AFI - ESCUELA DE FINANZAS

Máster en *Data Science* y *Big Data*

TFM - TRABAJO FIN DE MÁSTER

Modelo de clasificación de géneros musicales basado en recuperación de información musical (MIR) y análisis de espectrogramas

Alumno: Ludwig Gerardo Rubio Jaime

Tutor AFI: Borja Foncillas García

Junio – 2019

Tabla de contenidos

1. Introducción

1.1 Aplicación de negocio

La aplicación de algoritmos de *machine learning* en la industria musical ha tenido un crecimiento considerable durante los últimos años, siendo integrados a distintos procesos de la industria que van desde la composición, la mezcla, la producción, la remasterización, la venta, la recomendación, etc.

Actualmente, las plataformas de *music streaming* son uno de los claros ejemplos de adaptación de la industria musical al contexto digital, y que, debido a su volumen de usuarios y contenido, se enfrentan a nuevos retos de organización y recomendación de contenido.

Aun cuando la tendencia de estas plataformas para recomendar música es la de utilizar sistemas de recomendación basado en filtros colaborativos, persiste la necesidad de contar con algoritmos que mejoren la organización, así como la recomendación de nueva música para la cual aún no existen datos de usuarios que han interactuado con ella.

Por tal razón, en este proyecto se exploran alternativas que faciliten la organización de música de forma más eficiente y automatizada basada en **algoritmos de machine learning**, que utilicen características de la música como una estrategia para facilitar su escalabilidad a un sistema de recomendación musical.

Debido a que el género musical es un dato casi siempre intrínseco a los metadatos de archivos de música, y con el objetivo de contar con un criterio de evaluación del algoritmo, se propone generar:

Un modelo de clasificación de géneros musicales basado en recuperación de información musical (*MIR*) y análisis de espectrogramas.

1.2 Estado del Arte

El uso de algoritmos de *machine learning* se ha extendido de manera considerable en la industria musical, en la actualidad, no sólo se utilizan como un herramienta, sino que en algunos casos son el sustento de un modelo de negocio entero.

Algunos de los sus usos más comunes que se pueden encontrar son:

- **Sistemas de recomendación**: Plataformas de *streaming* o redes sociales musicales que requieren de recomendación de contenido.
- ***Autotagging*** y **organización**: Tiendas de música, plataformas de *streaming*, bancos de música y letras, etc.
- **Huella digital**: Detección de derechos de autor, reconocimiento de música, etc.

- **Filtrado y reconstrucción de audio**: Producción musical, preservación de patrimonio inmaterial, remasterización, limpieza, separación de instrumentos en pistas, etc.
- **Composición de música**: Composición automática, producción musical, etc.
- **Reconocimiento de acordes**: Para la creación de tablaturas y hojas de acordes.

Algunos de las empresas más conocidas que utilizan algoritmos de *machine learning* en la industria musical son:

Shazam: Es una de las primeras empresas de la industria musical que basa su funcionamiento en algoritmos de *machine learning*. Su función es la de reconocer una canción y entregar un conjunto de metadatos asociada a ella como lo es: el nombre de la canción e información del artista, álbum, letra, vídeos, etc.

Algunos de los usos de algoritmos de *machine learning* son: Reconocimiento de la canción a través de redes neuronales, recomendación musical a través de técnicas de recuperación musical y filtros colaborativos.

Spotify: Es la plataforma de *streaming* musical con mayor número de audios y usuarios del mundo, con más de 35 millones de archivos de audio y más de 217 millones de usuarios, es de las principales referencias en uso de algoritmos de *machine learning* para la recomendación y organización de música.

Su sistema de recomendación utiliza modelos de *machine learning* basado en técnicas de recuperación de información musical, procesamiento de lenguaje natural, así como modelos de recomendación basados en factorización de matrices, con una compleja arquitectura que combina tecnologías del ecosistema *hadoop* con servicios de *Google Cloud*.

Para la solución de problemáticas llamadas como "*Cold start*", para la cual no se cuenta con información colaborativa, utilizan técnicas de recuperación musical como el que se tratará en este proyecto.

Humtap: Es una aplicación móvil que mezcla interacción social con generación de música automatizada. A través de videos con audio que generan los usuarios, una mezcla de algoritmos de *machine learning* genera música a través del *humming* grabado, la cual es compartida y valorada por usuarios de la plataforma.

Watson Beat IBM: Es un proyecto que forma parte del proyecto Watson de IBM, su función es la de generar música nueva basada en algoritmos de *machine learning*, donde se utiliza un archivo en formato MIDI (*Musical Instrument Digital Interface*) como entrada, y se obtiene como salida una composición musical basada en 8 modos distintos. Cada modo está definido en términos de una combinación limitada de opciones a elegir de los siguientes parámetros: bits por segundo (BPS), compás y distintos instrumentos musicales.

Pandora: Es un proyecto cuyo propósito es capturar la esencia de la música en su nivel más fundamental. Semejante al proyecto del genoma humano, su idea es clasificar de alguna forma la música (incluyendo algoritmos de *machine learning*), analizando los "genes" de cada canción: melodía, armonía, ritmo, instrumentación, orquestación, arreglos, letra y otros.

Con la clasificación obtenida se generan "*Cold start radio stations*" basados en *machine learning*, que consisten en estaciones de radio por usuario, donde el usuario puede seleccionar gustos musicales como géneros, artistas, canciones, etc., y a través de sistemas de recomendación híbridos que incluyen sistemas de recomendación basados en filtros colaborativos y algoritmos de *machine learning* basados en contenido, se crean dichas estaciones de radio.

Además, las listas de radio pueden ser personalizadas, y cada recomendación evaluada por el usuario permite un reentrenamiento del modelo por cada usuario.

1.3 Contexto del proyecto

El sistema de clasificación propuesto toma características de sistemas actualmente utilizados en empresas como *Spotify* y *Pandora*, como parte de un sistema complejo de múltiples algoritmos que componen su sistema de recomendación.

Para conocer el contexto de aplicación, se sitúa este modelo de clasificación y de recuperación de información musical en el contexto de flujo de datos utilizado por *Spotify* (2017).

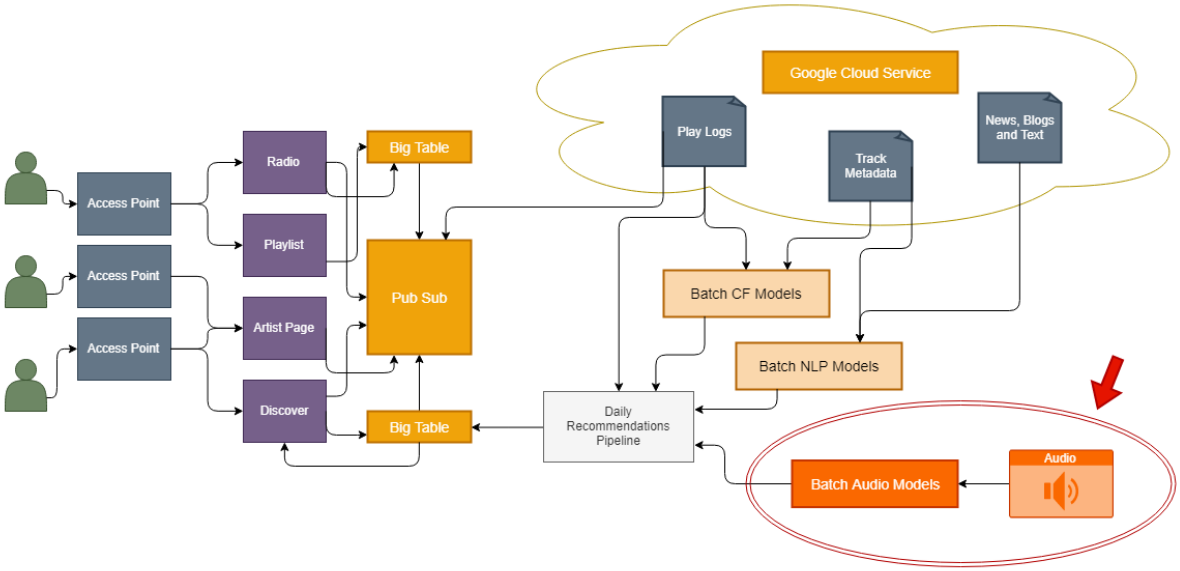


Imagen 1 – Contexto del proyecto en el diagrama de flujo de datos - Caso de uso Spotify (Recuperado de la conferencia: Machine Learning & Big Data for Music Discovery).

En color rojo se observa el contexto del modelo propuesto, el cual es utilizado en *Spotify* como parte del mecanismo de recomendación *Daily recommendations* en combinación con algoritmos de filtros colaborativos y de procesado de lenguaje natural.

2. Fases de Desarrollo

2.1 Selección de fuente de datos

El proceso de selección de fuentes de datos consistió principalmente en la investigación de distintos sets de datos y bancos de música que pudieran ser utilizados de forma libre.

Para la selección final del conjunto de sets de datos se consideraron los siguientes criterios:

- **Inclusión de archivos de audio:** Debido a que uno de los objetivos de este proyecto es el explorar técnicas de *deep learning*, se requiere que incluya total o parcialmente un archivo de audio para su procesado.
- **Volumen de datos**: Un volumen de datos considerable que permita el aprendizaje de patrones a través de la red neuronal.
- **Actualización**: Considerar que el set de datos contenga géneros, artistas y archivos de audio contemporáneas que permitan incluir en el aprendizaje automático, patrones, estilos y tendencias actuales.
- **Artículos científicos relacionados**: La existencia de trabajo previo que permita tener un punto de partida con mayor credibilidad y certeza, además de poder considerarlo como punto de referencia a mejorar.
- **Escalabilidad**: Un set de datos cuya organización y lógica permita la inclusión de nuevos archivos de música para el entrenamiento y prueba del algoritmo.

A continuación, se listan algunos de los set de datos recopilados y considerados:

Sets de datos	Archivos de audio	Última actualización	Artículos científicos	Número de canciones	Volumen de datos
Million Song Dataset	No	2012	Sí	1’000,000	500 GB
Spotify	No	No aplica	Sí	No aplica	No aplica
FMA: A Dataset For Music Analysis	Sí	2017	Sí	106,000	879 GB
TagATune	Sí	2010	No	25,000	Desconocido
USPop2002	Sí	2010	No	8,764	Desconocido
Music Audio Benchmark Dataset	Sí	2005	No	1,886	2 GB

GTZAN	Sí	2002	Sí	1,000	1.2 GB
AudioSet	No	2019	Sí	2’042,985	Desconocido
Ballrom	Sí	2006	Sí	698	400 MB

Tabla 1 - Data sets de música.

Como set de datos final se ha seleccionado ***FMA: A Dataset For Music Analysis*****,** debido a que contiene archivos de audio, es el set de datos más actualizado, con artículos científicos relacionados y con un volumen de datos considerable para intentar hacer uso de algoritmo de *deep learning*.

Algunas de las razones por las que se descartaron otros sets de datos, son: debido a que no contienen archivos de audio, son demasiado pequeños, o tiene dependencia de librerías, *APIs* y otros servicios que han dejado de existir o han sido incluidas dentro de plataformas de *streaming* privativas, limitando su escalabilidad.

Otra de las opciones que ha sido descartadas como set de datos ha sido la inclusión de técnicas de *text mining* en las letra de las canciones, esto debido a múltiples factores:

- Artículos científicos demuestran el **poco aporte de técnicas de *text mining*** a la clasificación de música por género, siendo casos excepcionales géneros como el rap y el hip-hop a los que se les aportaba una mejora predictiva.
- Debido a que se requiere de un set de datos robusto, surgió la **problemática de obtener un set de datos con un sólo lenguaje (inglés).**
- Al querer realizar el cruce del set de datos de audio, con letras, se generan una gran cantidad de *missing values*, debido a que el set de datos contiene estilos musicales y música en general sin letra.
- Poca cantidad de texto final para entrenar, debido principalmente a pocas fuentes de información, limitadas además por **derechos de autor.**

2.2 Selección de tecnologías y ambiente de pruebas

2.2.1 Tecnologías

Las tecnologías han sido seleccionadas con respecto al objetivo de este trabajo, conocimiento de la herramienta, capacidad computacional y filosofía de código libre.

Preprocesamiento y algoritmos:

- Python
  - Scikit-learn y scikit-optimize
  - Keras

    - Labrosa y ffmpeg
    - Imblearn

Visualización dinámica:

- HTML
- CSS
- Javascript

Visualización estática:

- Python
    - matplotlib
- R
    - ggplot2
    - dplyr

2.2.2 Ambiente de pruebas

Para el proceso de análisis descriptivo, generación de variables con técnicas *MIR*, así como la ejecución de algoritmos de *machine learning*, se configuró un ambiente *Anaconda*, que incluye librerías de codificación de audio, librerías *MIR*, y librerías de *machine learning*.

Para el entrenamiento de algoritmos de *machine learning* se hace uso de un procesador **Core i5, de 4 núcleos** físicos (8 lógicos) a **1.80 GHz, 8 GB de memoria RAM**, con sistema operativo Windows 10. Esta información es relevante para conocer las limitaciones de recursos físicos y su implicación en los tiempos de entrenamiento en la sección “Análisis y resultados”.

Debido a la cantidad de procesamiento requerido y el volumen de información que se requiere procesar para la **extracción de variables de los archivos de sonido** se hace uso de servicios de almacenamiento y procesamiento GCS (*Google Cloud Services*).

2.3 Preprocesado

A continuación, se describe detalladamente la composición de creación del set de datos final, en algunos casos se dividirá entre los preprocesados llevados a cabo para **algoritmos basados en técnicas *MIR*,** y el preprocesado para el entrenamiento de algoritmos **de *deep learning*.** En caso de omitir su relación con alguno de los ámbitos, es debido a que se trata de un proceso utilizado para ambos casos.

El set de datos consiste en un conjunto de archivos de audio que han sido extraídos a través del **API: Free Music Archive**, la cual permite la descarga de metadatos asociados y su archivo de audio.
El *dataset* utilizado contiene un archivo origen raws_tracks.csv, al cual añadimos **9,342 archivos de música nuevos** que contiene metadatos acerca de los archivos de música. Los

más relevantes para el análisis realizado son: Nombre de la pieza musical, URL del archivo, nombre del álbum, URL de álbum, nombre del artista, URL del artista, etc.

Con el archivo inicial se han creado nuevos sets de datos con información enriquecida, realizando consultas a *FMA API*, estos archivos han sido:

- **genre.csv**: Archivo que contiene información de los distintos géneros musicales utilizados, la estructura de dichos géneros es jerárquica por lo que se tiene información que permitirá crear árboles.
    - **genre_id**: id único del género musical
    - **parent**: id del género padre
    - **title**: nombre del género
    - **top_level**: id del padre más alto

- **tracks.csv**: Archivo que contiene información de cada pieza musical, que incluye los siguientes campos:
    - **tid**: identificador numérico único del archivo
    - **title**: título del archivo de música
    - **bit_rate**: la tasa de muestreo del archivo MP3
    - **date_created**: la fecha de creación de la pieza música
    - **duration**: la duración en segundos del archivo de música
    - **genre_top**: nombre del género principal del archivo de música
    - **genres_all**: nombre de todos los géneros musicales involucrados
    - **language_code**: código ISO del lenguaje de la letra de la canción

El set de datos de archivos de música inicial consiste en **106,574 archivos en formato MP3** con un **peso total de 879 GB** , que han sido organizado en carpetas de 3 dígitos, que van desde el 000, hasta los tres primeros dígitos del último identificador único por archivo (*tid*), de esta manera se cuenta con un máximo de 999 archivos por carpeta.

Toda esta información permitirá conocer información valiosa de cómo está compuesto el set de datos, así como detectar y corregir posibles sesgos. Sin embargo, el objetivo de este proyecto es el de crear un algoritmo cuyos datos de entrada se basen en técnicas de recuperación de información musical**,** por lo que estos datos no son parte del entrenamiento de modelos y se requiere de la creación de nuevas variables basadas en *MIR*.

2.4 Feature engineering

2.4.1 MIR - *Music Information Retrieval*

Por sus iniciales en inglés *Music Information Retrieval,* es una ciencia multidisciplinaria que recupera información de la música, regularmente se encuentra asociada a música en formato digital, donde a través de distintos procesos casi siempre asociados al procesamiento de señales y matemáticas en general, se obtienen patrones que representan las principales características de la música pasando por el ritmo, la armonía y la melodía.

Su uso está asociada a distintas disciplinas, que van desde el estudio de la música, la psicología, la musicología, el aprendizaje automático, etc., cuya aplicación puede ser la separación de pistas, reconocimiento de instrumentos, transcripción automática de música, creación de música, así como en sistemas de recomendación y categorización u organización de música.

Como parte de la creación de variables, se hará uso de técnicas de procesamiento de señales como lo son la Transformada de Fourier y el análisis de señales en el domino del tiempo.

Para cada uno de los siguientes métodos de recuperación de información, se hará uso de distintos **momentos matemáticos** y otras métricas de estadística univariantes, como lo son:

- La media (Momento ordinal 1)
- La varianza (Momento ordinal 2)
- La simetría (Momento ordinal 3)
- El exceso de curtosis (Momento ordinal 4)
- El mínimo
- El máximo

En algunos de los siguientes métodos de recuperación de información se tendrá más de un valor para cada momento estadístico, dependiendo de la forma de cálculo de cada método.

2.4.2 Transformada de Fourier

Es una transformación matemática empleada para transformar señales en el domino del tiempo a señales en el dominio de la frecuencia, una de sus principales bondades es que no solo permite la descomposición de una señal al dominio de la frecuencia, sino su reconstrucción al dominio del tiempo sin pérdida de información.

La transformada de Fourier es básicamente el espectro de frecuencias de una señal, y un ejemplo de su funcionamiento es el oído humano, el cual percibe una onda auditiva y la transforma en una descomposición de distintas frecuencia que es lo que escuchamos.

Su definición formal es:

$$\mathcal{F}^{-1}\{\hat{f}\} = f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{2\pi i \xi x}\, d\xi,$$

Fórmula 1 - Transformada de Fourier.

2.4.3 Espectrograma

El espectrograma es el resultado gráfico de calcular el espectro de frecuencias de una señal, por ejemplo, a través de la transformada de Fourier, donde está representado el tiempo en el eje *x*, la frecuencia en el eje *y*. La energía o potencia de un espectro se representa a través de colores que representan mayor o menor intensidad en el tiempo *t* y la frecuencia *f*.

Debido a la naturaleza digital de los datos, se utilizó una transformada de Fourier de Tiempo Reducido - STFT (por sus siglas en inglés *Short Time Fourier Transformation*), en la cual la información a ser transformada se divide en pedazos o tramas (que usualmente se traslapan unos con otros, para reducir irregularidades en las fronteras). Cada pedazo, una transformación de Fourier y el resultado complejo se agrega en una matriz, que almacena magnitud y fase para cada punto en tiempo y frecuencia.

