Práctica de Audio

Procesamiento Digital del Audio y Habla

H l

Curso 2020-2021

Sofía Bonifasi Asturias

Sandra Cea Torrescana

Rogelio Sansaloni Sanjuan

ÍNDICE

[I. Estudios 3](#_Toc59144707)

[A. KNN 3](#_Toc59144708)

[B. CART 3](#_Toc59144709)

[C. SVM 4](#_Toc59144710)

[D. GMM 5](#_Toc59144711)

[II. Resultados y Tablas 7](#_Toc59144712)

[A. Tabla de KNN 9](#_Toc59144713)

[B. Tabla de CART 9](#_Toc59144714)

[C. Tabla de SMV 10](#_Toc59144715)

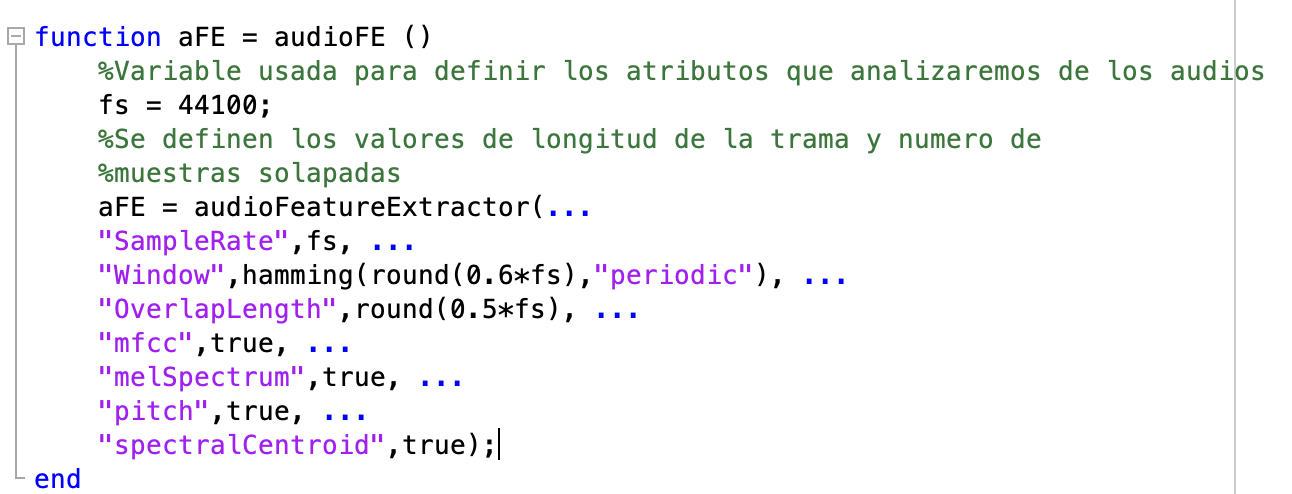
[D. Tabla de GMM 10](#_Toc59144716)

[III. Matriz de Confusión 11](#_Toc59144717)

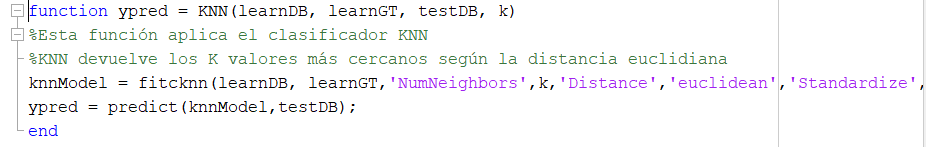
# Estudios

## KNN

Para el KNN, los parámetros que editamos fueron: el Window y el OverlapLength. También agregamos los parámetros de "mfcc", "melSpectrum", "pitch" y "spectralCentroid" con valores de *true*.



