Es por ello que los dos tamaños de reducción que probamos son un $\sim 72\,\%$ y un $85\,\%$, de forma que los conjuntos finales de entrenamiento supongan un $\sim 50\,\%$ y $\sim 59\,\%$ del conjunto original.

En general, si nos fijamos en una sola métrica, por ejemplo la overall accuracy, los conjuntos que mejor rendimiento tienen en test son los sacados con k-medias y con mezcla de Gaussianas. Además, mirando la matriz de confusión vemos que el método de mezcla de Gaussianas es el que mayor porcentaje de acierto tiene para todas las clases (los valores mayores se acumulan en la diagonal).

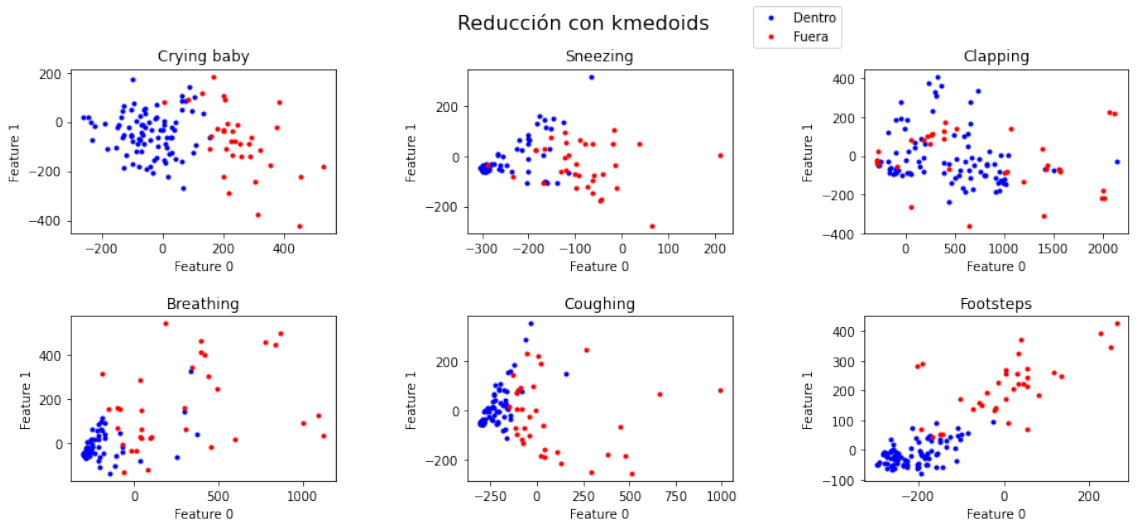

Figura 4: Resultados de la reducción mediante k-medoids.

Por otro lado, es crucial que los conjuntos reducidos sean representativos del conjunto original. Es interesante ver en un gráfico los puntos que forman parte de la reducción frente a los que se quedan fuera. Esto se muestra en las figuras 4, 5 y 6. En ellas se han representado las dos primeras características de los datos. En azul se muestran los datos que forman parte de la reducción y en rojo los que no. Se

observa que la reducción con k-medoids (Fig. 4) está bastante sesgada. Tiende a coger puntos de zonas con mucha densidad y deja fuera zonas enteras de baja (pero no despreciable) densidad. No parece que este método dé una reducción representativa del conjunto original, y por tanto no lo usaremos en los siguientes apartados.

En cambio, k-medias (Fig. 5) quita algunos puntos de las zonas con mucha densidad y mantiene puntos un poco más raros", de zonas de no mucha densidad. Esto hará que los modelos supervisados aprendan tanto las zonas densas como las menos densas. Lo mismo ocurre con el método de mezcla de Gaussianas (Fig. 6). En este último caso, además, los puntos parecen distribuidos más aleatoriamente. Dado que en principio ambos métodos parecen dar conjuntos representativos, para continuar en los siguientes apartados usaremos conjuntos obtenidos mediantes k-medias y mezcla de Gaussianas.