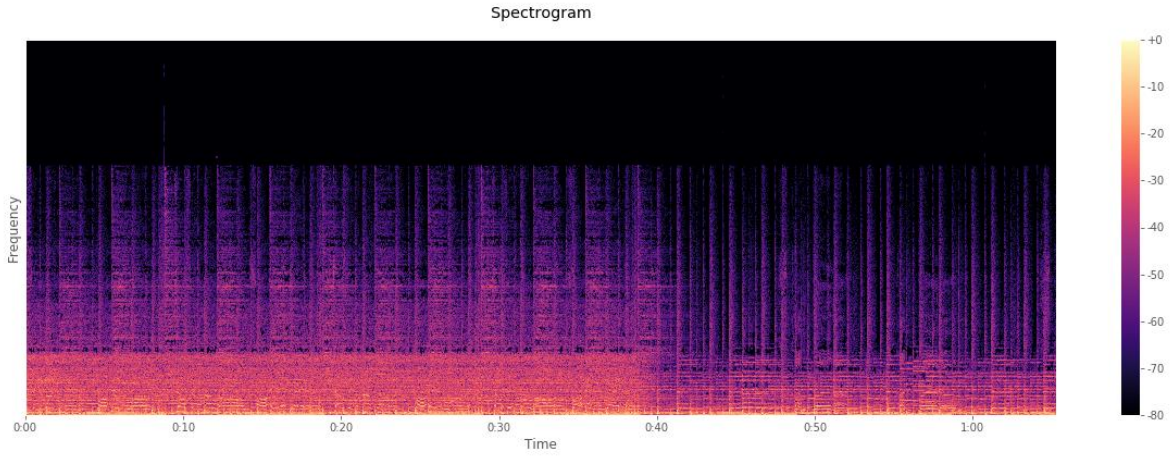


Imagen 2 - Espectrograma de una canción del género Rock.

2.4.4 *Mel scale* – Espectrograma

Uno de los principales problemas en la interpretación de un espectrograma, es que el dominio de frecuencias se representa de manera lineal, sin embargo, en el estudio de la audición suele no ser muy útil.

El oído humano suele percibir los cambios de frecuencia de una manera similar a un comportamiento logarítmico, por lo que suele ser utilizada la escala Mel como una alternativa

que tome en cuenta este comportamiento, permitiendo representar de una manera más sensible al contexto humano el dominio de frecuencias. Esto suele mejorar los resultados e interpretación del análisis de audio.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Fórmula 2 - Fórmula para convertir hertz (hz) en mels (m).

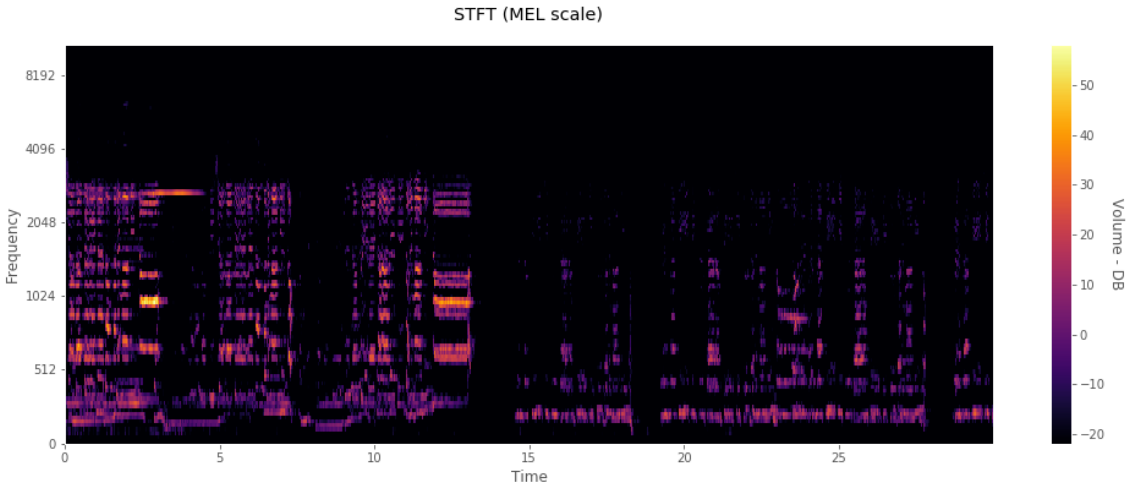


Imagen 3 - Escala Mel en un espectrograma.

2.4.5 Mel Frequency Cepstral Coefficients (MFCCs)

Estos coeficientes suelen utilizarse para la representación del habla basados en la percepción auditiva humana y el reconocimiento del timbre, se calculan con los siguientes pasos:

- Separar la señal en pequeños tramos (regularmente de 10 milisegundos).
- A cada tramo aplicar la Transformada de Fourier discreta y obtener la potencia espectral de la señal.
- Aplicar el banco de filtros correspondientes a la Escala Mel al espectro obtenido en el paso anterior y sumar las energías en cada uno de ellos.
- Tomar el logaritmo de todas las energías de cada frecuencia en escala Mel.
- Aplicarle la transformada de coseno discreta a estos logaritmos.

El número de coeficientes suelen definirse entre los 10 y 20. Para la exploración en el proyecto se utilizan 20 coeficientes, y por cada uno de ellos, se obtienen los momentos matemáticos antes descritos.

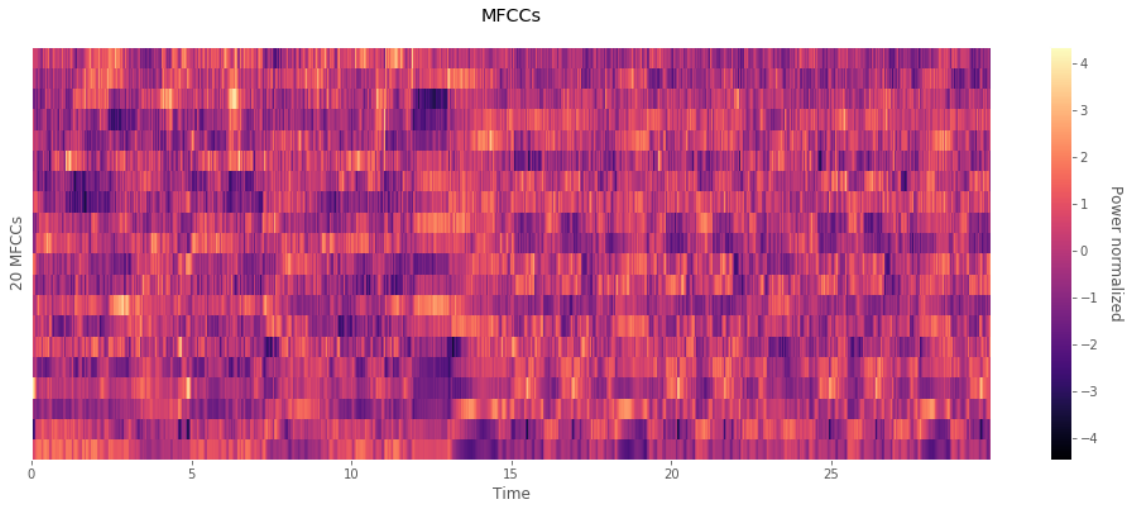


Imagen 4 - Representación de MFCCs.

Para su cálculo se hace uso de la función librosa.feature.mfcc().

2.4.6 Chroma - Energy Normalized (Cens)

Chroma es una forma de representación basado en armonía musical de temperamento igual, por lo que consiste de 12 bloques de frecuencias que representa a cada semitono en el contexto del tiempo y su potencia o energía. Para su cálculo se requiere de STFT, y un agrupamiento de frecuencias múltiplos exactos de la frecuencia fundamental de cada semitono.

Por cada ventana calculada por STFT se suma la energía de todas las frecuencias pertenecientes a cada semitono, por ejemplo, la frecuencia 440 Hz y 880 Hz representa la nota *la* central de un piano y la de la siguiente escala a la derecha, respectivamente. La energía de estas dos frecuencias es sumada y normalizada.

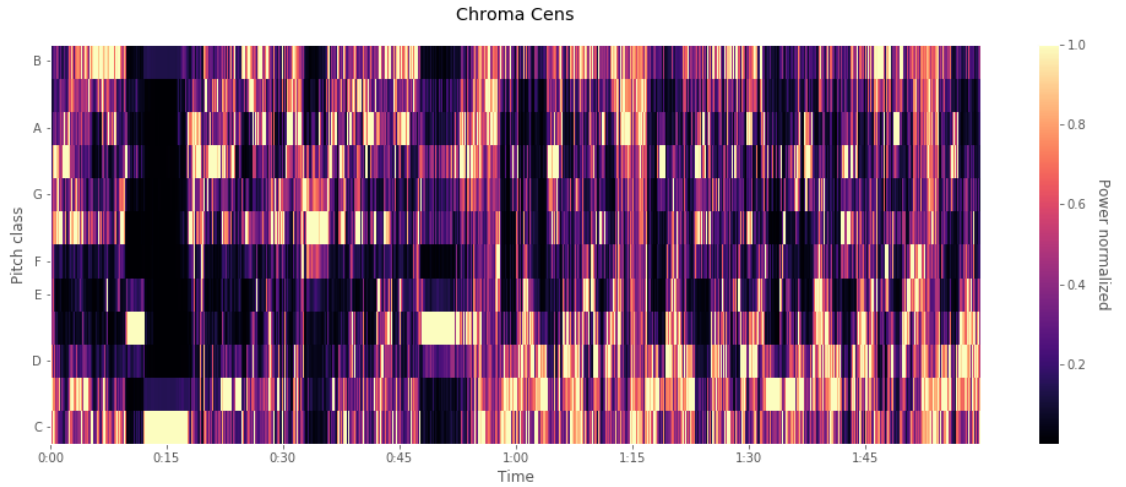


Imagen 5 - Chroma Cens de 12 semitonos.

Para su cálculo se hace uso de la función librosa.feature.chroma_cens().

2.4.7 Tonnetz

Es una forma de representación armónica basada en cercanía y octavas, suele utilizarse para el reconocimiento armónico de acordes musicales.

Para el cálculo se hace uso de 7 octavas, por lo que se obtendrán 7 vectores y sus momentos matemáticos antes mencionadas para cada uno de ellos.

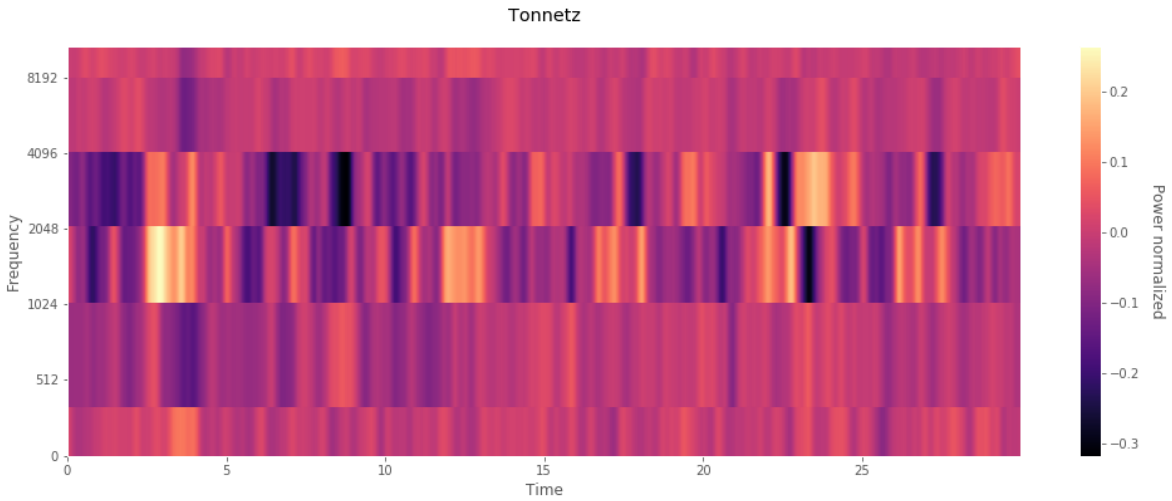


Imagen 6 - Representación de Tonnetz con escala de frecuencias lineal.

Para su cálculo se hace uso de la función librosa.feature.tonnetz().

2.4.8 Contraste Espectral

Se trata de una novedosa forma de recuperación de información musical que toma en consideración lo picos y valles del espectro y su diferencia en un número de bandas definida.

Para la exploración se hace uso de 6 bandas + 1, para las cuales se obtiene los momentos matemáticos antes descritos.

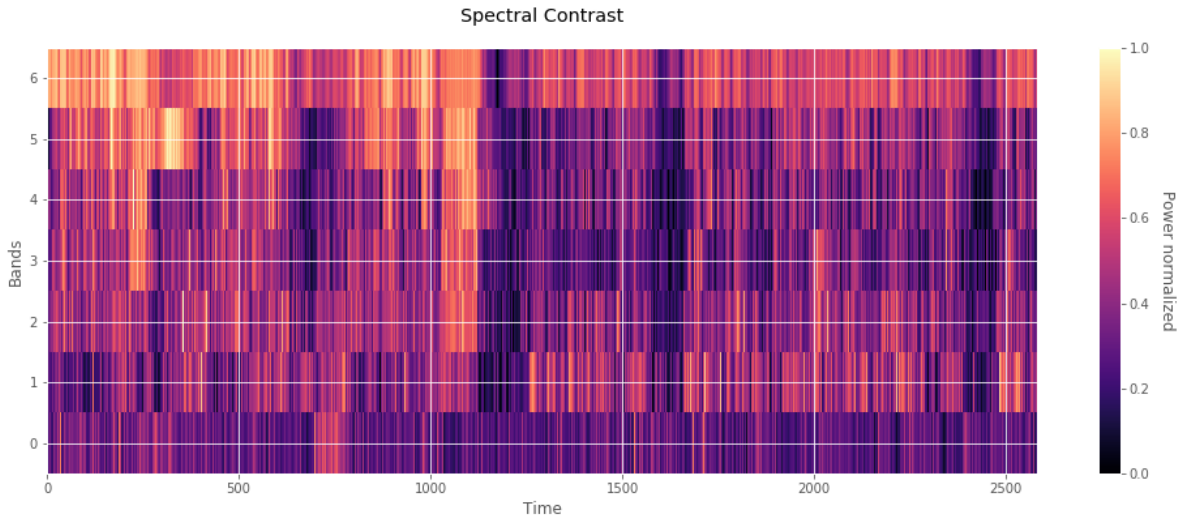


Imagen 7 - Representación de contraste espectral de 6 bandas.

Para su cálculo se hace uso de la función librosa.feature.spectral_contrast().

2.4.9 Zero Crossing Rate - ZCR

A diferencia de los métodos anteriormente descritos, este método y los consecuentes se basan en el procesamiento de señales en el dominio del tiempo. Cabe aclarar que para la representación gráfica de estos métodos de recuperación de información musical el eje x representa el dominio del tiempo, y el eje *y* la potencia o energía de la señal.

Si el audio en cuestión es monofónico, entonces por encima y por debajo del valor cero del eje *y* se visualiza una réplica exacta de la señal, por el contrario, si se trata de un audio estéreo, entonces, por debajo del cero se visualiza la señal correspondiente al auricular izquierdo, y por encima del cero correspondiente al auricular derecho.

Como su nombre lo indica, representa una tasa del número de veces que una señal cruza por el eje *y* en cero.

A diferencia de las técnicas anteriores, para este método y para todos los que a continuación se describen, sólo se obtendrá un valor asociado a cada momento matemático.

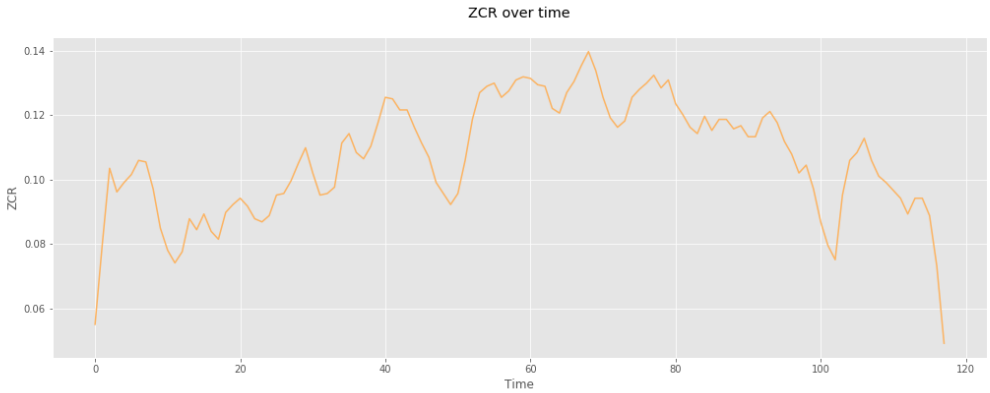


Imagen 8 - ZCR correspondiente a 120 ms.

Para su cálculo se hace uso de la función librosa.feature.zero_crossing_rate().

2.4.10 RMSE – Energía

Por sus siglas en inglés *Root Mean Square*, es la suma de las medias aritméticas de un lapso elevando al cuadrado y dividido por la raíz cuadrática, asociado al nivel de ruido de una señal.

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_N^2}{N}}$$

Fórmula 3 – RMSE.

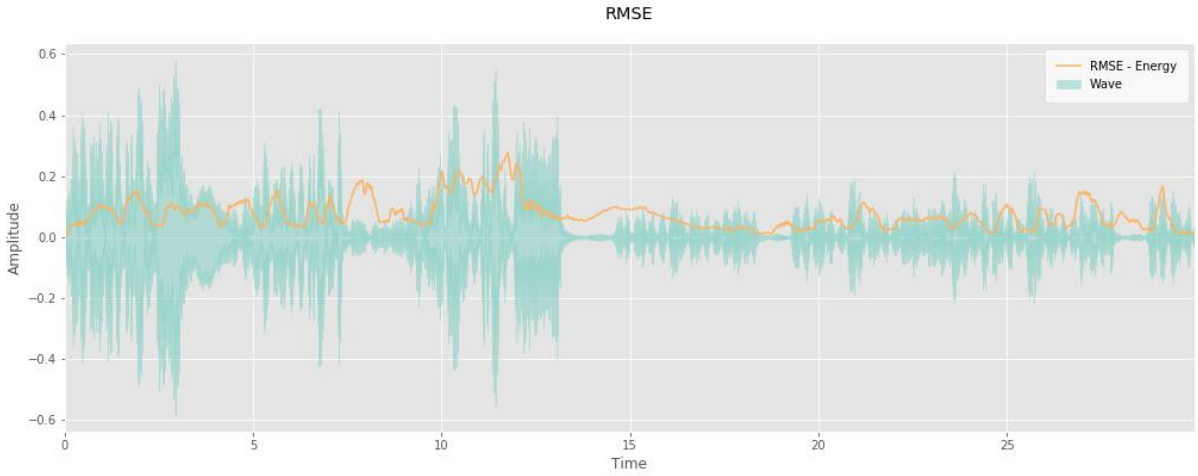


Imagen 9 - RMSE de una señal.

Para su cálculo se hace uso de la función librosa.feature.rmse().

2.4.11 Centroide Espectral

Indica en qué frecuencia, la energía del espectro se encuentra centrada, es parecido al cálculo de una media ponderada, y se calcula por cada ventana de un STFT.

