# **PyAudioAnalysis**

* Extrae las características del audio (espectograma…)
* Clasifica sonidos
* Entrena y utiliza modelos de regresión (aplicación en Reconocimiento de emociones)
* Aplica reducciones de la dimensionalidad para visualizar datos de audio similares (similitud de datos)

## Descripción de la librería

* 6 archivos de Python: audioAnalysis, audioFeatureExtraction, audioTrainTest, audioSegmentation, audioBasicIO, audioVisualization.

1. **audioAnalysis.py**: este archivo implementa la interfaz de línea de comandos de las funcionalidades básicas de la biblioteca, junto con algunas funcionalidades de grabación.
2. **audioFeatureExtraction.py**: aquí es donde se implementa toda la extracción de características de audio. En total, se computan 21 características a corto plazo, mientras que también se implementa una técnica de ventanas de mediano plazo para extraer estadísticas de las características de audio.
3. **audioTrainTest.py**: este archivo implementa los procedimientos de clasificación de audio. Contiene funciones que se pueden utilizar para entrenar una máquina de vectores de soporte (SVM) o un clasificador k-vecino más cercano (kNN). Además, las funciones y los scripts de envoltura se proporcionan para cuestiones generales de entrenamiento, evaluación y normalización de funciones.
4. **audioSegmentation.py**: este archivo implementa funcionalidades de segmentación de audio, por ejemplo, clasificación y segmentación de segmento de tamaño fijo, diarización de altavoz, etc.
5. **audioBasicIO.py**: este archivo implementa algunas funcionalidades básicas de IO de audio, así como también las conversiones de archivos
6. **audioVisualization.py**: el propósito de este conjunto de funciones es producir visualizaciones de contenido representativas y fáciles de usar.

En la carpeta data/, se proporcionan un par de archivos de muestra de audio, junto con algunos modelos entrenados de SVM y kNN para tareas de clasificación particulares (por ejemplo, voz frente a música, clasificación de género musical, etc.).

### 1.- Extracción de características.

Hay dos etapas en la metodología de extracción de características de audio:

* **Extracción de características a corto plazo (Short-term feature extraction)**: se implementa en la función stFeatureExtraction() del archivo audioFeatureExtraction.py. Divide la señal de entrada en ventanas a corto plazo (frames) y calcula una serie de características para cada frame. Este proceso conduce a una secuencia de vectores de características a corto plazo para toda la señal.
* **Extracción de características a medio plazo (Mid-term feature extraction)**: en muchos casos, la señal se representa mediante estadísticas sobre las secuencias de las características extraídas a corto plazo descritas anteriormente. Con este fin, la función mtFeatureExtraction() del archivo audioFeatureExtraction.py extrae un número de estadísticas (por ejemplo, media y desviación estándar) sobre cada secuencia de características a corto plazo.

El total del número de características a corto plazo (short-term) implementado en pyAudioAnalysis es de 34. En la siguiente tabla se muestra una lista completa de las características implementadas:

|  |  |  |
| --- | --- | --- |
| **Feature ID** | **Feature Name** | **Description** |
| 1 | Zero Crossing Rate  (Tasa de cruce cero) | La tasa de cambio de signo de la señal durante la duración de un frame particular. |
| 2 | Energy  (Energía) | La suma de los cuadrados de los valores de la señal, normalizados por la respectiva longitud del frame. |
| 3 | Entropy of Energy  (Entropía de la energía) | La entropía de las energías normalizadas de los sub-frames. Puede ser interpretado como una medida de cambios abruptos. |
| 4 | Spectral Centroid  (Centroide espectral) | Centro de gravedad del espectro. |
| 5 | Spectral Spread  (Extensión espectral) | El segundo momento central del espectro. |
| 6 | Spectral Entropy  (Entropía espectral) | Entropía de las energías espectrales normalizadas para un conjunto de sub-frames. |
| 7 | Spectral Flux  (Flujo espectral) | La diferencia cuadrática entre las magnitudes normalizadas del espectro de dos frames consecutivos. |
| 8 | Spectral Rolloff  (Desplazamiento espectral) | La frecuencia por debajo de la cual se concentra el 90% de la distribución de la magnitud del espectro. |
| 9-21 | MFCCs | Coeficientes Cepstrales de Frecuencia Mel de una representación cepstral donde la frecuencia de las bandas no es lineal, pero se distribuye según la escala mel. |
| 22-33 | Chroma Vector  (Vector chroma) | Una representación de 12 elementos del espectro de energía donde los contenedores representan 12 clases de tonos igualmente atemperados de música tipo occidental (espaciado de semitonos). |
| 34 | Chroma Deviation  (Desviación cromática) | La desviación estándar de los 12 coeficientes de chroma. |

### 2.- Clasificación y Regresión

Se proporciona una funcionalidad de clasificación de segmentos en la biblioteca, para entrenar y usar modelos supervisados ​​que clasifican un segmento de audio desconocido en un conjunto de clases predefinidas (por ejemplo, música y voz). Con este fin, el audioTrainTest.pyarchivo implementa los siguientes tipos de clasificadores:

* k Vecino más cercano kNN (implementado en la biblioteca)
* SVM
* Random Forests
* Extra Trees
* Gradiet Boosting

Regresión

Aparte de la calsificación, la regression puede ser bastante más importante en el análisis de audio, por ejemplo, en el contexto de reconocimiento de emociones del habla, donde el estado emocional no es una clase discreta sino un valor de medición real (por ejemplo, excitación o valencia). audioTrainTest.py proporciona también funcionalidades relacionadas con la regresión.

La función utilizada para este fin es **featureAndTrain (listOfDirs, mtWin, mtStep, stWin, stStep, classifierType, modelName, computeBEAT)** o **featureAndTrainRegression** (clasificación/regresión) del paqueteTrainTest.py.

ListOfDirs-> El primer argumento es una lista de rutas de directorios. Cada directorio contiene una clase de audio de señales cuyas muestras se almacenan en archivos WAV separados.

mtWin, mtStep, stWin, stStep -> Después, la función toma el tamaño y el paso de la ventana temporal y su tamaño a medio plazo (mtWin, mtStep) y el paso de la ventana temporal y su paso a corto plazo (stWin, stStep).

Los argumentos classifierType y modelName están asociados al tipo y nombre del clasificador. Este último también se utiliza como nombre del archivo donde se almacena el modelo para uso futuro (consulte las siguientes secciones sobre clasificación y segmentación).

computeBEAT -> Finalmente, el último argumento es booleano, se establece en True si se deben calcular las características relacionadas con el latido a largo plazo (por ejemplo, para tareas de clasificación de música). Además, también se crea un archivo ARFF (con el mismo nombre que el modelo), donde se almacena todo el conjunto de vectores de características y etiquetas de clase respectivas.