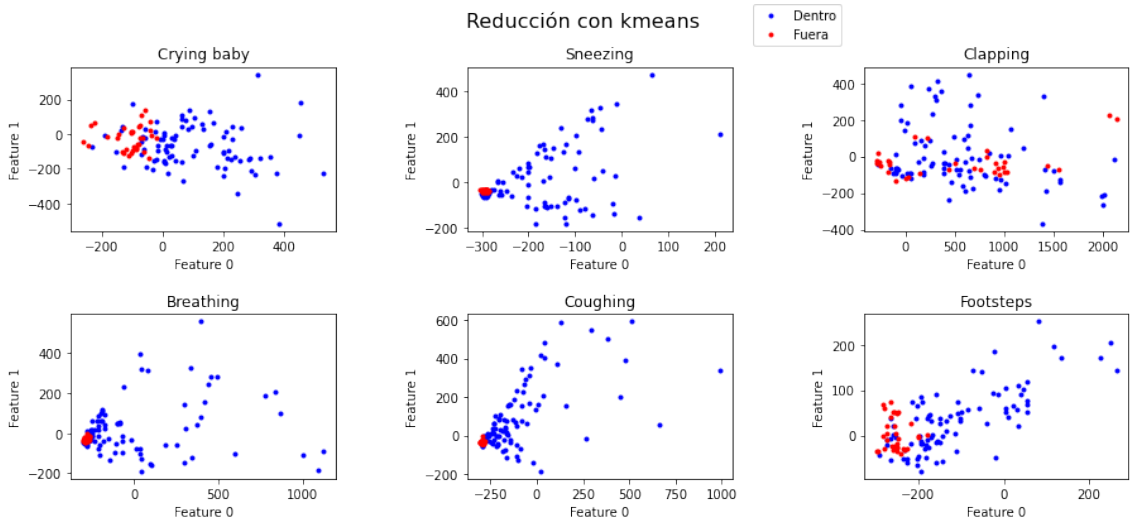

Figura 5: Resultados de la reducción mediante k-medias.

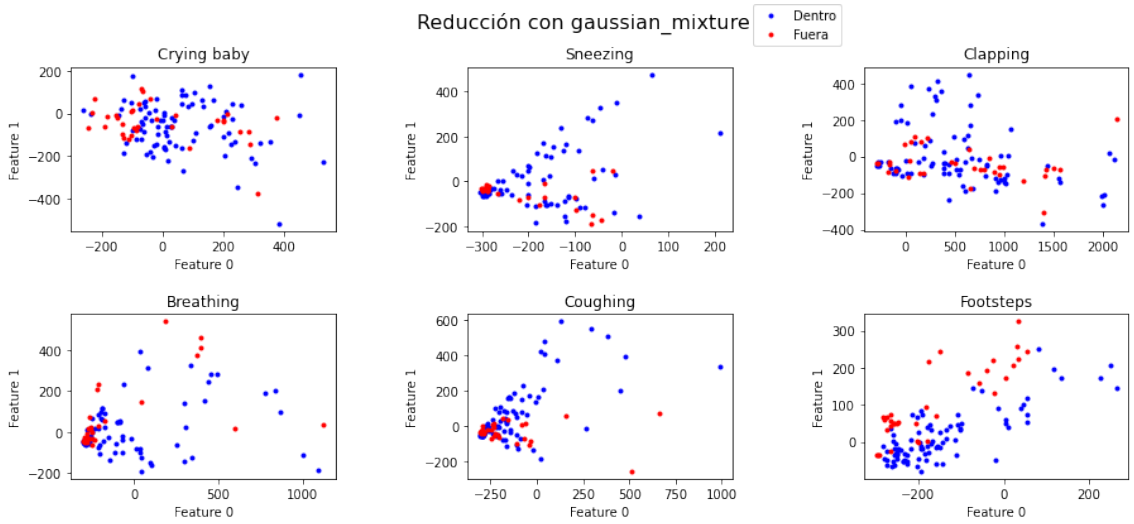

Figura 6: Resultados de la reducción mediante mezcla de Gaussianas.