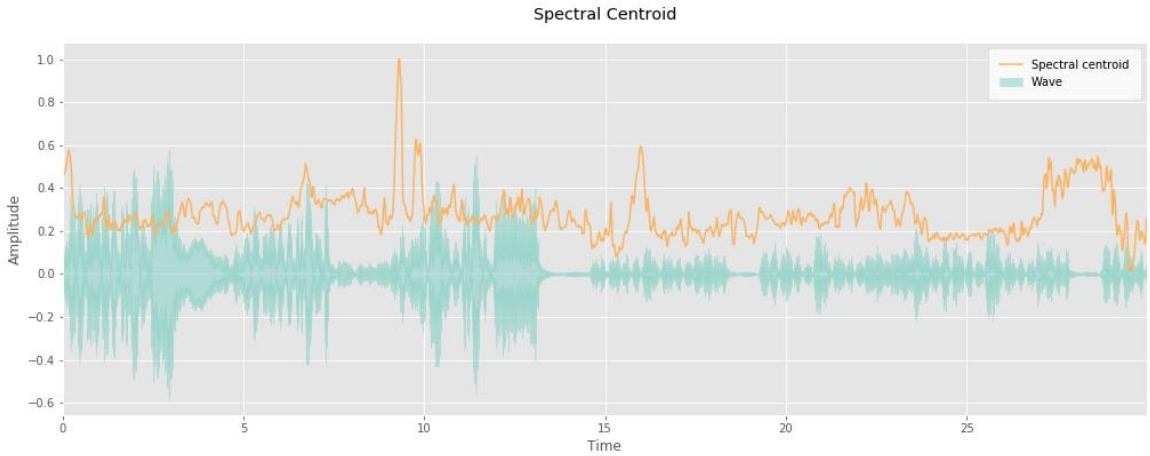


Imagen 10 - Spectral Centroid.

Para su cálculo se hace uso de la función librosa.feature.spectral_centroid().

2.4.12 Ancho de Banda Espectral

Se calcula un orden *P* del ancho de banda. Para la generación de las variables se hace uso del orden *P*=2.

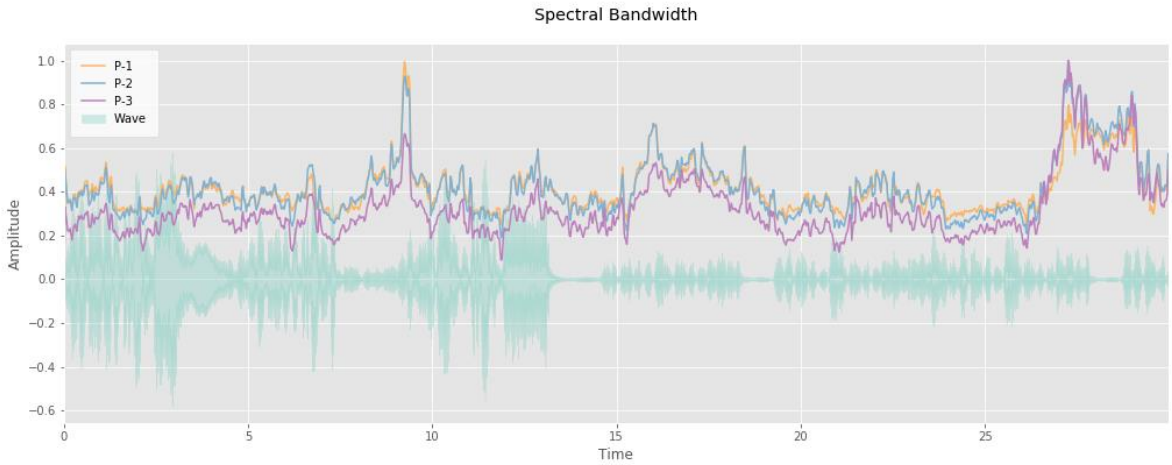


Imagen 11 - Spectral Bandwidth

Para su cálculo se hace uso de la función librosa.feature.spectral_bandwidth().

2.4.13 Reducción Espectral

La reducción espectral es la frecuencia por debajo de la cual un porcentaje específico de la energía espectral total cae, por ejemplo, por debajo del 85%.

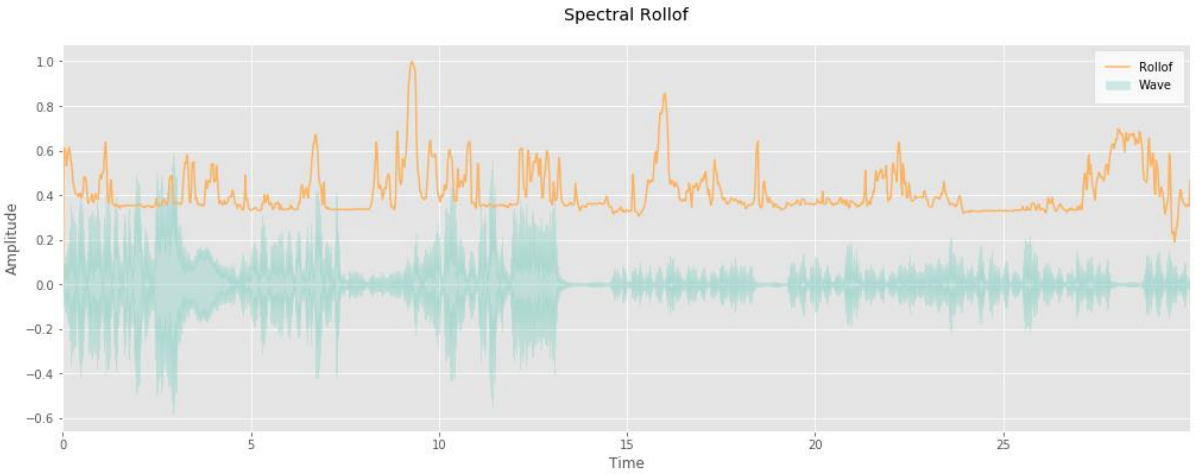


Imagen 12 - Spectral Roll-off

Para su cálculo se hace uso de la función librosa.feature.spectral_rolloff().

2.4.14 Variables

El resultado final del uso de estas técnicas es de **518 variables construidas**, asociados a 11 métodos distintos de recuperación de información musical, con el cálculo de 4 momentos estadísticos y 2 métricas estadísticas.

2.5 Limpieza y tratamiento de *missing values*

Parte de los metadatos obtenidos contiene una cantidad considerable de *missing values*. El análisis se centra en recuperación de información musical y su uso para organización a través del género musical, por lo que es muy importante tratar el considerable número de *missing values* del campo **genre_top de casi el 54%** de los archivos de audio.

File	*Feature*	*% NAN*
Artist	active_year_begin	78.69
	active_year_end	94.95
	associated_labels	86.60
	bio	33.23
	date_created	0.80
	latitude	58.20

Track	location	34.12
	longitude	58.20
	members	56.04
	related_projects	87.65
	website	25.63
	wikipedia_page	94.76
	composer	96.55
	date_recorded	94.22
	genre_top	53.40
	information	97.79
	language_code	85.90
	license	0.08
	lyricist	99.70
	publisher	98.81
	title	0.00

Tabla 2 - Missing values en metadatos.

Para su tratamiento se hace uso el campo **all_genres** el cual contiene todos los géneros asociados a cada archivo de música.

Existen **2,231 registros con el campo all_genres vacío**, por lo que en este caso han sido eliminados y descartados del análisis.

Para el resto, se ha construido una función recursiva con la que se obtiene el padre de todos los géneros asociados y regresando aquel que se encuentre la mayor cantidad de veces, sí existe más de un padre con el mismo número de apariciones, se elige uno de manera aleatoria.

Para las variables obtenidas a través de *MIR*, no se cuenta con *missing values*, y además se aplica un análisis de correlación en búsqueda de correlación perfectas, sin encontrar ningún caso.

El set de datos final después de la limpieza es de **104,343 archivos de música.**

2.6 Análisis exploratorio

Después de la limpieza de *missing values,* se ha realizado un análisis exploratorio con los metadatos en búsqueda de patrones que permitan identificar sesgos. En el caso de las variables creadas, se buscan patrones a través de análisis univariante y multivariante.

El análisis está compuesto por la descripción de los metadatos obtenidos a través de la *API FMA*, los aspectos técnicos relevantes asociados a los archivos de música, así como un análisis de las variables construidas a través de técnicas *MIR*.

2.6.1 Metadatos

2.6.1.1 Tamaño del set de datos

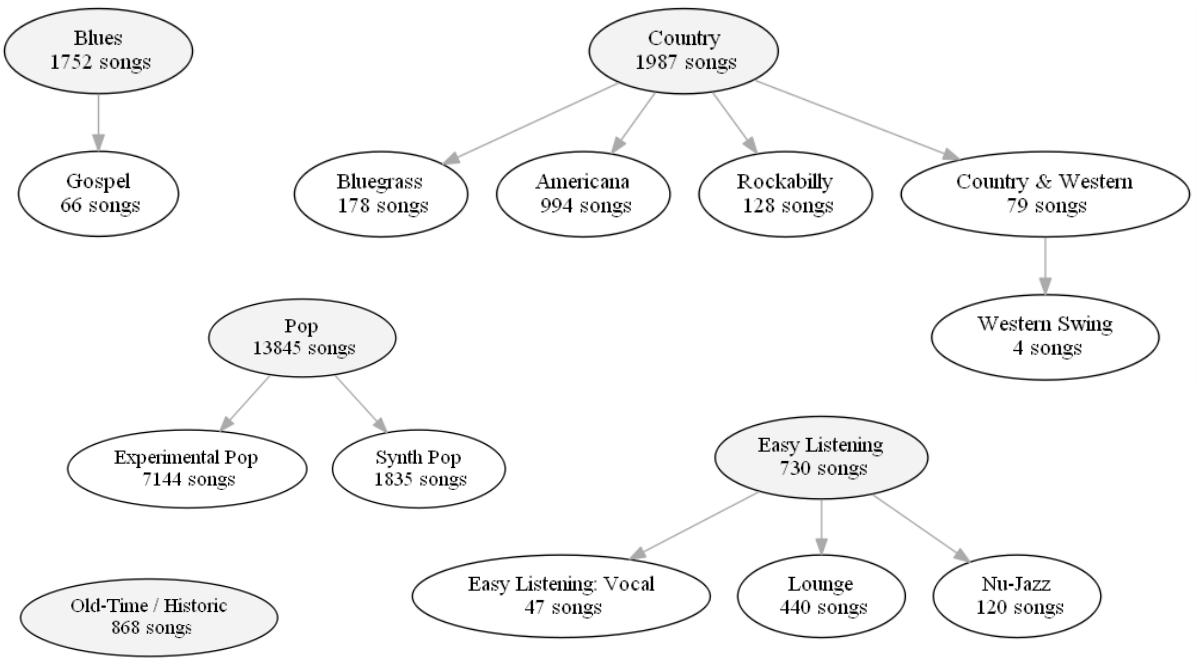
332.22 days of music (Media of 275 seconds per file).

Imagen 13 - Tamaño de componentes del set de datos.

2.6.1.2 Géneros musicales

Los géneros musicales del set de datos se encuentran organizados como estructuras de árbol, contando con 16 principales géneros de música, y con un total de 161 géneros.

Los géneros se organizan en estructura de árbol, conociendo cuál es su género padre:



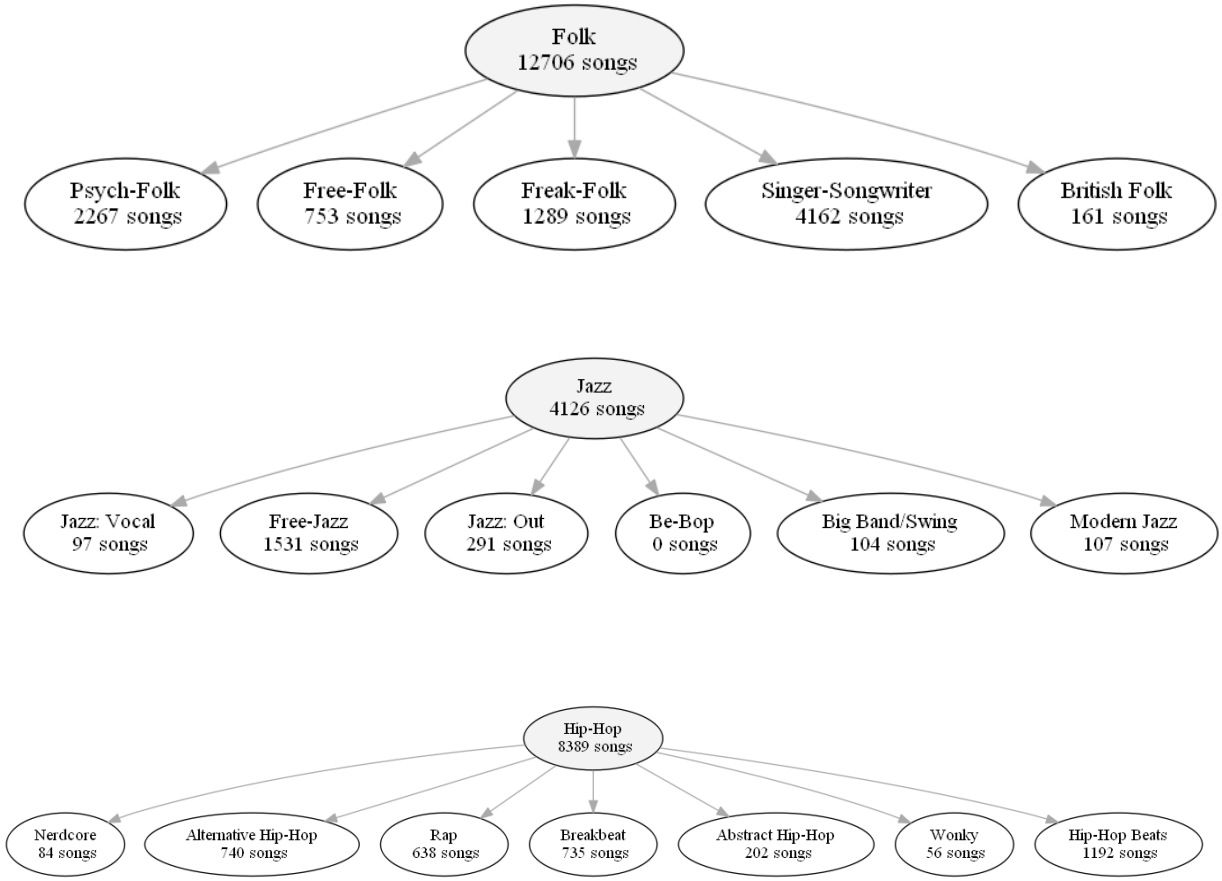


Imagen 14 - Árboles de 8 de los 16 géneros musicales padre.

Se cuenta con un set de datos muy desbalanceado, con 3 géneros musicales predominantes: experimental, electrónica y rock.

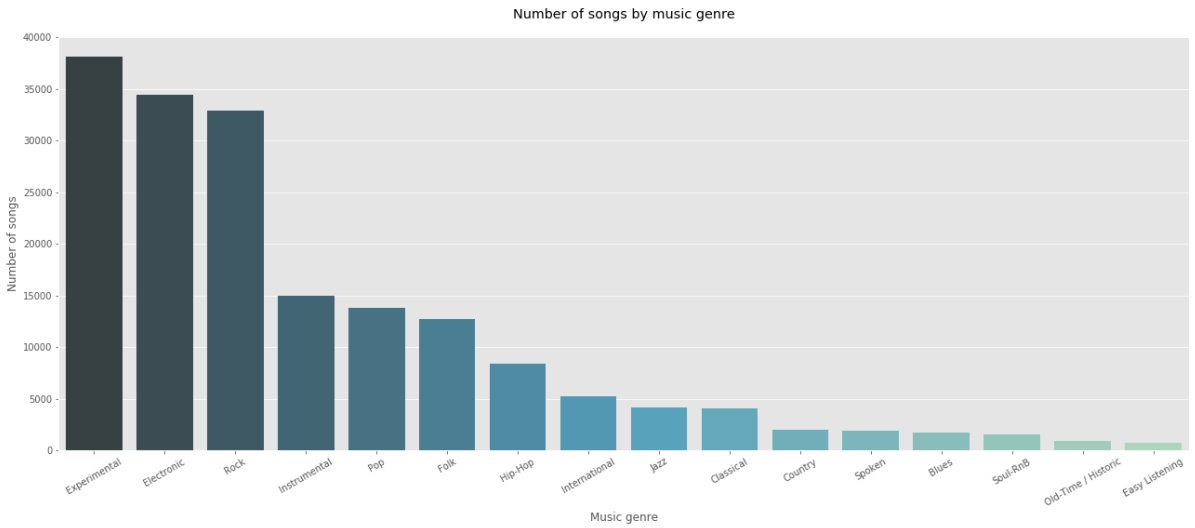


Imagen 15 - Número de archivos de música por género musical padre.

El desbalance se observa de igual manera tomando en cuenta los 161 géneros musicales.

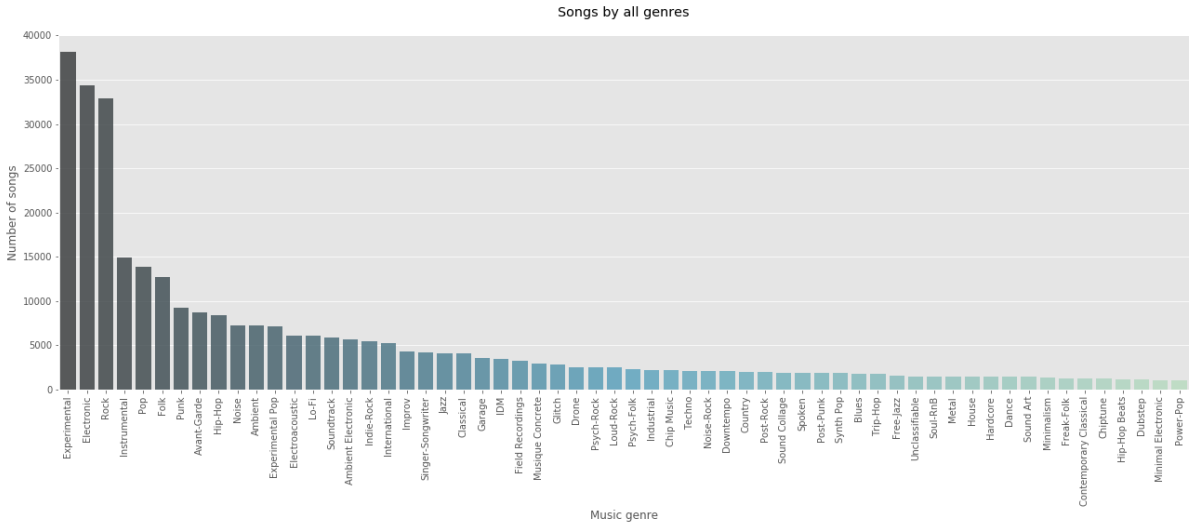


Imagen 16 - Número de archivos de música por género musical con más de 1,000 archivos.

En cuanto al número de artistas por género musical, se observa una mayor diversidad de los artistas de música experimental, rock y electrónica.