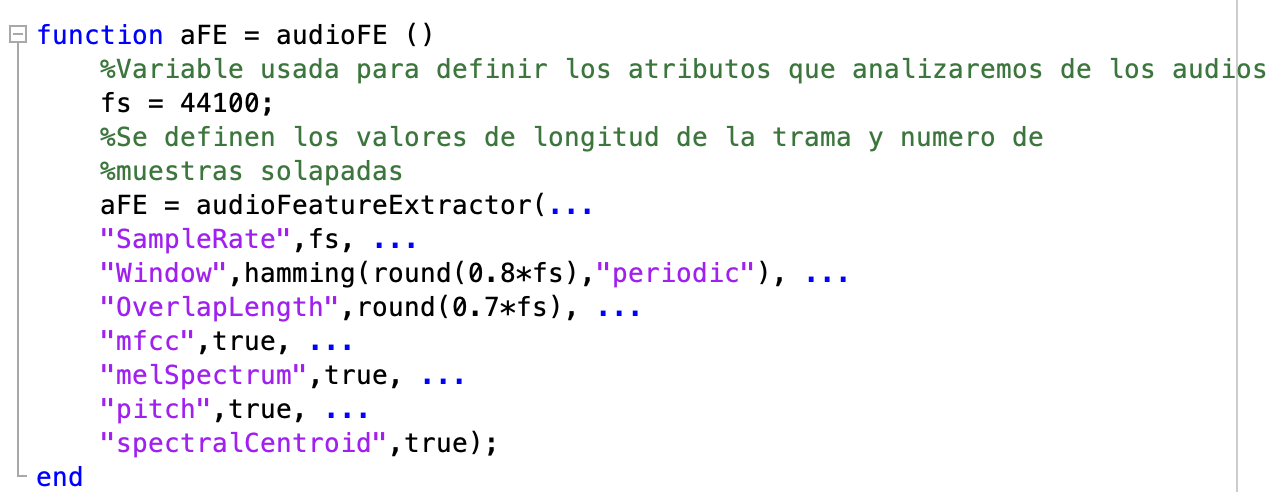
El código en Matlab de la función que implementa KNN es el siguiente:



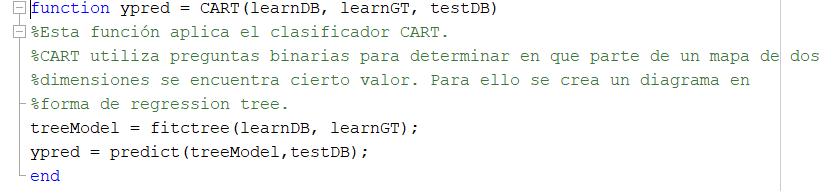
Tal y como indica el enunciado, creamos un modelo KNN mediante *fitcknn*, al que le pasamos la distancia k (definida como *knn\_K* en *practica\_audio*) y usamos *predict* para obtener el *testGT* predicho por el clasificador.

## CART

Para el CART, los parámetros que editamos fueron: el Window y el OverlapLength. También agregamos los parámetros de "mfcc", "melSpectrum", "pitch" y "spectralCentroid" con valores de *true*.



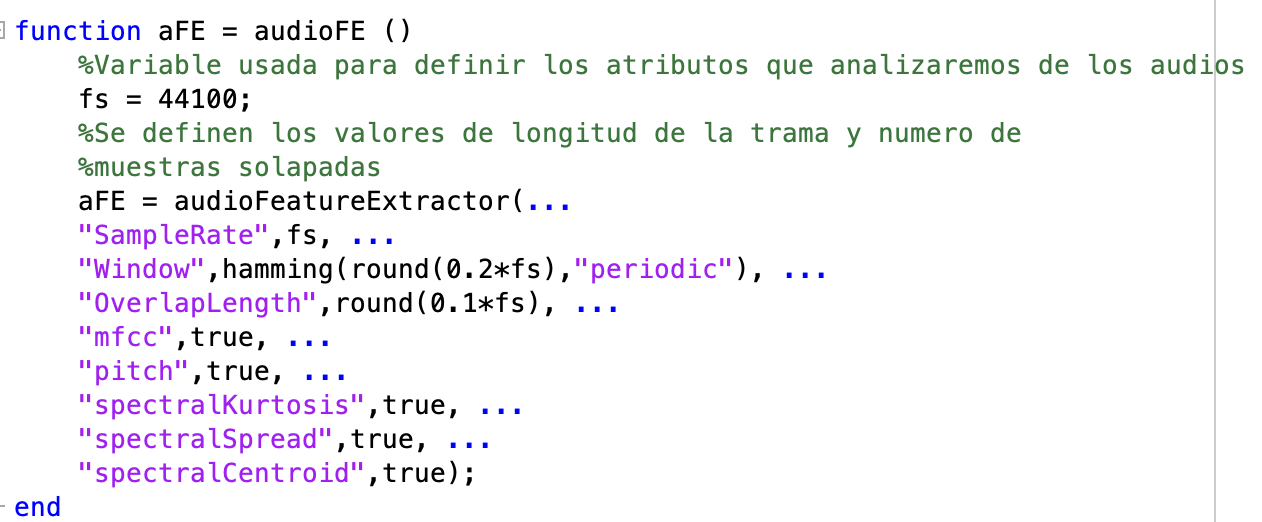
El código en Matlab de la función que implementa CART es el siguiente:



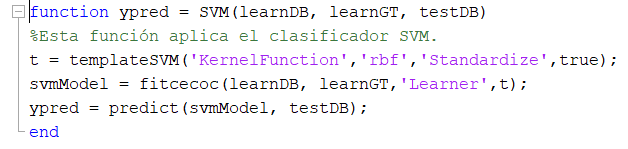
Tal y como indica el enunciado, creamos un modelo CART mediante *fitctree* y usamos *predict* para obtener el *testGT* predicho por el clasificador.

## SVM

Para el SVM, los parámetros que editamos fueron: el Window y el OverlapLength. Además, agregamos los parámetros de "mfcc", "pitch", " spectralSpread", “spectralKurtosis” y "spectralCentroid" con valores de *true*.



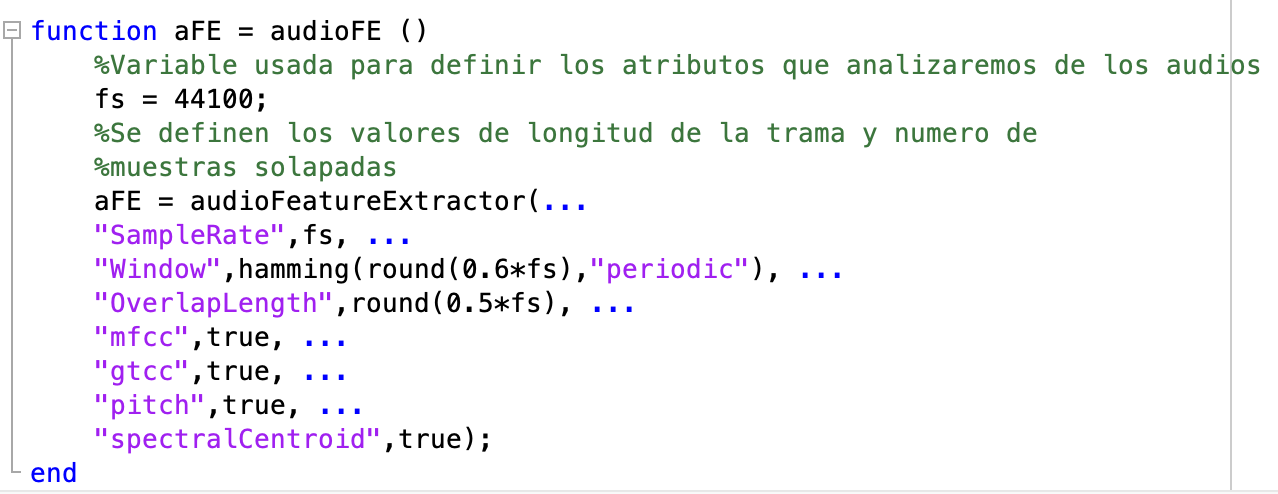
El código en Matlab de la función que implementa SVM es el siguiente:



Tal y como indica el enunciado, creamos un modelo SVM mediante *fitcecoc*, al que le pasamos el template generado, el cual obtenemos indicándole que queremos usar la función Kernel. Posteriormente usamos *predict* para obtener el *testGT* predicho por el clasificador.

## GMM

Para el GMM, los parámetros que editamos fueron: el Window y el OverlapLength. Además, agregamos los parámetros de "mfcc", "pitch", "gtcc" y "spectralCentroid" con valores de *true*.