5. Aprendizaje supervisado

Una vez que contamos con conjuntos de entrenamiento reducidos, pero representativos de cada clase, podemos pasar a la construcción de nuestro clasificador de eventos sonoros supervisado. En este apartado desarrollaremos modelos mediante las tres principales técnicas de aprendizaje supervisado vistas a lo largo de la asignatura, es decir, árboles de decisión (DT), máquinas de vectores de soporte (SVM) y perceptrones multicapa (MLP).

A la hora de comparar modelos entre sí nos basaremos en métricas de clasificación. En primer lugar se evaluará la puntuación (score), tanto de entrenamiento como de test,que en estos modelos es la precisión general. Asimismo se tendrá en cuenta el coeficiente kappa de Cohen y el área bajo la curva de ROC. Finalmente se mostrará la matriz de confusión.

En general si se construyen los modelos por defecto, se sobreajustan. En la totalidad de los casos, se obtienen mejores o iguales resultados haciendo un ajuste de hiperparámetros, que realizando la ejecución por defecto. Nos valdremos de la clase GridSearchCV para realizar el ajuste. Es de lo más importante asignar un conjunto de parámetros a ajustar de lo más heterogéneo. A modo de ejemplo, en el árbol de decisión habrá que probar los dos criterios de poda, árboles poco profundos, más profundos, sin límite de profundidad, variar el número de muestras por hoja, etc. En suma, la idea es que no se quede nada sin probar, pero sin caer en entrenamientos redundantes o muy similares. Del mismo modo, se pondrán a prueba los dos conjuntos seleccionados en el apartado anterior.

En este afán de intentar todas las técnicas posibles, generamos un modelo de árbol de decisión con post-purning. Sin embargo, la puntuación obtenida fue peor que con el método estándar.

En la tabla 7 se muestran los resultados de las pruebas tras el ajuste de hiperparámetros. Se observa como el uso de métodos más avanzados como las redes neuronales no supone necesariamente una mejor clasificación, de hecho roza ligeramente el sobreajuste. Podemos asegurar en base a estos resultados que el modelo que mejor resuelve este problema son las SVM, con una precisión del 55 % y una AUC de 0,87.

Resultados		DT	SVM	MLP
Train	OA	0.63	0.84	**0.95**
	Kappa	0.56	0.80	**0.94**
	AUC	0.89	**0.99**	**0.99**
Test	OA	0.48	**0.55**	0.47
	Kappa	0.38	**0.46**	0.36
	AUC	0.78	**0.87**	0.81

Tabla 7: Resultado de las ejecuciones de los métodos de aprendizaje supervisado tras ajustar sus hiperparámetros.

6. Ranking de características

Cuando trabajamos con árboles de decisión podemos calcular la importancia de cada característica en base a ganancia promedio de pureza por división (conocida como importancia de Gini) y a partir de esos valores establecer un ranking. Sin embargo, en aprendizaje automático están los que se conocen como modelos de caja negra, debido a su difícil interpretabilidad. En este trabajo vamos a valernos de la herramienta SHAP (SHapley Additive exPlanations [4]) para construir los rankings de características. En concreto utilizaremos el método de kernel (KernelExplainer [4] [3]), debido a que es agnóstico al modelo, es decir, no depende del método de ML, funciona con cualquier función de predicción. Una desventaja de este explainer es su ineficiencia frente a cualquier explainer específico. Afortunadamente para este problema no se requiere una gran cantidad de tiempo de ejecución. Antes de evaluar los resultados, es necesario introducir una serie de conceptos de teoría de juegos para comprender el funcionamiento de SHAP.