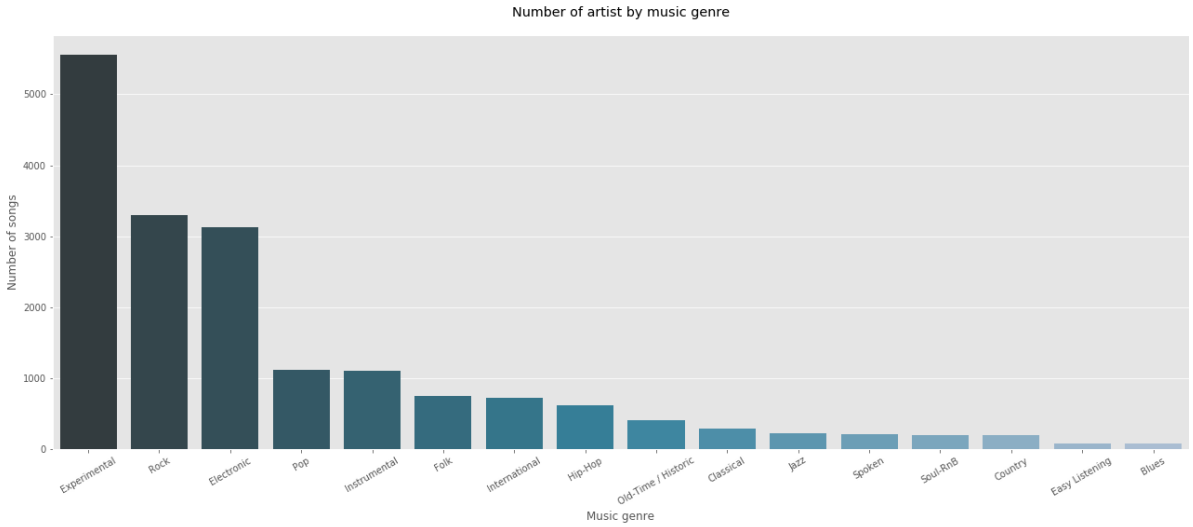


Imagen 17 - Número de artistas por género musical principal.

La distribución de géneros por cada archivo de música, podría considerarse como una distribución asimétrica a la derecha, 25% de los archivos de música tienen 3 géneros de música asociados.

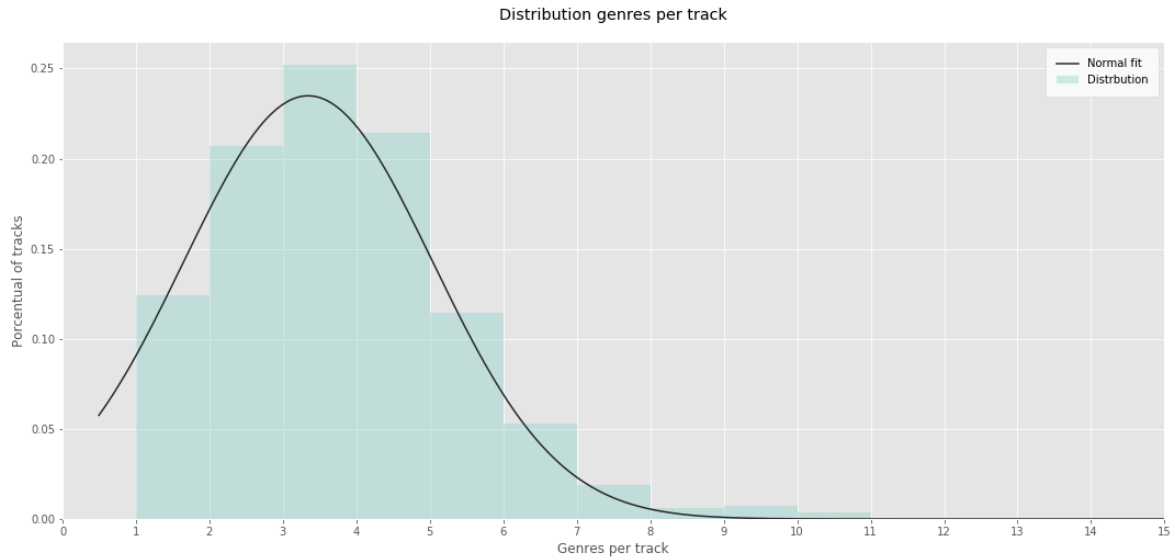


Imagen 18 - Distribución de número de género por canción.

Con el objetivo de encontrar una correlación entre los distintos 161 géneros musicales, se realizó un conteo cruzado escalado de 0 a 1, del número de veces que un género se encuentra presente al mismo tiempo en el set de datos.

Se puede observar:

- Géneros como el pop, el rock, el instrumental y el experimental se encuentran presentes en archivos de música con casi todos los géneros de el set de datos.

- La matriz tiene una clara carga en las filas y columnas superiores y a la izquierda debido a que la mayoría de estos son los géneros musicales padres, lo cual aumenta la probabilidad de que estos géneros se encuentren cruzados con sus géneros hijos.

- Los géneros tradicionales y asiáticos suelen no estar presente en otro tipo de géneros musicales.

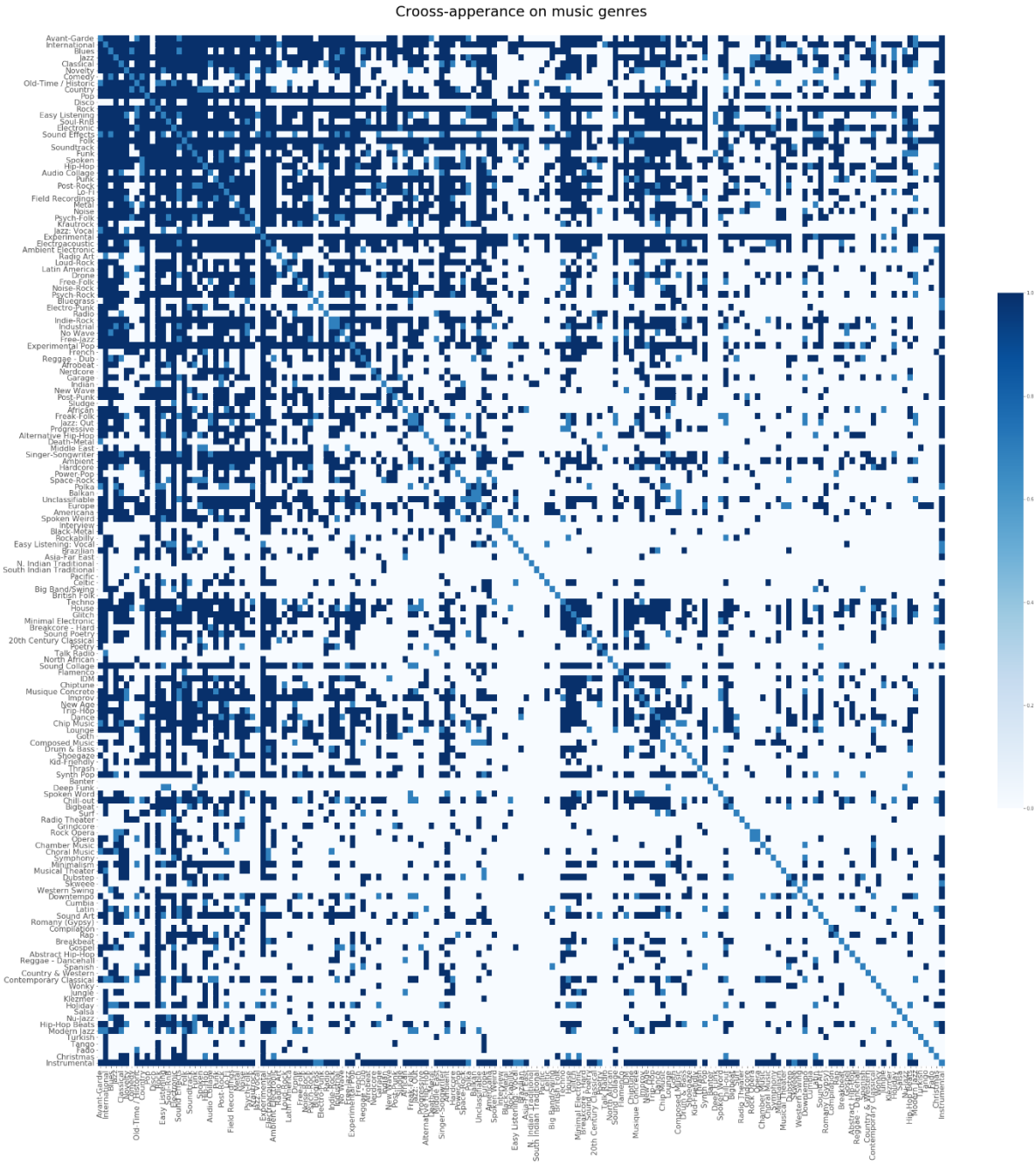


Imagen 19 - Matriz de aparición cruzada de géneros musicales.

2.6.1.3 Canciones por idioma

Solo se cuenta con 15.18% de las canciones con este metadato, de las cuales más del 96% pertenecen al idioma inglés.

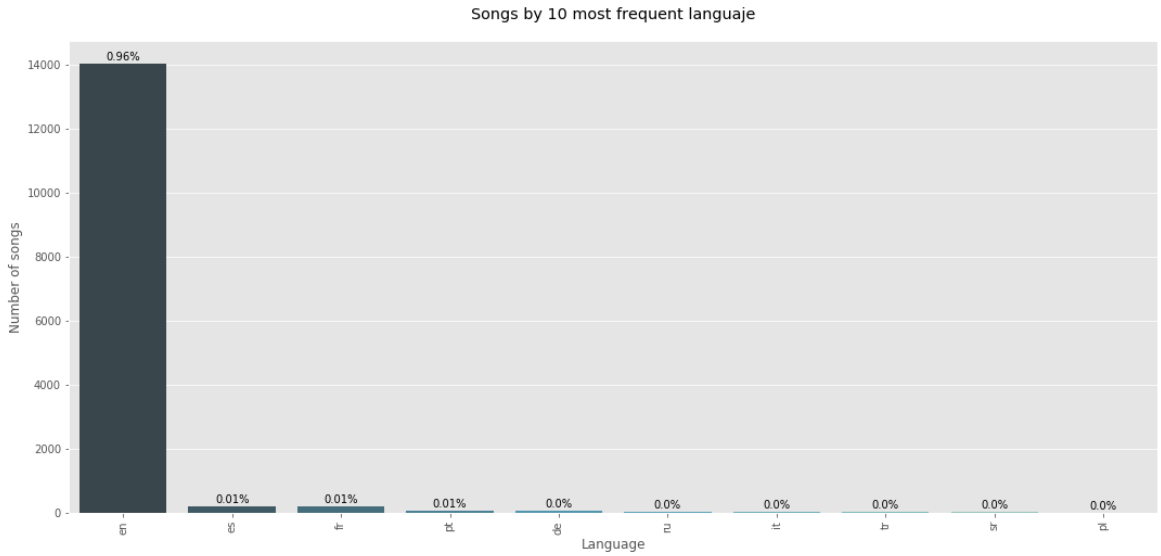


Imagen 20 - Porcentaje de canciones de los 10 idiomas más frecuentes.

2.6.1.4 País de origen de los artistas

Solo se cuenta con información del origen de los artistas de un **24%** de los archivos de música**,** se puede observar un claro sesgo del origen geográfico de los artistas incluidos en el set de datos, teniendo un sesgo de origen de América del norte y Europa (se debe considerar que la industria de música comercial es la fuente de dicho sesgo), además, una considerable cantidad de artistas del género musical rock provienen de Estados Unidos, y una considerable cantidad de artistas de música experimental de Europa del este.

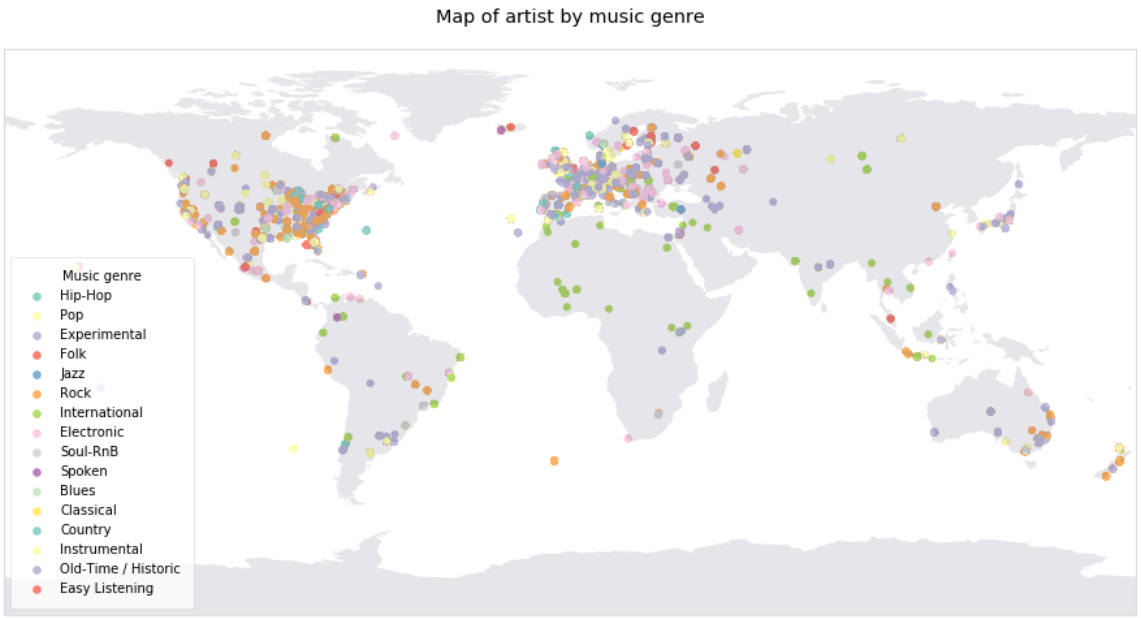


Imagen 21 - Mapa de origen de los artistas por género musical.

2.6.1.5 Fechas de lanzamiento

El set de datos contiene lanzamientos desde el año 1896 hasta el año 2017, con una clara tendencia de crecimiento desde el año 1995.

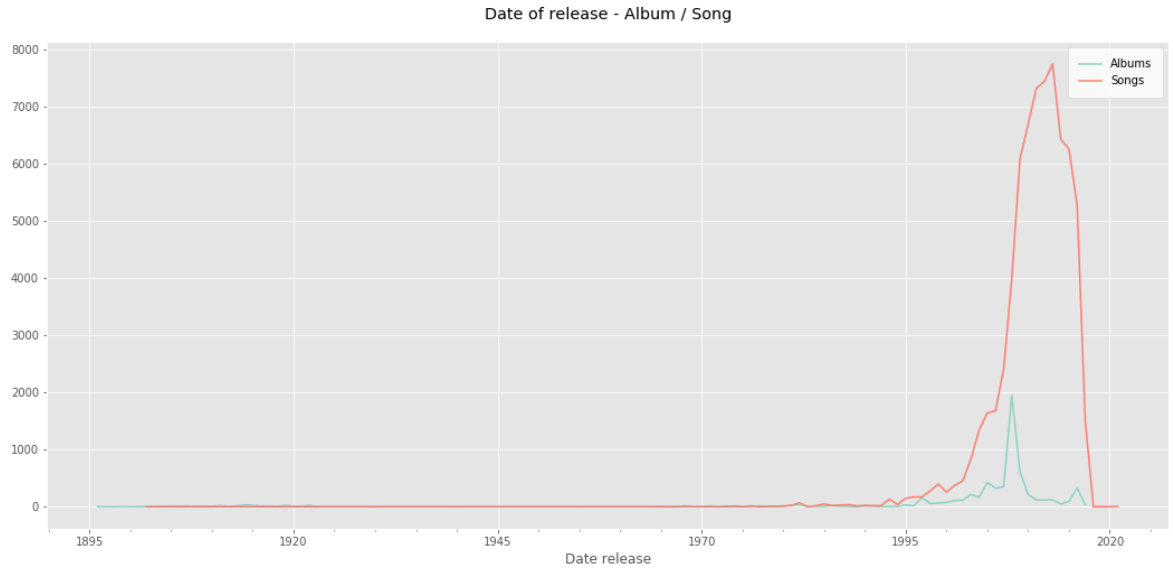


Imagen 22 - Fecha de lanzamiento de álbumes y archivos de música.

2.6.2 Técnicos

Es muy importante conocer algunos aspectos técnicos que permitan tomar decisiones de cómo se tratarán los archivos antes de la generación de variables.

2.6.2.1 Duración de archivos de música

Se observa una distribución asimétrica a la derecha, leptocúrtica.

El archivo de música con **mayor duración es de 5 horas**, cuenta con 16 archivos de audio con duración igual a 0, los cuales fueron verificados y recalculados. Además, se cuenta con 1,986 archivos con tamaño menor a 30 segundos.

La media de tiempo del set de datos es de **241.16 segundos** y una mediana de **213.00 segundos.**

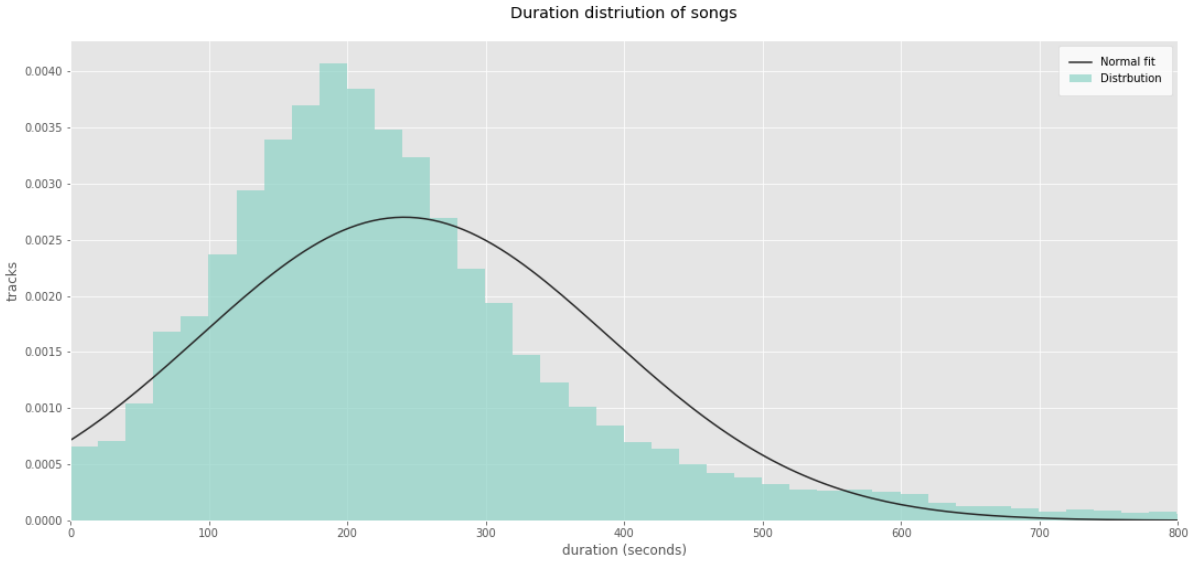


Imagen 23 - Distribución de duración de archivos de música.

2.6.2.2 Tasa de muestreo

La tasa de bits o tasa de muestreo, es el número de bits por segundo que se guardan como muestra al convertir una señal analógica a digital.

Debido a que la creación de variables dependerá de este valor, es importante conocer la forma en la que se distribuyen. Regularmente es un estándar en música muestrear con 128, 248 o 320 Kbps, por lo que se puede observar son algunos de las tasas que se repiten con mayor frecuencia. Más de la mitad del set de datos tiene una excelente calidad.