El código en Matlab de la función que implementa GMM es el siguiente:



Esta función es bastante distinta a la que usamos para los demás clasificadores. Inicialmente se generan diversos modelos GMM, uno por cada categoría. La forma de entrenar estos modelos es conseguir generar una matriz de aprendizaje(*data\_category*) únicamente con los atributos de aquellas tramas de la categoría que me interese. Para generar el modelo se usa *fitgmdist,* al que le hemos puesto una regularización de 0.1, y una covarianza diagonal para que pudiera converger.

Posteriormente, lo que se hace es generar diversas probabilidades con *pdf* para cada trama de *testDB* y para cada categoría, de forma que se obtiene un vector de 15 columnas (15 categorías) de probabilidad para cada trama de *testDB*. Luego se obtiene la máxima probabilidad (función *max*) para cada trama, es decir, a que categoría es más probable que pertenezca. Se repite el proceso para cada trama de *testDB* y se obtiene el *testGT* predicho.

# Resultados y Tablas

Para realizar los estudios, hicimos dos pruebas para cada uno de los clasificadores; la primera prueba con el 5% de los datos y la segunda con el 50%. Una vez hechos los estudios, obtuvimos la siguiente tabla con el porcentaje de exactitud para cada método de clasificación de acuerdo al porcentaje de datos utilizado:

|  |  |  |
| --- | --- | --- |
| Clasificador | Porcentaje de Datos | Porcentaje de Exactitud |
| KNN | 5% | 80.60% |
| 50% | 48.21% |
| CART | 5% | 83.37% |
| 50% | 54.07% |
| SMV | 5% | 78.80% |
| 50% | 49.89% |
| GMM | 5% | 70.97% |
| 50% | 42.39% |

Como podemos ver, el método más exacto fue el CART. Para realizar cada estudio, primero fue necesario quitar, agregar y editar parámetros con solo el 5% de los datos para que la computadora con la que se hacían las pruebas no se saturara. Una vez los estudios regresaban más de 70% de exactitud, pudimos hacer las pruebas finales con el 50% de los datos. A continuación, mostraremos las tablas que generó Matlab con cada estudio usando el 50% de datos.

Por tanto, el orden la de las *practica\_audio* es el siguiente:

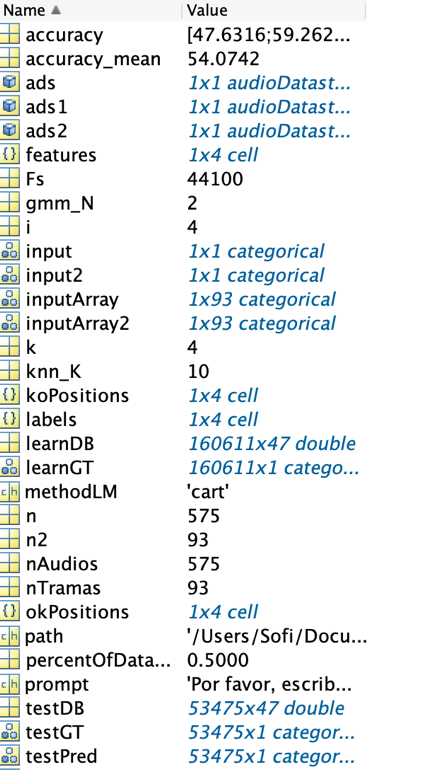
* *practica\_audio1:* clasificador CART con los parámetros de *audioFeatureExtractor* explicados anteriormente.
* *practica\_audio2:* clasificador KNN con los parámetros de *audioFeatureExtractor* explicados anteriormente.
* *practica\_audio3:* clasificador SVM con los parámetros de *audioFeatureExtractor* explicados anteriormente.
* *practica\_audio4:* clasificador GMM con los parámetros de *audioFeatureExtractor* explicados anteriormente.

También se incluye un practica\_audio sin ninguna enumeración, que hemos usado con la configuración del mejor accuracy (es decir, *practica\_audio1*) para obtener la matriz de confusión.