En juego cooperativo una coalición de jugadores busca la manera más justa de adjudicar las ganancias totales teniendo en cuenta la contribución individual de cada jugador. Contamos con una función característica v, que dada una coalición devuelve el valor que produce. Por tanto podemos definir la contribución marginal como el beneficio (o pérdida) que aporta un miembro a la coalición, y calcularlo a partir de v.

$$v(S \cup \{i\}) - v(S)$$

El valor de Shapley de un jugador i se obtiene mediante el promedio de sus contribuciones.

$$\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S)) \tag{1}$$

Donde n es el número de elementos del conjunto y S una coalición a la que no pertenece i. Como promedio tenemos el número de permutaciones que pueden formarse con S dividido por el número total de permutaciones. El valor de Shapley goza de una serie propiedades importantes: eficiencia (la suma de las contribuciones es igual a la ganancia $\sum_{i \in N} \varphi_i(v) = v(N)$), anonimato (no se tiene en cuenta la etiqueta del jugador, solo la ganancia que aporta), simetría, linealidad ($\varphi_i(v + w) = \varphi_i(v) + \varphi_i(w)$) y jugador nulo ($v(S \cup \{i\}) = v(S), \forall S \implies \varphi_i(v) = 0$).

En este contexto, cada variable es un jugador y su valor de Shapley es la manera en ha contribuido a la clasificación. Vamos a explicar la puntuación final a través de un modelo lineal:

$$g(z) = \varphi_0 + \sum_{j=1}^{M} \varphi_j z_j \tag{2}$$

El parámetro z determina está compuesto de M variables binarias, siendo M el número de características, cuando z_j toma el valor de 1 quiere decir que pertenece

a la coalición, si vale 0, no pertenece, φ_j es el valor de Shapley de característica j-esima y φ_0 es la esperanza del output sobre los datos de entrenamiento.

El modelo ha sido entrenado con todas las características, por tanto a la hora de calcular el valor de Shapley de una coalición no podemos eliminar esas variables del modelo. Tampoco podemos asignar el mismo valor arbitrario (como podría ser 0 o NA) a todos los elementos de las variables no pertenecientes a la coalición. La solución consiste en asignar a las variables ajenas a la coalición valores que toman en otra instancia (background), así se reduce la dependencia de esas características.

Las instancias del modelo lineal se terminan ponderando por el siguiente kernel:

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)} \tag{3}$$

Donde M es el número total de variables y $|z'|$ el de miembros de la coalición. De esta manera damos más peso a las coaliciones con cardinalidad más cercana a 1 o a $M-1$, que son las que mejor explican el comportamiento de una sola característica.

Finalmente quedaría realizar el problema de regresión lineal ponderada, es decir, optimizar la siguiente función:

$$L(f, g, \pi_x) = \sum_{z' \in Z} [\, f(h_x(z')) - g(z')\,]^2 \pi_x(z') \tag{4}$$

f sería la función de predicción de nuestro modelo, h_x la función que devuelve que toma x en la variable j de si $z_j = 1$ y un valor de esa característica en otra instancia aleatoria si $z_j = 0$, el resto de elementos ya se han descrito anteriormente.

Como se ha dicho anteriormente, este método resulta ineficiente porque existe un total de 2^M combinaciones de coaliciones. Cabe tener en cuenta que los pesos de las mayorías de combinaciones (las que tienen un número de miembros alejado de 1 o $M-1$) son tan pequeños que se omiten esos cálculos.