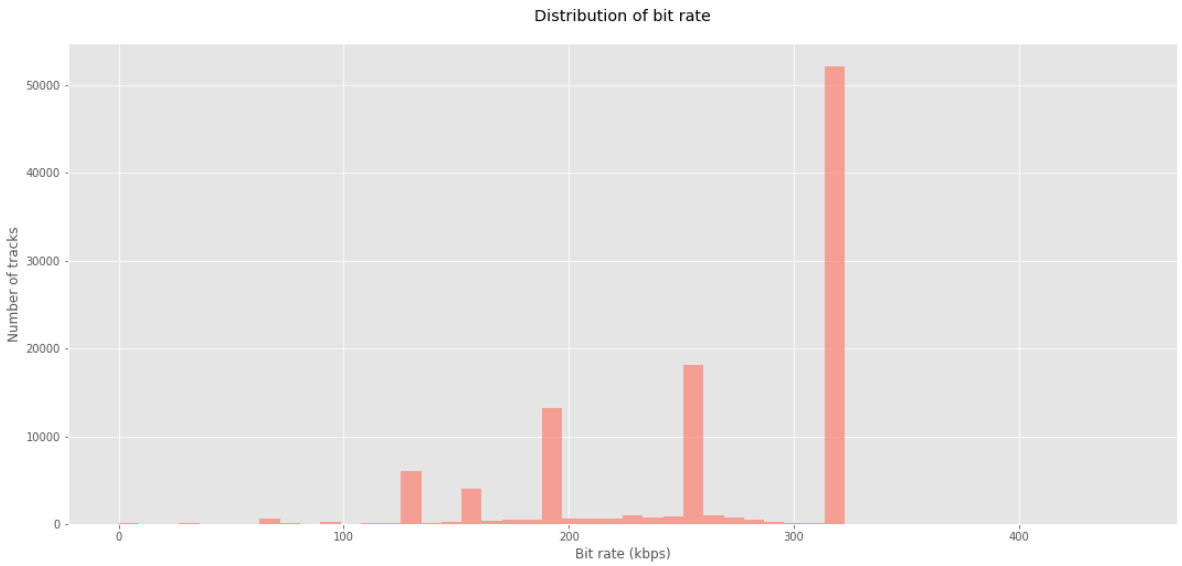


Imagen 24 - Distribución de tasa de bits.

2.6.3 Variables

Una vez creado y definido el set de variables basadas en *MIR*, se realizó un análisis univariante y multivariante del set de datos.

Del análisis de correlaciones se puede destacar:

- No existen correlaciones perfectas entre las variables.
- La correlación más alta es de 0.99, sin embargo, se observa poca correlación entre las variables.
- 90% de las variables tienen correlaciones por debajo del 0.50 en valor absoluto.

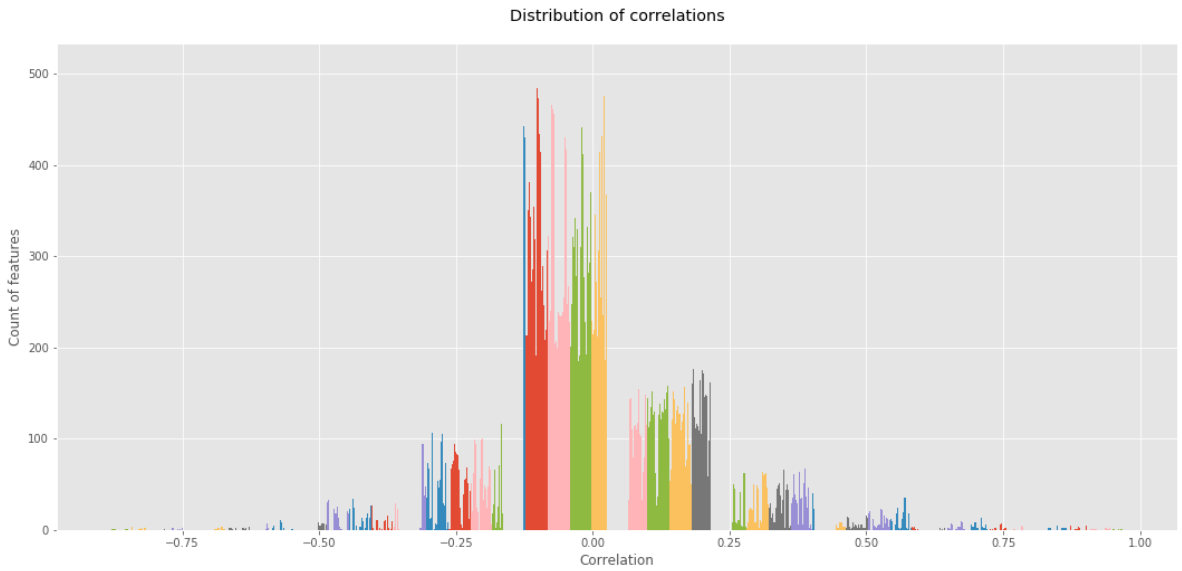


Imagen 25 - Distribución de correlaciones.

Debido a la alta dimensión del set de datos solo se realizó el análisis de distribución de algunas variables creadas y su relación lineal con valores del mismo método de recuperación de información musical.

Como ejemplo, se tomó el primer momento estadístico ordinal (media), de 6 de los 20 valores de MFCC para analizar su distribución y su relación lineal entre ellos. Se puede observar muy poca relación lineal entre ellas, lo cual podría indicar que el conjunto de variables de este método puede describir una diversidad existente entre los distintos coeficiente MFCC obtenidos.

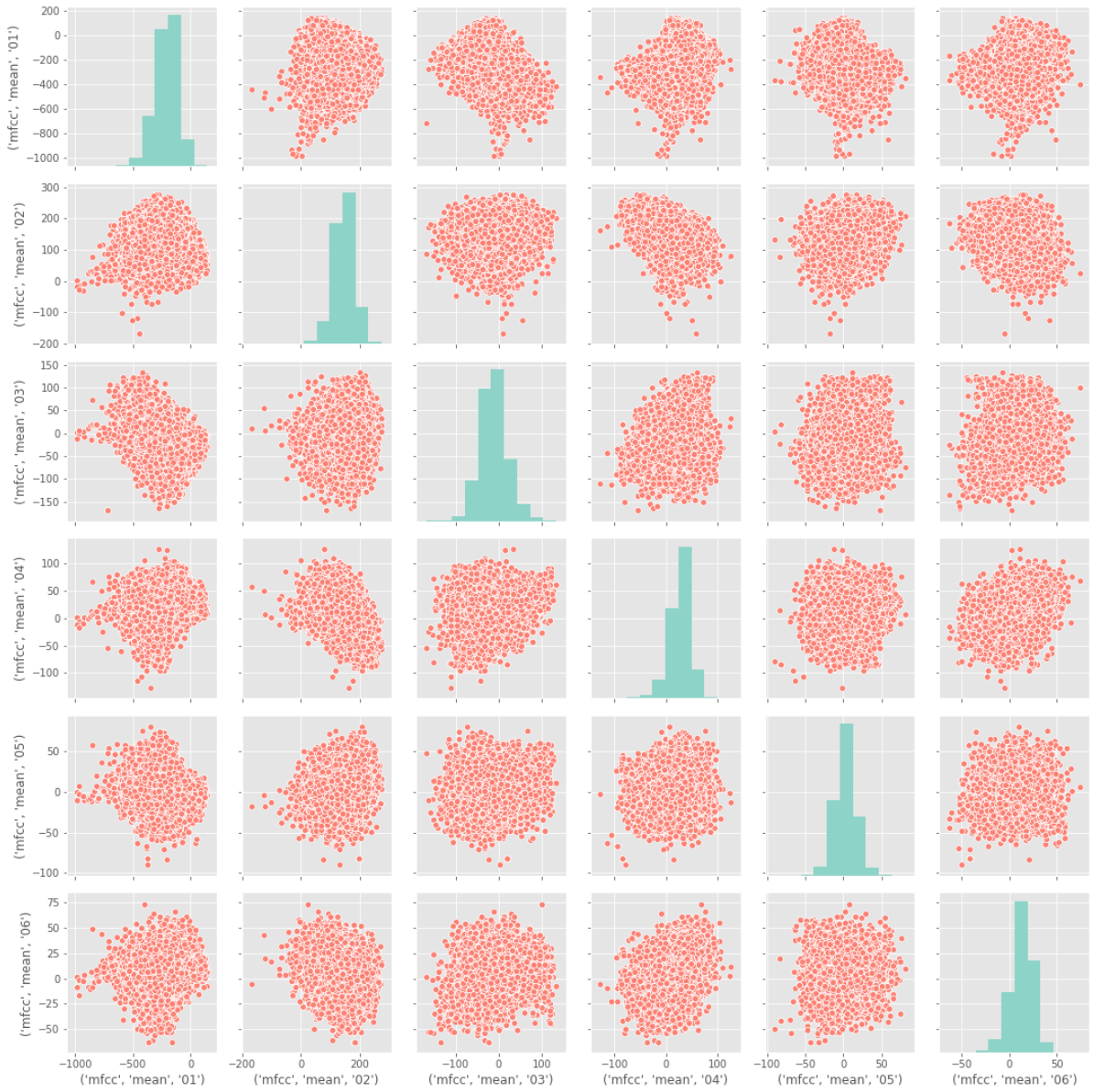


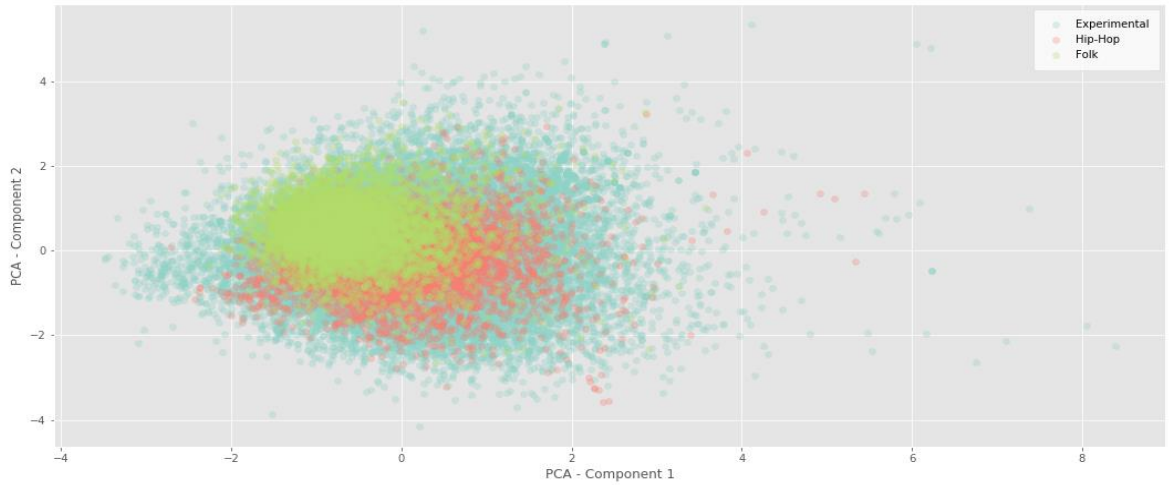
Imagen 26 - Relación lineal y distribución de media - MFCC.

Debido a la alta dimensión de nuestras variables, y con el objetivo de poder visualizar su posible aportación real en la clasificación de géneros musicales, se ha aplicado un algoritmo de análisis de componentes principales sobre las variables, obteniendo las primeras dos componentes con el objetivo de visualizar, por cada método de *MIR*, su posible aporte a la predicción del género musical.

Para su representación, se han elegido las dos primeras componentes principales y 3 géneros musicales que permiten la más clara visualización del aporte de cada método a la separación de clases, estos géneros musicales son: experimental, hip-hop y folk.

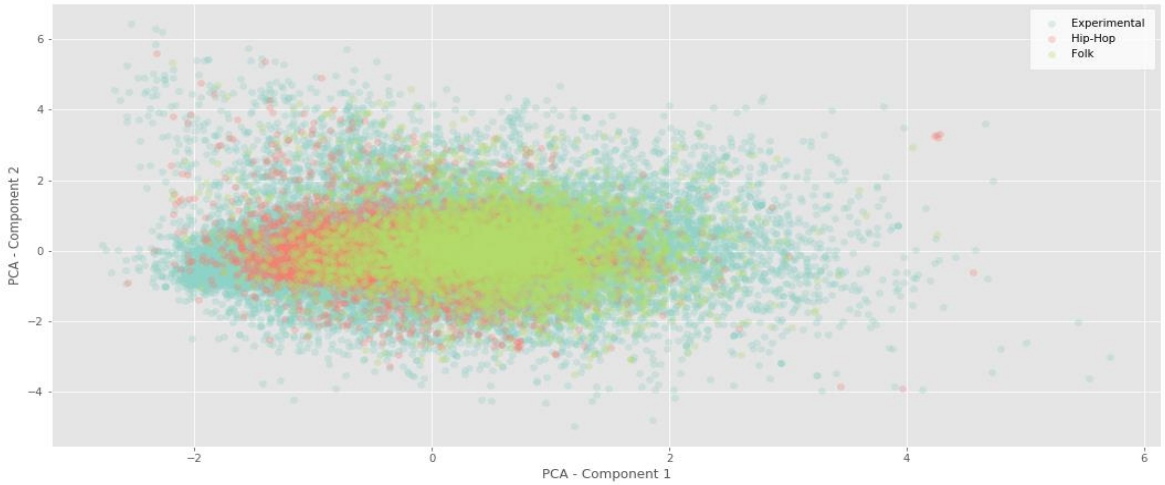
Con el análisis se puede observar que MFCC y contraste espectral, son los dos métodos que parecen aportar más a la separación de géneros musicales, mientras que Tonnetz es el que aporta menos.

PCA: mfcc

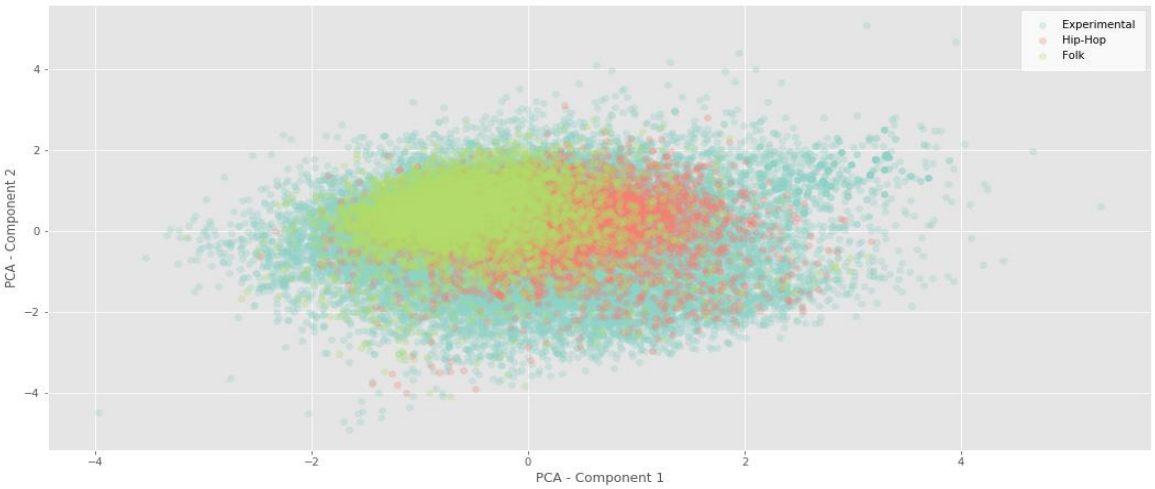


.

PCA: spectral_bandwidth



PCA: spectral_contrast



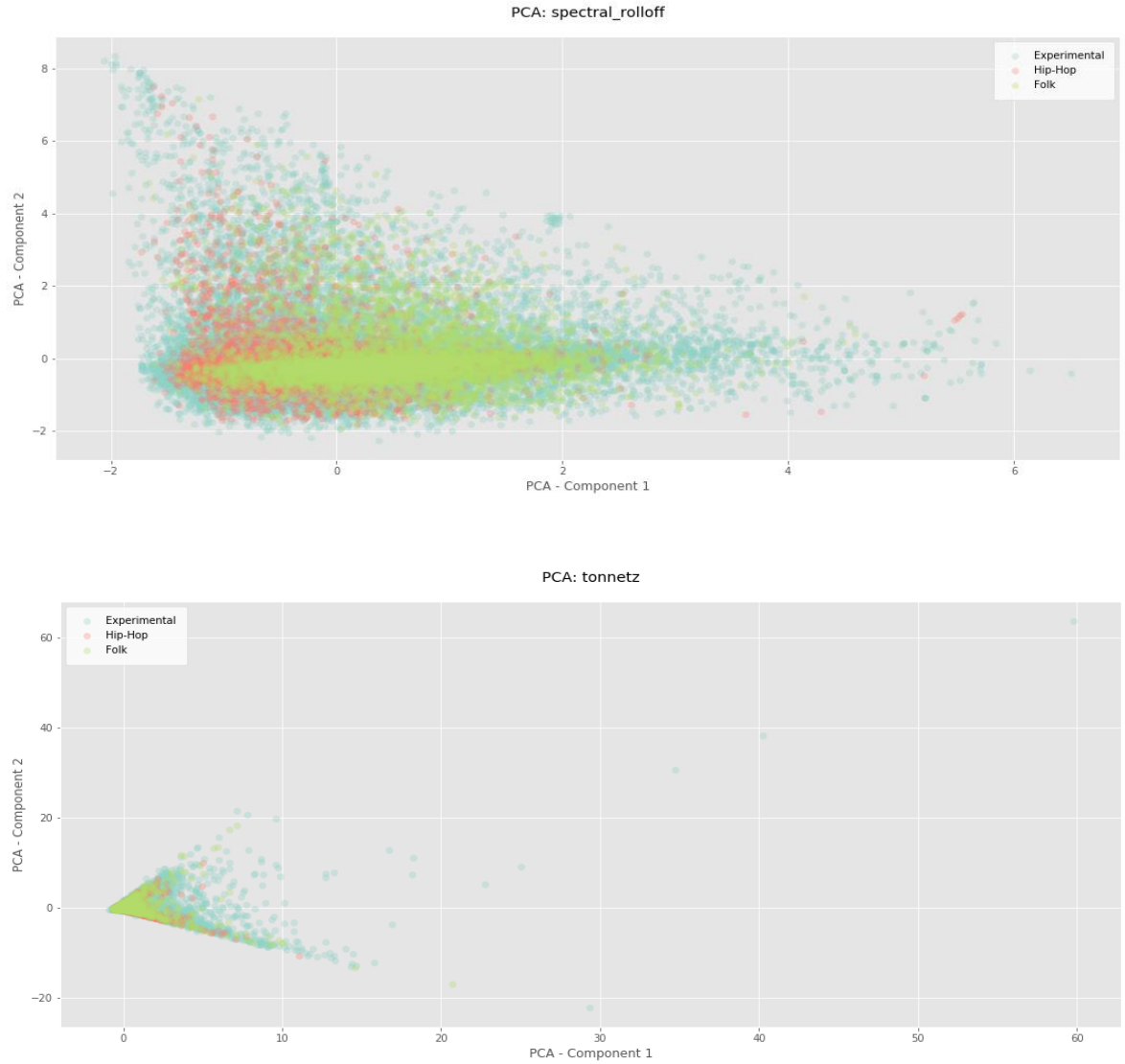


Imagen 27 - PCA de variables MIR proyectadas en 3 géneros musicales.