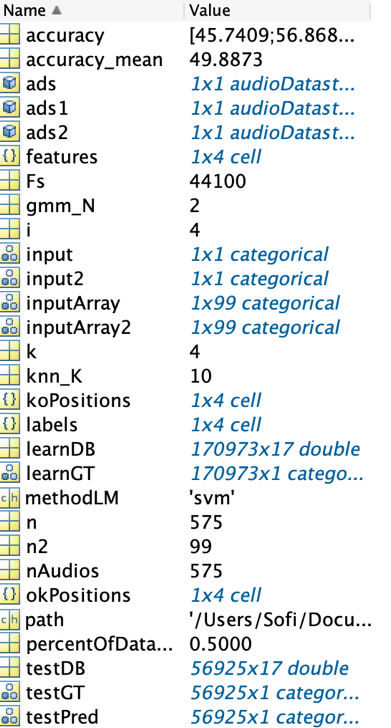
A continuación, se muestran las matrices obtenidas con cada clasificador:

## Tabla de KNN

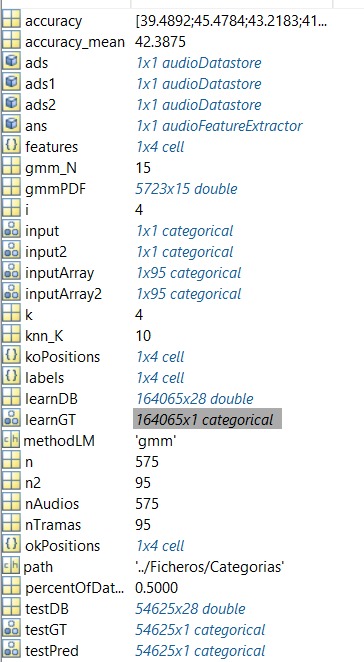
## Tabla de CART



## Tabla de SMV

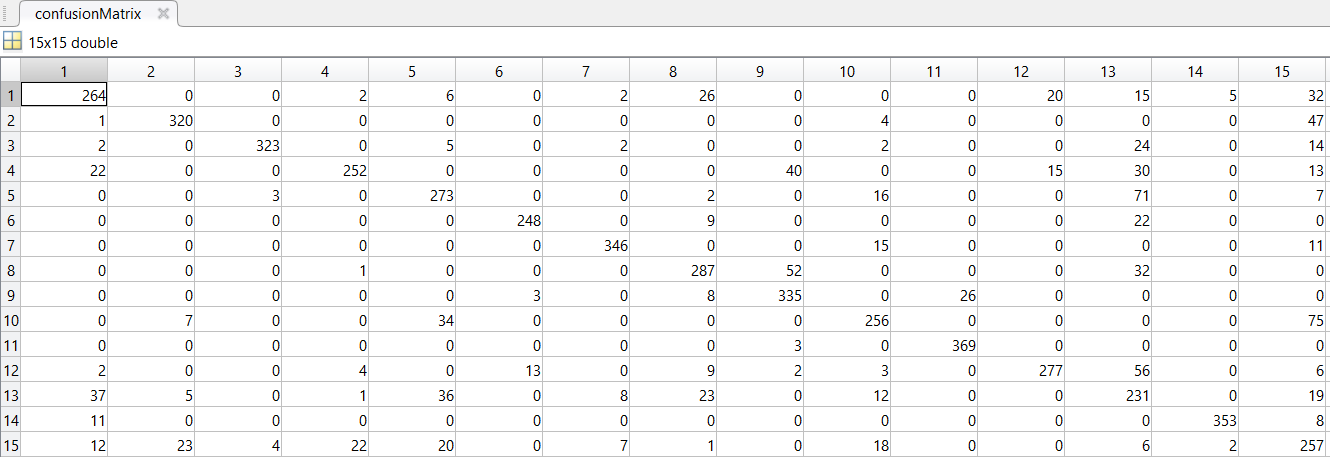


## Tabla de GMM

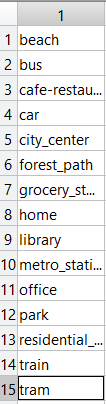


# Matriz de Confusión

La matriz de confusión que se obtiene con el mejor resultado, es decir, con practica\_audio1 con el 5% de los datos, es la siguiente:



Los índices de las filas y columnas representan el nombre de las categorías ordenadas alfabéticamente, como se puede observar en la siguiente matriz de categorías:



El código para obtener la matriz de confusión es el siguiente, que se encuentra en el archivo *practica\_audio.m:*