Una vez introducidos los conceptos más teóricos, evaluaremos los resultados sobre los modelos. En la figura 8 se muestran los diagramas de barras, el primero se ha obtenido en base a la importancia de Gini en nuestro árbol de decisión, por SHAP. Las gráficas de SHAP nos indican la importancia que tiene cada variable en la predicción de cada clase. Lo primero que observamos es que las características más importantes para árboles de decisión coinciden independientemente del método que utilicemos. Como el método de ganancia promedio de pureza es instantáneo, mientras que el de valores de Shapley necesita tiempo, para futuros análisis es aconsejable evitar SHAP en árboles de decisión, a no ser que sea de interés el aporte para la predicción de cada clase. El orden de los rankings no corresponde con el de las componentes principales. Ahora bien, no se consideran importantes las componentes a partir de la 34, y es natural teniendo en cuenta que son las que menos valor aportan.

En la clasificación de pisadas interviene más la componente 2, que la 0, a pesar de aportar menor información. En el resto de categorías la que más contribuye a la

predicción, independientemente del modelo, es la componente 0, como era esperable. Los aplausos son la clase con valores de Shapley más altos.

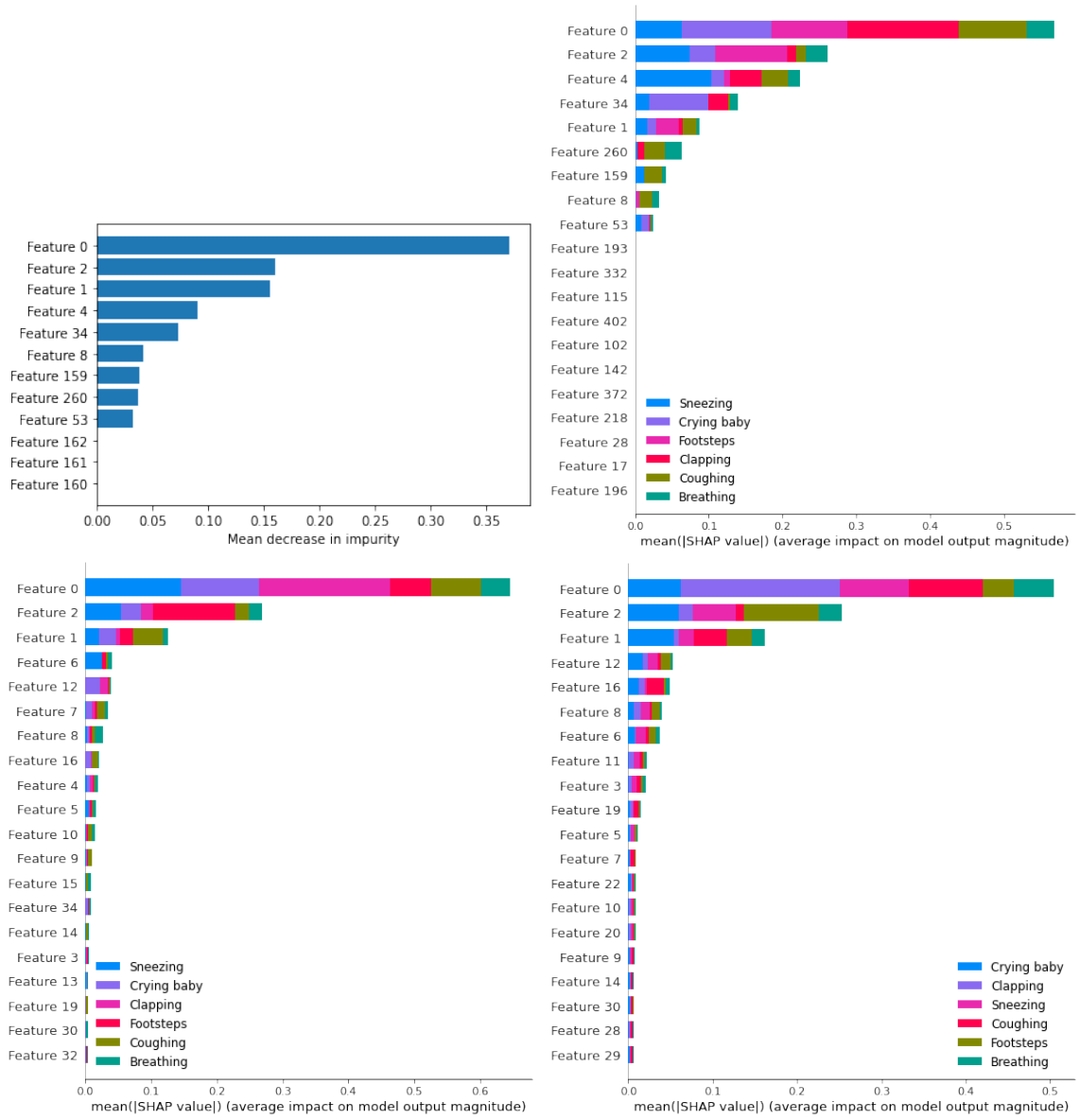

Figura 8: Ranking de características de cada método de aprendizaje supervisado. En la zona superior figuran los rankings del árbol de decisión calculado mediante importancia de Gini (izq) y SHAP (der). En la parte inferior se encuentran los de SVM (izq) y MLP (der).

Una vez elaborados los rankings de importancia de cada variable vamos a evaluar los modelos habiendo reducido las variables. En la tabla 9 se muestran los resultados. Se observa como tras una drástica reducción de variables, no sólo se mantienen los resultados, sino que en algunos casos llegan hasta a mejorar. El récord lo siguen ostentando las SVM, ahora con una puntuación de 0.57.

Resultados		DT (9 var)	SVM (20 var)	MLP (20 var)
Train	OA	0.63	0.70	**0.71**
	Kappa	0.56	0.64	**0.65**
	AUC	0.89	**0.95**	0.92