2.7 Selección de variables

Debido a la alta dimensionalidad de las variables obtenidas, y a los recursos físicos limitados, no se exploró la opción de selección de variables por métodos tipo *wrapper,* debido al tiempo de ejecución excesivo de algoritmos stepwise, *forward*, o *backward* con dimensiones tan altas.

Para la selección de variables se exploraron 3 distintas estrategias:

2.7.1 PCA - Reducción de dimensionalidad

Se aplicó un análisis de componentes principales - PCA (por sus siglas en inglés *Principal components analysis*), y se utilizó la gráfica de codo para seleccionar el número de componentes de acuerdo a la inflexión que se visualiza como la que acumula la mayor cantidad de varianza.

Debido a la existencia de dos quiebres se realizaron pruebas con las **primeras 4 componentes**, ya que la primera componente explica un 86% de la varianza, y la suma de las primeras 4 componentes cerca del 95%.

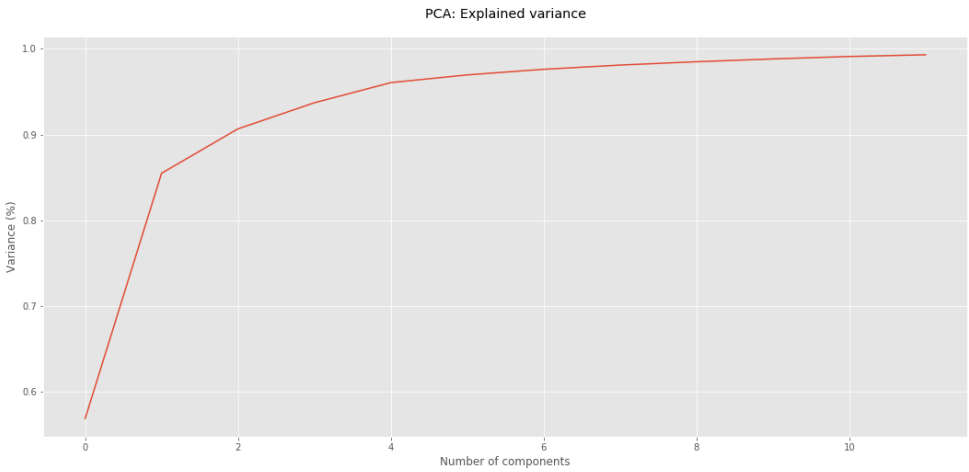


Imagen 28 - Gráfica de codo (Varianza explicada por componente principal).

2.7.2 MDA - *Random Forest*

Mean Decrease in Accuracy (MDA), es un método que fue seleccionado tras una búsqueda de métodos eficientes para alta dimensionalidad que consiste en realizar *n* ejecuciones del algoritmo, tomando una variable de forma aleatoria, y permutando sus valores aleatoriamente. Se calcula la diferencia causada en el *accuracy* debido a la permutación, y al final de obtiene una media que permite calcular el peso de importancia de una variable.

Se hace uso del algoritmo *Random Forest* debido a:

- Comprobada bondad de ajuste en problemas de los que se desconoce o es complejo conocer el valor de las variables en el negocio.
- Su fácil paralización.
- Su tiempo final de ejecución.

Para facilitar la selección de variables, se tomaron aquellas cuyo peso fuera mayor a 0.001, finalizando con 40 variables, las cuales son:

Variable	*Weight*
mfcc_max_04	0.005383
mfcc_max_01	0.002876
mfcc_mean_03	0.002526
mfcc_mean_04	0.002341
mfcc_max_03	0.002305
mfcc_std_14	0.002286
spectral_contrast_median_04	0.002194
spectral_contrast_mean_02	0.002157
spectral_contrast_mean_03	0.002102
spectral_contrast_mean_04	0.002065
rmse_median_01	0.001936
mfcc_mean_01	0.001936
spectral_contrast_median_02	0.001881
spectral_contrast_skew_03	0.001881
mfcc_median_06	0.001862
mfcc_std_16	0.001825
mfcc_median_20	0.001715
mfcc_std_12	0.001641
mfcc_mean_06	0.001622
spectral_rolloff_skew_01	0.001567
mfcc_std_10	0.001549
mfcc_max_07	0.001530
mfcc_std_20	0.001530
mfcc_std_02	0.001493
mfcc_std_06	0.001493
tonnetz_std_06	0.001401
tonnetz_std_03	0.001383
mfcc_std_04	0.001309
mfcc_median_03	0.001272
spectral_centroid_mean_01	0.001254
spectral_contrast_skew_01	0.001254
mfcc_skew_07	0.001235
mfcc_std_15	0.001217
chroma_cens_max_12	0.001180
spectral_contrast_max_07	0.001162
mfcc_std_09	0.001143
mfcc_median_18	0.001143
spectral_centroid_skew_01	0.001051
mfcc_median_01	0.001032
spectral_contrast_min_02	0.001014

Tabla 3 - Pesos de variables en MDA - Random Forest.

2.7.3 Combinatoria de técnicas *MIR*

Para la obtención de las variables más significativas, se trató cada técnica *MIR* como un conjunto de variables atómicas, probando distintos métodos de clasificación sobre cada set de variables de cada técnica *MIR* de forma independiente.

Con aquellos métodos que mostraron un mejor resultado, se realizaron distintas combinatorias en búsqueda de la mejor combinación, simulando un método *wrapper* de selección de variables, los resultados serán detallados más adelante en este documento.

2.8 Selección de métricas

Para la selección de la métrica utilizada para los modelos, se han tomada las siguientes consideraciones:

- El set de datos está desbalanceado, la clase minoritaria representa apenas el 0.4% del total de archivos de música, mientras que la clase mayoritaria tiene el 21% del total de archivos.
- Se trata de un problema de clasificación multiclase, donde no se tiene prioridad por ninguna de las clases de forma teórica, pero debería de analizarse el peso positivo de clasificar de forma adecuado una canción de la clase mayoritaria al ser alcanzable por un mayor número de usuarios.
- El desbalance reflejado en el set de datos existe como un caso real en la industria, la cantidad de música de algún género en particular en cualquier plataforma de *streaming* dependerá de las tendencias del mercado y de su público meta.

Con base a estos criterios se entrenará el modelo sin realizar un balance de clases, considerando que este escenario pondera de forma natural la complejidad real en la industria.

Para tener un mejor entendimiento de las métricas exploradas, se sintetiza la investigación de métricas aplicadas a problemas de clasificación multiclase desbalanceados.

Métrica	***Pros***	***Contras***
Accuracy	Sencilla de entender. Se puede establecer un modelo base para conocer la mejora predictiva de los modelos.	No refleja la realidad predictiva del modelo, pues si se asigna una nueva observación a las clases mayoritarias, existirá una alta probabilidad de acertar.
F1 - Micro	Pondera su resultado a la clase más grande.	En realidad, se obtiene el mismo resultado que el *accuracy*.
F1 - Macro	Pondera su resultado a la clase más pequeña.	Su comportamiento no es del interés real del caso de uso final.
ROC	Se puede conocer un comportamiento por clase.	Ha demostrado poca eficacia con desbalance de clases por no tomar en cuenta la aleatoriedad.
Kappa	Toma en cuenta la aleatoriedad por cada clase.	Su naturaleza no es para uso en predicciones, pero puede adaptarse con cierta prudencia.

Con base en el análisis de métricas realizado, se seleccionaron:

- **Kappa**: Permite conocer el nivel de acuerdo entre dos anotadores o expertos, el cual en este caso será utilizado entre la predicción y la etiqueta real, será la métrica que permita decidir entre lo que se considerará un mejor modelo en comparación a otro. Con el objetivo de conocer su mejoramiento respecto a un modelo base se utilizará además el ***accuracy*** como una forma de comparación, más intuitiva.

- **Matriz de confusión**: Para analizar con detenimiento el comportamiento del modelo final, se utilizará la matriz de confusión para obtener algunas conclusiones.

3. Análisis y resultados

3.1.1 Machine learning

También conocido como aprendizaje automático, es una rama de la Inteligencia Artificial y las ciencias computacionales cuyo objetivo principal es desarrollar técnicas que permitan a la computadora “aprender” por sí misma.

En la actualidad existe una vasta cantidad de algoritmos que mezclan técnicas estadísticas y matemáticas en general con poder computacional para encontrar o resolver problemas casi siempre asociados a predicción.

Estas predicciones se categorizan en dos tipos principales:

- **Regresión**: Son aquellas en las que se requiere predecir un número continuo, por ejemplo, dado los datos relacionados al sector vivienda, predecir el costo de una casa.
- **Clasificación**: Son aquellas predicciones en las que se desea saber si una observación pertenece a una clase u otra, y dependiendo del número de clases y la forma en la que se desea clasificar, se les llama de distintas maneras.
    - *Binary*: Se predice si una observación pertenece a una clase u otra, por ejemplo, en el diagnóstico de cáncer, se tiene o no se tiene la enfermedad.
    - *Multiclass*: Se predice a qué clase pertenece una observación cuando existen más de 2 clases.
    - *Multilabel*: Se etiqueta a una observación en un conjunto de clases a las que podría pertenecer.

  Cabe destacar que un conjunto considerable de estos algoritmos, no dan como resultado una clasificación, sino una probabilidad de pertenecer a una clase y por lo tanto la decisión de clasificación queda en manos de un humano.

Para este proyecto fueron seleccionados algoritmos de *machine learning* que han demostrado eficacia en la clasificación *multiclass*, debido a que el objetivo mensurable es clasificar un archivo de audio en alguno de los 16 géneros musicales padres.

Además, los algoritmos utilizados para este proyecto fueron seleccionados por criterios de diversidad en la forma de resolver un problema de clasificación, por lo que se omiten pruebas de algunos algoritmos de poca eficacia para problemas de clasificación *muticlass*, o por su parecido a otro de los algoritmo seleccionado.

Los algoritmos podrán ser identificados de la siguiente manera:

- *Statistical learning*
    - **LR** *(Logistics regression* con optimización *LBFGS):*
    - Clasificadores Bayes:
        - **NB** (*Gaussian Naive Bayes*)
        - **QDA** (*Quadratic discriminant análisis)*
- *Lazy Learning*
    - **Knn** (*K- nearest neighbor)*
- *Machine learning*
    - *Vector de soporte:*
        - **SVC** (Support Vector Classifier)
        - **linSVC** (*Linear Support Vector Classifier)*
    - Basados en árboles**:**
        - **RF** (*Random Forest)*
        - **GBC** (*Gradient Boosting Classifier)*
        - **XGB** *(Extreme Gradient Boosting Classifier)*

El conjunto de variables y técnicas de *MIR,* pueden ser identificados por:

- **ctr***: Spectral contrast*
- **chr**: *Chroma cens*
- **cen**: *Spectral centroid*
- **ton**: *Tonnetz*
- **zcr:** *Zero Cross Rate*
- **all**: Todos las técnicas *MIR* explorados
- **columns_mda**: selección de variables por *MDA*
- **pca:** 4 componentes principales como método de selección de variables

3.1.1.1 Selecciones variables y algoritmos

Como una estrategia de selección de modelos y al mismo tiempo de selección de variables se realizó una prueba independiente de cada uno de los conjuntos de variables obtenidas en cada método de recuperación de información musical.

Los resultados obtenidos son:

	dim	*LR*	*kNN*	*SVC*	*linSVC*	*RF*	*NB*	*QDA*	*XGB*	*GBC*	*mean*
mfcc	140	42.31	42.37	**46.45**	42.06	42.79	23.97	24.73	45.32	44.12	*39.35*
spectral_contrast	49	36.99	38.05	42.57	36.89	39.83	24.89	20.10	41.30	40.09	*35.64*
spectral_centroid	7	30.95	35.39	35.42	30.77	33.80	19.98	22.01	36.32	35.47	*31.12*
zcr	7	30.27	34.66	34.69	30.44	33.26	22.28	22.53	36.00	35.38	*31.06*
spectral_rolloff	7	31.14	35.20	35.39	31.42	33.57	17.25	18.71	35.65	34.92	*30.36*
spectral_bandwidth	7	30.40	33.92	33.77	30.65	32.68	20.19	21.80	34.29	33.58	*30.14*
tonnetz	42	32.06	33.29	36.00	31.94	34.12	15.48	14.84	35.13	33.84	*29.63*
rmse	7	31.93	34.66	35.01	32.04	33.24	9.92	9.81	35.00	34.69	*28.48*
chroma_stft	84	32.96	34.56	37.74	33.25	35.65	1.95	2.17	37.87	36.34	*28.05*
chroma_cens	84	32.34	32.12	36.42	32.37	33.47	6.89	10.27	34.56	33.55	*28.00*
chroma_cqt	84	29.73	31.74	36.38	29.69	33.11	1.81	2.03	35.39	33.95	*25.98*
mean_algorithm	-	*32.82*	*35.09*	*37.26*	*32.87*	*35.05*	*14.96*	*15.36*	*36.98*	*35.99*	-

Tabla 4 - Porcentaje de aciertos por técnica MIR y algoritmo ML con valores en media por algoritmo y por técnica MIR.

Con una media de 39.35% de *accuracy*, el método Mel *Frequency Cepstral Coefficients* demostró ser el mejor método de recuperación de información musical para la clasificación por género, siendo el algoritmo *Support Vector Classifier* con 46.45% de *accuracy*, el mejor algoritmo.

	LR	*kNN*	*SVC*	*linSVC*	*RF*	*NB*	*QDA*	*XGB*	*GBC*
chroma_cens	0.07	10.66	55.01	12.24	0.59	0.02	0.04	18.02	40.30
chroma_cqt	0.04	8.94	37.42	9.42	0.59	0.02	0.04	18.37	39.77
chroma_stft	0.04	8.14	35.50	8.89	0.55	0.03	0.04	18.44	38.94
mfcc	0.06	13.52	44.32	11.28	0.55	0.04	0.07	34.60	55.64
rmse	0.02	0.19	7.60	2.73	0.50	0.00	0.00	1.75	10.20
spectral_bandwidth	0.01	0.16	10.05	3.05	0.54	0.00	0.00	2.04	10.55
spectral_centroid	0.01	0.14	8.92	3.05	0.53	0.00	0.00	2.02	10.52
spectral_contrast	0.02	5.61	20.48	6.08	0.52	0.01	0.02	12.13	30.59
spectral_rolloff	0.01	0.14	8.57	2.93	0.47	0.00	0.00	1.52	9.14
tonnetz	0.02	5.07	25.72	5.85	0.57	0.01	0.02	10.42	28.47
zcr	0.01	0.13	7.84	2.94	0.47	0.00	0.00	1.56	9.22

Tabla 5 - Tiempo de ejecución en minutos por técnica MIR y por algoritmo ML.

Con un **tiempo total de ejecución de 13.22 horas,** se observa que, algoritmos como *Support Vector Machine* o *Gradient Boosting* consumen un excesivo tiempo de entrenamiento.

Con la información obtenida en el entrenamiento de diversos modelos para cada método de recuperación de información musical y conociendo la diversidad de información que se puede obtener de cada técnica *MIR*, se configuraron algunas combinaciones, así como se incluyeron los métodos de selección de variables antes explorados, en búsqueda de mejorar el *número de aciertos.*

	dim	LR	kNN	SVC	linSVC	RF	NB	QDA	XGB	mean
columns_mda	40	41.79	42.60	45.96	41.34	43.36	29.22	34.10	44.64	*40.38*
mfcc/ctr	189	44.31	42.24	48.30	43.84	43.49	24.75	27.97	46.95	*40.23*
mfcc/ctr/cen	196	44.60	42.16	48.15	43.93	43.57	24.71	27.78	46.86	*40.22*
mfcc/ctr/chr/cen/ton	322	45.90	41.49	49.11	45.21	42.41	20.74	27.96	47.78	*40.08*
mfcc/ctr/chr/cen/rmse	287	45.77	41.48	48.82	45.10	43.15	20.21	26.91	47.52	*39.87*
mfcc/ctr/chr/cen	280	45.31	41.26	48.47	44.41	42.27	20.53	27.27	47.28	*39.60*
mfcc/ctr/chr	273	44.98	41.26	48.30	44.46	42.54	20.35	26.96	47.23	*39.51*
all	518	46.97	40.80	**49.40**	43.74	42.16	4.60	10.69	47.91	*35.78*
pca	4	23.51	24.13	24.83	23.58	21.37	23.96	24.03	24.49	*23.74*
mean_algorithm	-	*42.57*	*39.71*	***45.70***	*41.73*	*40.48*	*21.01*	*25.96*	*44.52*	-

Tabla 6 - Porcentaje de aciertos por combinatorias MIR y algoritmo ML con valores en media por algoritmo y por combinatorias MIR.

Se observa que la selección de variables por *MDA* obtuvo el mejor resultado con 40.38% de aciertos en media, sin embargo, *SVC* logró obtener con una combinatoria de todas las técnicas *MIR*, un nivel de acierto de **49.40%.** En media por algoritmo, el mejor ha sido de igual manera *SVC* con un **45.70**%.

	LR	kNN	SVC	linSVC	RF	NB	QDA	XGB
mfcc/ctr	0.07	16.85	51.27	11.87	0.54	0.05	0.09	45.78
mfcc/ctr/chr	0.11	24.50	73.10	15.03	0.56	0.07	0.16	67.78
mfcc/ctr/cen	0.07	17.49	52.83	12.31	0.56	0.05	0.10	51.27
mfcc/ctr/chr/cen	0.11	24.87	75.13	15.45	0.56	0.07	0.16	70.62
mfcc/ctr/chr/cen/ton	0.13	28.67	87.29	17.09	0.57	0.08	0.20	73.21
mfcc/ctr/chr/cen/rmse	0.11	25.93	77.01	15.54	0.56	0.08	0.18	64.30
all	0.22	41.10	128.33	26.65	0.62	0.15	0.45	118.42
columns_mda	0.03	4.89	17.59	6.22	0.52	0.01	0.02	10.08
pca	0.01	0.09	9.37	2.67	0.55	0.00	0.00	1.42

Tabla 7 - Tiempo de ejecución en minutos de algoritmos ML por combinatoria de técnicas MIR.

El **tiempo total de ejecución total es de 23.16 horas,** donde algoritmos como SVC y XGB, demostraron un costoso tiempo de entrenamiento por encima de las 2 horas de ejecución con todas las variables obtenidas de las técnicas *MIR*.

Cabe destacar que debido al excesivo tiempo de ejecución del algoritmo *Gradient Boosting* y su similitud con *Extreme Gradient Boosting*, se ha descartado dicho algoritmo para este análisis de combinatoria de técnicas *MIR*.

3.1.1.2 Hiperparametrización y Ensemble

Con los datos obtenidos en el análisis anteriores se tomaron las siguientes consideraciones para la construcción de un ensemble y la hiperparametrización de 3 modelos:

- **Tipo de ensemble**: Se ha elegido un **ensemble por votación** donde la predicción de cada modelo tendrá el mismo peso. La selección de este tipo de ensemble se debe a que es sencillo de entender, poco costoso y permite la integración de algoritmos diversos al no requerir de cálculo de probabilidades por parte de los algoritmos.
- **Algoritmos**: Se seleccionaron 3 algoritmos basados **en su nivel de acierto, su tiempo de ejecución y su diversidad de construcción**. Además, para lograr tener un ensemble por votación se seleccionó un número impar:
    - *Xtreme Gradient Boosting*
    - *Logistic Regression*
    - *Linear Support Vector Machine*
- **Variables seleccionadas**: En base a los 3 algoritmos seleccionados, se seleccionó el conjunto de técnicas *MIR* con mayor media en aciertos.
- **Hiperparametrización**: Se seleccionaron hiperparámetros específicos para cada modelo, tomando en cuenta recomendaciones de hiperparametrización optimizada con aspectos como:
    - Selección de hiperparámetro más significativos
    - Rango de valores con crecimiento logarítmico
- **Validación cruzada**: Se utiliza el método de *RandomizedSearchCV,* con una estrategia de *cross validation* estratificada k=3, y la obtención de 10 modelos Random. Se utiliza la métrica **Kappa** para obtener el mejor modelo tomando en cuenta el desbalance de clases y la probabilidad por azar.
- **Train**: Para la construcción de muestras estratificadas se tiene una base de 93,222 archivos de audio.
- **Test**: Se utiliza un set de datos excluido del entrenamiento, que consiste en 11,121 archivos de audio (10% aproximado del total).

3.1.1.3 Evaluación

El ensemble obtiene un **46.32% de aciertos**, y un *kappa* de 33.22%, lo cual demuestra que se logra un ajuste por encima del modelo base definido.

Music Genre	*Accuracy %*
Blues	0.00
Classical	23.97
Country	0.00
Easy Listening	2.27
Electronic	55.20
Experimental	66.00
Folk	27.99
Hip-Hop	38.98
Instrumental	7.17
International	9.72
Jazz	4.55
Old-Time / Historic	74.24
Pop	1.59
Rock	71.84
Soul-RnB	0.00
Spoken	6.45

Tabla 8 - Porcentaje de aciertos por género musical del set de prueba.

Observaciones:

- Se puede concluir que el modelo tiene un valor predictivo malo en las clases minoritarias, obtenido en 3 de ellas 0% de aciertos.
- Las 3 clases mayoritarias: *experimental, rock y electrónica* ejercen una clara tendencia a que otros géneros sean clasificados como de estas 3 clases prioritarias.
- El género *old-time / historic*, a pesar de ser una clase minoritaria tiene el más alto ajuste predictivo de todo los géneros musicales.
- El género *hip-hop* tiene un nivel de confusión alto con géneros como electrónica y experimental.

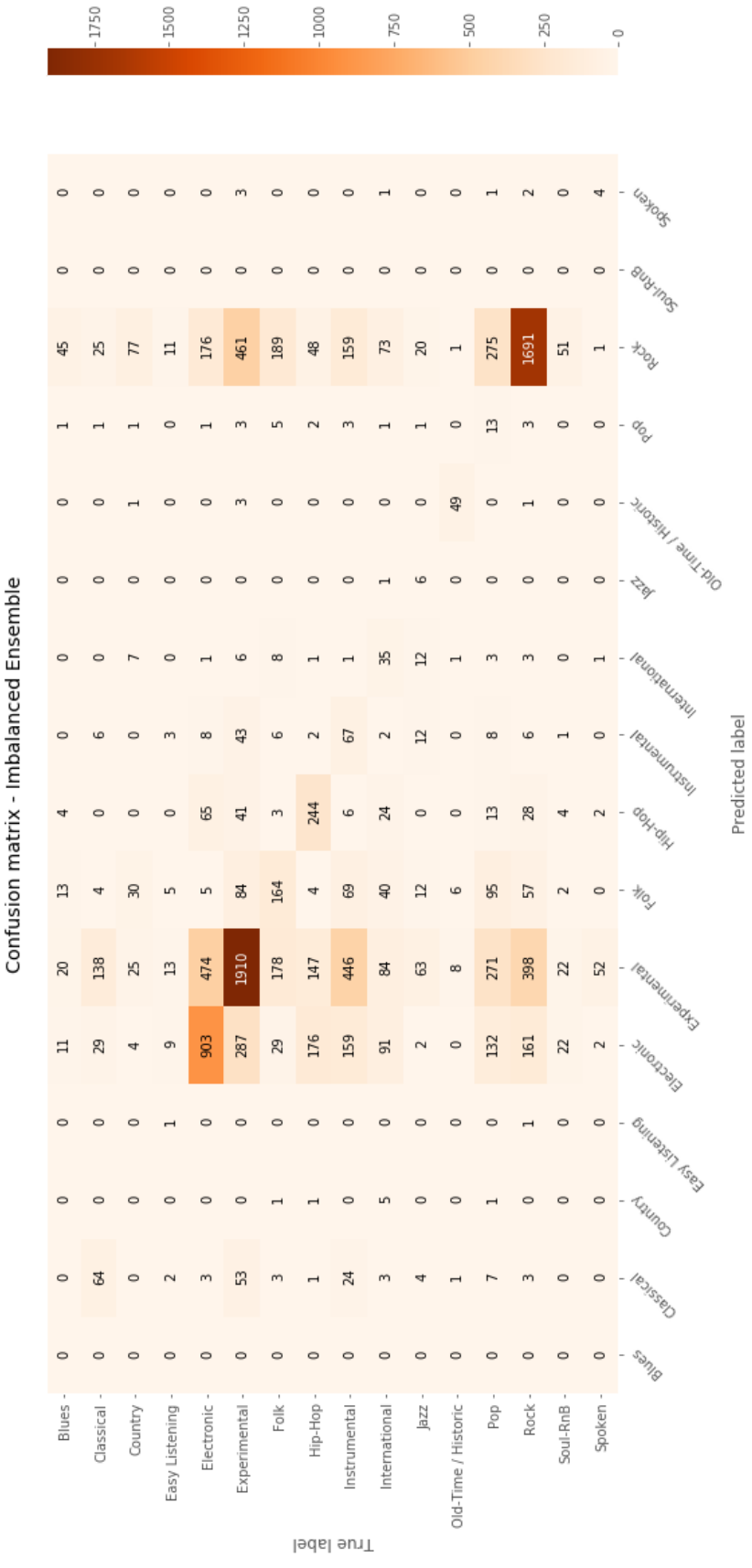


Imagen 29 - Matriz de confusión – Ensemble de datos desbalanceados.

3.1.2 Deep learning

Se trata de una clase particular de algoritmos que forman parte del aprendizaje automático. Como parte de sus características principales es que utilizan múltiples capas de abstracción que se conectan entre sí, transformando y aprendiendo características de los datos de entrada. Regularmente se suelen asociar a una red neuronal, donde cada capa tiene un conjunto de neuronas que se conectan con neuronas de otras capas.

3.1.2.1 Preprocesado

Del aprendizaje adquirido en los modelos anteriores y la investigación realizada, el preprocesado para los modelos *deep learning* propuestos es el siguiente:

- Debido a que una red neuronal requiere que todos sus datos de entrada tengan una misma dimensión, se recortan audios de 30 segundos por cada archivo de audio, con un comienzo de tiempo aleatorio al que se le restan 30 segundos.
- Se obtiene un espectrograma que transforma los archivos de audio en una matriz que representa el espectrograma con respecto a la escala Mel, y cuya dimensión es de 646 x 128 valores.
- Los datos de la matriz son escalados en valores de -1 a 1.

3.1.2.2 Arquitecturas

El entrenamiento de modelos está basado principalmente en 3 arquitecturas distintas, encontradas en artículos científicos y repositorios donde se demostró su potencial para la clasificación de archivos de audio por su género musical. Las 3 arquitecturas probadas son:

- **Arquitectura *Convoluciona - Spotify***: Es un modelo basado en redes convolucionales difundido por *Sander Dieleman,* el cual fue entrenado para generar listas de recomendación para data sets obtenidos de *Spotify*.
- **Arquitectura Convolucional – *DeepSound***: Es utilizado por la aplicación *DeepSound* para la clasificación de 8 géneros musicales basados en el set de datos *GTZAN.*
- **Arquitectura *CNN – LSTM*:** Es una exploración propia en la que se intenta mezclar el potencial comprobado de las redes convolucionales, con una capa recurrente tipo *LSTM (Long Short Term Memory)*, tratando de recuperar más riqueza de información respecto al tiempo.

Debido a que la arquitectura *CNN – LSTM*, incluye todos los tipos distintos de capas que se usan en los otros modelos, siendo además la arquitectura con la que se obtuvo el mejor ajuste, a continuación, se describe a detalle.

La arquitectura *CNN–LSTM* está compuesta por los siguientes tipos de capas:

- ***Convolutional****:* Es una operación de producto de matrices donde se multiplica una pequeña matriz de los datos de entrada por determinados valores o pesos de tamaño pequeño y de dimensiones determinada llamado *kernel*. Es utilizado para encontrar relación entre un dato dado y sus valores más próximos.
- ***Pooling****:* Es una operación que disminuye la dimensión de una entrada, tomando una matriz de dimensión *n* y obteniendo un valor escalar que la representa, en este modelo se hace uso de *MaxPolling*, cuyo valor escalar es el valor máximo de la submatriz de dimensión *n*.
- ***Dropout****:* Facilita la implementación de una regularización, la intención es la disminución del sobreajuste del modelo, apagando algunas de las neuronas de forma aleatoria durante el entrenamiento.
- ***LSTM****:* Permite el aprendizaje con respecto a eventos o datos anteriores, es útil para datos que dependen del tiempo. En el modelo propuesto, es utilizado para aprender respecto al tiempo de duración del archivo de audio.
- ***Time distributed****:* Permite la creación de capas densas u ocultas en el dominio del tiempo, su entrada debe ser datos que dependen del tiempo y por lo tanto contar con al menos 3 dimensiones.
- ***Lambda****:* Permite crear una capa personalizada, en el modelo propuesto se utiliza para obtener una media de los valores en la dimensión del tiempo, disminuyendo así su dimensión para tener solo un valor medio por clase.

Utilizando las siguientes funciones de activación:

- **Relu:** Una función de activación que suele utilizarse por su fácil derivación y comprobado potencial para redes neuronales. Su comportamiento es muy sencillo si su valor de entrada es menor o igual 0, entonces su salida es 0, si es mayor a 0 su salida es igual a la entrada.
- **Softmax**: Es útil para problemas de clasificación, forzando a que la suma de las predicciones realizadas para todas las clases sea igual a 1.

Debido a su demostrada superioridad sobre otros métodos de optimización en redes neuronales, se hace uso de un optimizador ***Adam*** con un *learning rate* de 0.0001, utilizando *ReduceLROnPlateau*, para reducir el *learning rate* cuando el valor de la función de pérdida en la set de validación comience a detener su disminución.

Se implementa una estrategia de *EarlyStopping* en el que, al no existir mejoramiento en la función de pérdida del set de validación en por lo menos dos épocas, se detiene su entrenamiento.

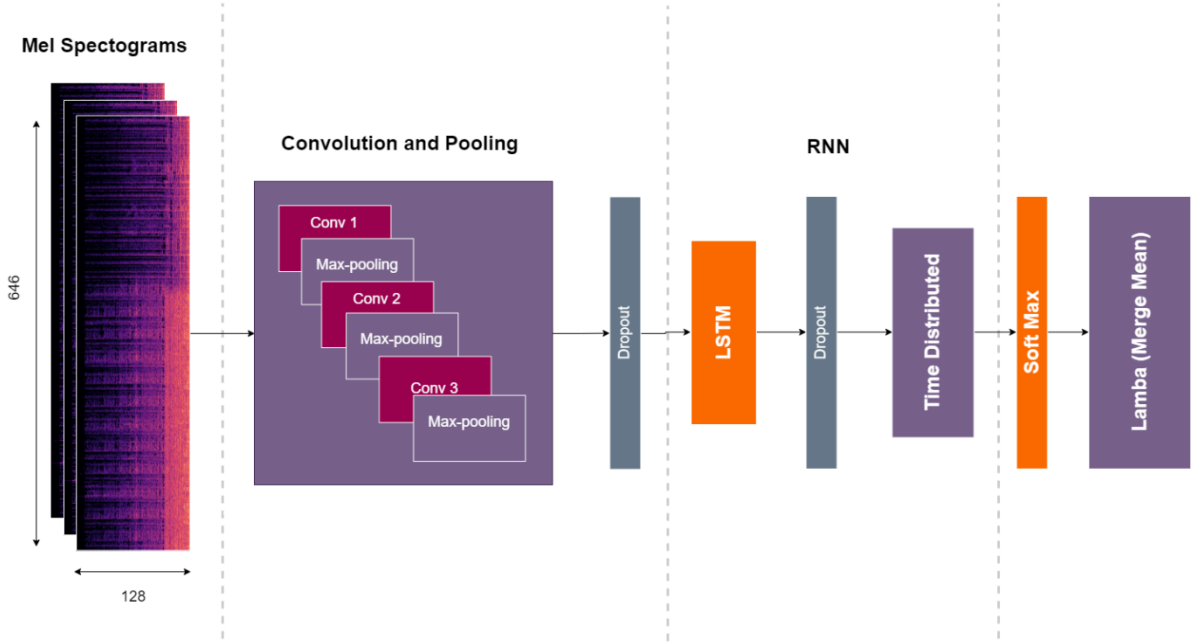


Imagen 30 - Arquitectura de red neuronal convolucional – LSTM

En la tabla siguiente, se puede observar las dimensiones tratadas en cada capa, así como el número de parámetros entrenados, con un total de **1'349,392.**

Layer (type)	*Output Shape*	*Parameters*
input (InputLayer)	(None, None, 128)	0
convolution_1 (Conv1D)	(None, None, 256)	164,096
activation (Activation)	(None, None, 256)	0
max_pooling1d (MaxPooling1D)	(None, None, 256)	0
convolution_2 (Conv1D)	(None, None, 256)	327,936
activation_1 (Activation)	(None, None, 256)	0
max_pooling1d_1 (MaxPooling1)	(None, None, 256)	0
convolution_3 (Conv1D)	(None, None, 256)	327,936
activation_2 (Activation)	(None, None, 256)	0
max_pooling1d_2 (MaxPooling1)	(None, None, 256)	0
dropout (Dropout)	(None, None, 256)	0
lstm (LSTM)	(None, None, 256)	525,312
dropout_1 (Dropout)	(None, None, 256)	0
time_distributed (TimeDistributed)	(None, None, 16)	4,112
output_realtime (Activation)	(None, None, 16)	0
output_merged (Lambda)	(None, 16)	0

Tabla 9 - Arquitectura de red neuronal para clasificación de géneros musicales con tamaños de salida por capa y número de parámetro a entrenar.

3.1.2.3 Resultados

De las arquitecturas finales probadas, se puede observar que *CNN-LSTM* tiene el comportamiento más estable, a pesar de comenzar con una valor de función de pérdida menor y un *accuracy* mayor en *test* que en *train* en las primeras épocas.

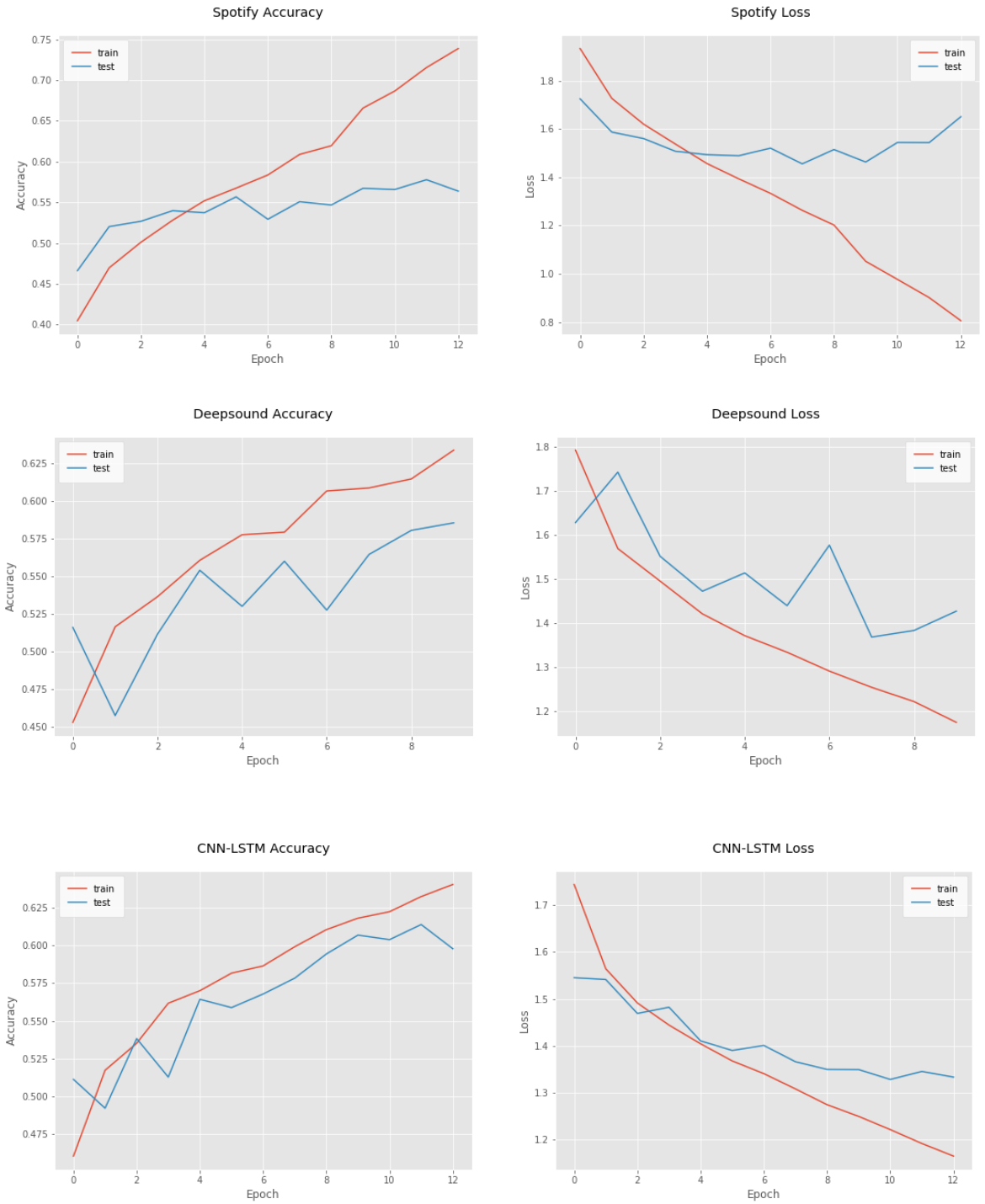


Imagen 31 - Gráficas de función de pérdida y accuracy para modelos deep learning.

En el modelo construido con la arquitectura *Spotify* se observa un claro sobreajuste en entrenamiento y una nula mejora de la función de pérdida en *test.*

La arquitectura *Deepsound* parece tener un comportamiento más estable en las primeras épocas en comparación con la arquitectura *CNN-LSTM,* sin embargo, termina por tener un peor ajuste.

Architecture	*Max. Acc. Train*	*Min. Loss Train*	*Max. Acc. Validation*	*Min. Loss Validation*	*Epochs*
Convolucional	0.73	0.80	0.57	1.45	14
Convolucional Deep Sound	0.63	1.17	0.58	1.37	10
CNN – LSTM	0.63	1.21	0.61	1.37	13

Tabla 10 - Métricas de función de pérdida y accuracy para modelos deep learning.