Test	OA	0.48	**0.57**	0.51
	Kappa	0.38	**0.49**	0.42
	AUC	0.78	**0.86**	0.85

Tabla 9: Resultado de las ejecuciones de los métodos de aprendizaje supervisado tras ajustar seleccionar características.

7. Ensembles

En esta sección comprobaremos si al juntar diversos clasificadores y utilizar métodos de *ensembles* conseguimos mejorar los resultados. Los métodos en cuestión serán algunos de los vistos en clase: bagging, random forest, adaboost y stacking. Al igual que antes, ajustamos cada modelo con GridSearchCV para dar con el mejor posible.

A la vista de que los clasificadores anteriores eran buenos, trataremos de agruparlos para optimizarlos. Empezaremos por los árboles de decisión. Si hacemos bagging con cien árboles de decisión obtenemos una precisión similar a la de SVM, 6 puntos por encima del árbol por defecto. Sin embargo, hacer bagging de nuestro modelo de SVM es equivalente a utilizar una única SVM (Tabla 10). En lo que respecta a la selección de características, en árboles de decisión, tanto en bagging como por separado, no supone una mejora de puntuación. Lo mismo sucede en SVM.

Siendo que un conjunto de árboles de decisión funciona mejor que un solo árbol, a primera vista podría parecer que un random forest sería una buena decisión. Sin embargo, lo cierto es que, por más que intentemos ajustar sus parámetros, el modelo siempre termina sobreajustandose o devolviendo peores resultados que el sobreajustado. Aunque sus puntuaciones en test sean similares a las del árbol decisión, debería descartarse por el sobreajuste. Esto se deba, posiblemente, a que es un modelo muy complejo y con una alta varianza.

Un modelo clásico de *ensembles* es el AdaBoost, que consiste en entrenar varios clasificadores débiles, cada uno emitirá un voto, que será ponderado y el resultado final será lo que considere la mayoría. En scikit learn los clasificadores débiles por defecto son arboles de decisión de uno de profundidad. Aún habiendo ajustado hiperparámetros y probando con los distintos conjuntos, las puntuaciones son las peores de todas. Todo apunta a que este no es el modelo adecuado para este problema.

En último lugar quedaría probar stacking. Este procedimiento trata de agrupar los tres modelos de aprendizaje supervisado vistos a lo largo de este trabajo en un único modelo. El de estos clasificadores de primer nivel será utilizado para alimentar a un meta-clasificador de segundo nivel, en este caso un regresor logístico, que de-

terminará la decisión final. Como se puede comprobar en la tabla 10, los resultados son muy similares a los de SVM por separado, además de que se aprecia sobreajuste.

Resultados		SVM	Bagging DT	Bagging SVM	RF	AdaBoost	Stacking
Train	OA	0.84	0.86	0.74	**1.00**	0.62	0.95
	Kappa	0.80	0.84	0.68	**1.00**	0.54	0.94
	AUC	0.99	0.99	0.95	**1.00**	0.86	**1.00**
Test	OA	0.55	0.54	0.55	0.48	0.40	**0.56**
	Kappa	0.46	0.44	0.46	0.37	0.28	**0.47**
	AUC	**0.87**	0.83	0.86	0.81	0.76	0.86

Tabla 10: Comparación de puntuaciones entre SVM y los diferentes modelos de ensembles.

La conclusión general que podemos extraer de esta sección es que no siempre un modelo más complejo o la unión de varios modelos nos va a otorgar un mejor resultado. A modo de excepción tenemos los árboles de decisión, que es mejor contar con un conjunto que con uno solo. Ahora bien, sigue siendo mejor opción las SVM.

8. Conclusiones

Como era sabido, cuando recibes un conjunto de datos y tienes que elaborar un clasificador, no existe un método universal que sirva de solución, sino que en función de los datos habrá algunos que clasifiquen mejor que otros. En esta ocasión parece que el modelo que mejor se adapta a este problema son las SVM con selección de características, si bien hay otros que nos conceden resultados parecidos (por ejemplo el bagging de árboles).

Como se puede comprobar, para determinar cual es el mejor método se ha tenido que atravesar un largo proceso. Ya desde el comienzo tuvimos que decantarnos por una forma de reducir la dimensionalidad del audio, ya que con los datos en bruto no es posible trabajar. Se tenía que encontrar la estrategia que mejor redujera sin perder información y la que mejor satisfacía ese objetivo era MFCC. Más adelante, después de haber establecido el ranking de características, se ha visto que si seguíamos reduciendo la dimensionalidad con un subconjunto reducido con las mejores variables, no solo se acelera la ejecución, sino que además el clasificador es más preciso.

En lo que respecta a la parte de aprendizaje no supervisado, el análisis de distintos métodos de clustering ha puesto de manifiesto su potencial para ser usados como métodos de muestreo en distribuciones arbitrarias, permitiendo reducir el número de instancias en grandes conjuntos de datos manteniendo sus propiedades. De igual forma, nos ha mostrado algunos de sus puntos débiles, en concreto la dificultad que presentan los algoritmos de agrupamiento cuando los grupos subyacentes tienen bastante solapamiento.

Una vez más recalcamos la importancia de ajustar los parámetros y no lanzar

modelos por defecto, de lo contrario es muy posible que salgan sobreajustados. Así como, cuando se emplean modelos complejos y de alta varianza como las combinaciones de clasificadores (ensembles), es bastante probable caer en sobreajuste. Lo que no significa que sean malos métodos, lo que ocurre es que en conjuntos reducidos como este se ha visto que no son tan eficientes.

Referencias

[1] Kiyoharu Aizawa, Yuichi Nakamura, and Shin'ichi Satoh, editors. *Advances in multimedia information processing: PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30-December 3, 2004: proceedings*. Number 3331-3333 in Lecture notes in computer science. Springer, Berlin ; New York, NY, 2004. `https://doi.org/10.1007/b104114`.

[2] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[3] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017. `https://arxiv.org/abs/1705.07874/` [Último acceso 20 de Enero de 2022].

[4] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/` [Último acceso 13 de Junio de 2021].

[5] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.