También se realizaron pruebas con una versión de datos balanceados por *undersampling* del modelo *CNN- LSTM,* logrando un ajuste de apenas el 16% de *accuracy* en validación, lo que demostró un sobreajuste debido a la poca cantidad de datos de entrada (5,376 archivos).

Con un total de **93,896 archivos** de audio como set de entrenamiento y **10,433** archivos como set de validación, el modelo final fue ejecutado en **13 de 60 épocas** planteadas, logrando un mínimo de la función de pérdida en validación de 1.37 basado en *categorical_crossentropy*.

El accuracy máximo obtenido es de **61.38%.** Debe tomarse en cuenta que este valor no es del todo comparable con el resultado del *ensemble* antes propuesto, ya que la métrica de monitoreo utilizada para *deep learning* ha sido *accuracy*, y para el *ensemble Kappa*.

3.1.3 Principales obstáculos enfrentados

- **Tiempo de procesamiento**: En general los tiempos de procesamiento han sido un obstáculo, debido principalmente, a la capacidad física del equipo de cómputo o a la imposibilidad o falta de implementación de algoritmos que pueden ser ejecutados de manera paralela.

- **Falta de recursos físicos:** Algunos de los casos exploratorios que no pudieron ser concretados se debieron a la falta de memoria volátil del equipo disponible, algunas de las estrategias utilizadas lograban realizar paginado en memoria física, como es el caso del método *smote* del paquete *imbalance-learn*, sin embargo, esto causaba la ralentización del proceso de manera que no fue sostenible continuar con las pruebas, algunos de las estrategias de muestreo que se probaron son:

- Balance de clases por *oversampling - Smote*
- Balance de clases por *undersampling – Edited & Condensed Nearest Neighbors*
- Balance de cargas mixto – *Smoteen* (Mezcla de los dos métodos anteriores)

- **Errores de servicios de almacenamiento en la nube:** Se enfrentó con dos problemas principales:

  - Falta de integración de servicio para la carga de archivos de manera sencilla desde archivos disponibles vía *HTTP*.
  - Problemas de integridad de datos al querer utilizar herramientas de línea de comando en ambientes *HDFS*. Una de las soluciones encontradas para lograr la carga de datos masivos desde servicio *HTTP*, no pudieron concretar tareas secundarias como el descomprimido de información y relocalización en el servicio de almacenamiento en la nube.

- **Incompatibilidad de paquetes por versión**: Debido a que se hacen uso de un conjunto de librerías diversas, se enfrentó a la falta de compatibilidad de algunas versiones como lo son:

  - Integración de modelos generados en Python *keras* a su utilización en un navegador web a través de *tensorflow.js.*
  - Integración de librerías de codificación de audio con paquete *librosa.*
  - Versiones incompatibles entre *numpy*, *pandas* y su requisitos con versiones anteriores de *keras*.

3.1.4 Sesgos

El sesgo algorítmico ocurre cuando un sistema informático refleja la cultura de los humanos que están implicados en la codificación y recolección de datos usados para entrenar un algoritmo.

Es importante destacar que en los modelos utilizados se ha identificado un conjunto de sesgos considerables que a continuación se describen y cuya finalidad es la de visibilizar algunas problemáticas derivadas del sesgo algorítmico.

Género musical: Aun cuando regularmente un pieza musical suele estar asociada a un sólo género musical, se debe considerar que las fronteras entre los distintos géneros musicales son más de naturaleza cultural, y que no existe una sola forma de homologación de criterios para definir un género musical.

A su vez, una pieza musical puede tener características de dos o más géneros musicales en el mismo lapso, o dos o más géneros en distintos tiempos que componen la pieza musical. Se debe tomar a consideración que los criterios de asignación a un sólo género musical son muy diversos, ya que puede definirse en términos del estilo del álbum al que pertenece, el

compositor, el músico o cantante que lo ejecuta, así como por aquel género que sea predominante durante más tiempo en la pieza musical.

Música comercial: Debido a que el objetivo de aplicación fijado para esta exploración es el de que pueda ser escalado a un sistema de recomendación, debe tomarse en cuenta que el set de datos utilizado contiene principalmente música comercial-occidental, por lo que un considerable número de patrones que se ha obtenido a través de técnicas de recuperación de información musical o *deep learning* en la red neuronal, se encuentran sesgadas.

Algunas de las características que son diferentes en la música no comercial (música autóctona, tradicional, ritual, oriental, etc.) son:

- Uso de ritmos musicales complejos cuyos patrones de repetición son diversos en el dominio del tiempo.
- Uso de instrumentos musicales no convencionales, fabricados de manera artesanal, instrumentos únicos, construidos con materiales locales no convencionales para la construcción de instrumentos occidentales.
- Reglas armónicas diversas que desde la cultura occidental pueden ser consideradas como disonantes.
- Creación de escalas musicales propias, que no son construidas en términos del sistema de temperamento igual, cuya nota base no se encuentra en la frecuencia de los 440 Hz.

4. Conclusiones y trabajo futuro

Potencial de negocio y aplicación

Los algoritmos de *machine learning* están transformando la industria musical, su aplicación se encuentra actualmente en todos los procesos de la industria; desde el reconocimiento de talento artístico, pasando por la composición y la producción, y como es nuestro caso de estudio, en la venta y recomendación de contenido en plataformas digitales.

La investigación y el trabajo realizado demuestran que las técnicas de recuperación musical basadas en tratamiento de señales, así como el aprendizaje a través de redes profundas, proveen valiosa información acerca de los patrones musicales, permitiendo su utilización en múltiples áreas de aplicación.

Big Data y ambientes productivos

La complejidad computacional de tratar con archivos de audio, y las dimensiones reales utilizadas en la industria requieren de la búsqueda e implementación de las mejores prácticas, así como de la utilización de infraestructura y servicios escalables que soporten el almacenamiento y procesamiento de Big Data.

Modelización y arquitecturas

Los algoritmos de *deep learning* demostraron una mayor eficacia en la predicción de géneros musicales en comparación de otros algoritmos de *machine learning*, y en general en base a la investigación realizada, han demostrado su eficacia en la aplicación en diversas áreas relacionadas al sonido.

La potencia de aprendizaje en datos no estructurados y la flexibilidad de los algoritmos de *deep learning,* permiten aprovechar esfuerzos previos de una manera sencilla, por lo que se convierten en la mejor alternativa para ambientes productivos.

La integración de redes recurrentes *LSTM*, demuestran que el factor temporal en un archivo de audio es de suma importancia para el reconocimiento de patrones musicales, mejorando la predicción y comportamiento de los modelos en los que se incluyeron capas que recuperan dichas características.

Visualización

La visualización a través de espectrogramas, mapas y gráficos en general, estuvieron presentes en todas las etapas de desarrollo de este proyecto:

- Proceso de generación de variables: Facilitó el entendimiento de las técnicas de recuperación de información musical y sus bases matemático-estadísticas.

- Proceso de análisis descriptivo: Permitió entender de mejor manera las características, deficiencias y sesgos del set de datos generado.

- Proceso de entrenamiento de modelos: Facilitó conocer y comparar el comportamiento de los distintos modelos en su fase de entrenamiento y validación.

- Selección de modelos: Facilitó el entendimiento del comportamiento predictivo de los modelos, siendo una pieza clave para la toma de decisiones.

- Pruebas y conclusiones de modelos: La adaptación de una aplicación en la que se visualiza el comportamiento de la predicción del algoritmo de *deep learning* en tiempo real, permite conocer no solo el resultado de clasificación, sino inferir patrones musicales generales en los archivos de música.

4.1 Trabajo futuro

Como parte de trabajos futuros se identificaron 6 ejes de trabajo:

- Generar un set de datos más robusto y confiable que permita explorar tiempos de entrenamiento y complejidad más cercana a la realidad en la industria.

- En el contexto de entrenamiento de modelos, lograr alternativas de entrenamiento del modelo con cargas balanceadas sin un costo de recursos excesivo.
- Explorar más o nuevas arquitecturas *deep learning* que permitan recuperar más y mejores patrones musicales.
- Integración de capas entrenadas de un modelo *deep learning* a un sistema de recomendación basado en filtros colaborativos, mapeando así una mezcla de patrones musicales con información de usuarios.
- Explorar, definir e implementar *pipelines* que permitan el uso de los algoritmos entrenados de forma eficiente y escalable para ambientes productivos basados en tecnologías Big Data.
- Generar *playlists* y ambientes de interacción con usuarios para la validación continua y futura modificación de los algoritmos.

5. Bibliografía y referencias

5.1 Artículos científicos

5.1.1 Music information retrieval

Kaminskas, M., y Ricci, F. (2012). *Contextual music information retrieval and recommendation: State of the art and challenges*.

Sched, M., Emilia Gómez, E., y Urbano, J. (2014). *Music Information Retrieval: Recent Developments and Applications.*

Defferrard, M., Benzi, K., Vandergheynst, P., y Bresson X. (2017). *FMA: A dataset for music analysis.*

5.1.2 Machine and Deep learning

Sergey S., Hamza G., y Alexei A. (*2017*). *Representations of Sound in Deep Learning of Audio Features from Music.*

Keunwoo C., György F., Kyunghyun C., y Mark S. (2017). *A Tutorial on Deep Learning for Music Information Retrieval.*

Zhang, W., Wenkang, L., Xiangmin, X., y Xiaofeng, X. (2016). *Improved Music Genre Classification with Convolutional Neural Networks.*

Irvin, J., Chartock, E., Nadav Hollander, N. (2016). *Recurrent Neural Networks with Attention for Genre Classification.*

Choi, K., Fazekas, G., Sandler M., y Cho, K. (2017). *Transfer learning for music classification and regression tasks.*

Valerio, V., Pereira, R., Costa, Y., Bertolini, D., y Silla Jr. C. (2018). *A Resampling Approach for Imbalanceness on Music Genre Classification Using Spectrograms.*
Pui C., Long K., Kin, Y., Zeng, Z. y Hong, K. (2018). *Music Genre classification using a hierarchical Long Short-Term Memory (LSTM) model.*

Bahuleyan, H. (2018). *Music Genre Classification using Machine Learning Techniques.*

5.2 Artículos y notas

Smith, Z. (18 de diciembre del 2016). Hacker Noon. Medium. (https://hackernoon.com/artificial-intelligence-in-the-music-industry-43e809ecbcde).

Despois, J. (21 de noviembre de 2016) Finding the genre of a song with Deep Learning. (https://medium.com/@juliendespois/finding-the-genre-of-a-song-with-deep-learning-da8f59a61194)

Dwivedi, P. (13 de diciembre del 2018). *Using CNNs and RNNs for Music Genre Recognition*. (https://towardsdatascience.com/using-cnns-and-rnns-for-music-genre-recognition-2435fb2ed6af)

Pandey, P. (13 de diciembre del 2018). *Music Genre Classification with Python*. (https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8

Lerch, A. (5 de noviembre de 2018), *List of MIR Datasets*. (http://www.audiocontentanalysis.org/data-sets/)

Giacaglia, G. (10 de marzo de 2019), Spotify's Recommendation Engine. (https://medium.com/datadriveninvestor/behind-spotify-recommendation-engine-a9b5a27a935)

Recommending music on Spotify with deep learning (05 de agosto de 2015). (http://benanne.github.io/2014/08/05/spotify-cnns.html)

5.3 Videos

Rackspace Developers. (25 de septiembre de 2014). *Music Information Retrieval Using Locality Sensitive Hashing.* [Archivo de vídeo]. Recuperado de: https://www.youtube.com/watch?v=SghMq1xBJPI

Data Council. (10 de noviembre de 2014). *Music Information Retrieval using Scikit-learn.* [Archivo de vídeo]. Recuperado de: https://www.youtube.com/watch?v=oGGVvTgHMHw

Cognitive Builder. (10 de mayo de 2017). *Machine Learning & Big Data for Music Discovery presented by Spotify.* [Archivo de vídeo]. Recuperado de: https://www.youtube.com/watch?v=HKW_v0xLHH4

5.4 Repositorios

Guimarães, H., *Music Genre classification on GTZAN dataset using CNNs*, Consultado en: https://github.com/Hguimaraes/gtzan.keras

Despois, J., *Finding the genre of a song with Deep Learning.* Consultado en: https://github.com/despoisj/DeepAudioClassification

DeepSound. *CNN for Live Music Genre Recognitio*n. Consultado en: https://github.com/deepsound-project/genre-recognition

Pandey, P. *Music-Genre-Classification-with-Python.* Consultado en: https://github.com/parulnith/Music-Genre-Classification-with-Python

Defferrard, M. *Dataset for Music Analysis*. Consultado en: https://github.com/mdeff/fma

Choi, K. *Transfer learning for music classification and regression tasks.* Consultado en: https://github.com/keunwoochoi/transfer_learning_music

Bayle, Y. *Deep Learning for Music (DL4M).* Consultado en: https://github.com/ybayle/awesome-deep-learning-music

Ruotsi, R. *Music Genre Classification with LSTMs.* Consultado en: https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification

6. Anexos

6.1 Diagrama de proceso

El total de horas invertida es de **160 horas,** en un período de 1 mes y medio, donde se incluye el tiempo invertido en la memoria técnica, limpieza de código y documentación en general.

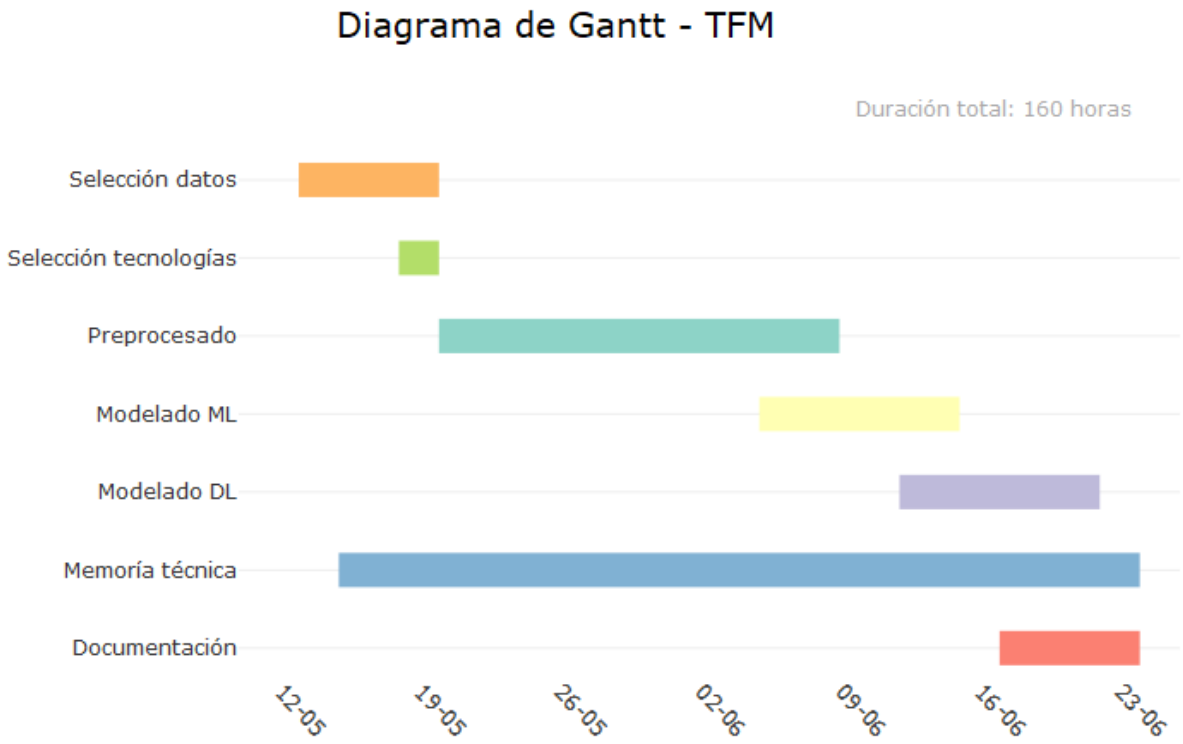


Imagen 32 - Diagrama de Gantt - Proceso TFM.

PROCESO	TAREA	HORAS
Selección de datos	Investigación	2
Selección de tecnologías	Investigación y configuración de ambiente	4
Preprocesado	Investigación	4
Preprocesado	Creación de set de datos	6
Preprocesado	Creación de variables y transformación matricial	40
Preprocesado	Limpieza	4
Preprocesado	Análisis exploratorio	6

Preprocesado	Selección de variables y métricas	2
Modelado ML	Investigación, codificación y pruebas	22
Modelado DL	Investigación, codificación y pruebas	24
Memoria técnica y documentación	Estilo y redacción	40
Desarrollo APP	Adaptación y mejoras APP	6

Tabla 11 - Desglose de tiempo invertido en tareas por proceso.

6.2 Glosario

Acorde: En música es un conjunto de notas que se producen o generan al mismo tiempo, guardan una relación matemática entre ellas, tal que se complementan y tienen un sonido agradable para la cultura occidental.

API's: De sus siglas en inglés *Application programming interface,* es un conjunto de subrutinas, funciones y procedimientos, que ofrecen como servicio, una capa de abstracción para ser utilizados por distintas *softwares*.

Código libre: También conocido como software libre u *open source*, es el software que está licenciado de tal manera que los usuarios pueden estudiar, modificar y mejorar su diseño mediante la disponibilidad de su código fuente.

Disonancia: Es la falta de armonía entre 2 o más notas, se dice que un sonido es disonante si no tiene una relación de frecuencia múltiple con otro sonido.

Hadoop: Es un *framework* de software de la fundación Apache, que soporta aplicaciones distribuidas.

HTTP: De sus siglas en inglés *Hypertext Transfer Protocol,* es el protocolo de comunicación que permite las transferencias de información en la *World Wide Web*.

HDFS: Es un sistema de archivos distribuido, escalable y portátil para el framework Hadoop, que se utiliza para la persistencia de datos y el procesamiento de Big Data.

Humming: Conocido también como tarareo, es la imitación de patrones musicales asociados a la melodía con distintas sílabas o sonidos que no son palabras.

ISO: De sus siglas en inglés *International Organization for Standardization,* es una organización que promueve el uso de estándares propietarios, industriales y comerciales. Estándares para música o representación de idiomas son de utilidad para este proyecto.

LBFGS: Es un método de optimización quasi-Newton, cuya principal característica es su bajo consumo de memoria. A diferencia del método newton, no requiere del cálculo de la matriz Hessiana, haciendo únicamente uso de una función y su gradiente.

Leptocúrtica: En teoría de la probabilidad y estadística, la curtosis es una medida que sirve para analizar el grado de concentración que presentan los valores de una variable analizada alrededor de la zona central de la distribución de frecuencias. Se dice que una distribución es leptocúrtica cuando presenta un alta concentración de curtosis.

MIDI: De sus siglas en inglés *Musical Instrument Digital Interface,* es un estándar tecnológico que describe un protocolo, una interfaz digital y conectores que permiten que varios instrumentos musicales electrónicos, ordenadores y otros dispositivos relacionados se conecten y comuniquen entre sí.

MP3: Es un formato de compresión de audio, que permite entre otras cosas, disminuir el tamaño de un archivo de audio a través de técnicas de muestreo con pérdida.

Audio monofónico: Es el audio que debido a que es grabado con un solo micrófono, al reproducirse se transmite el mismo audio en el número de canales de salida.

Muestreo (Digital): Es un proceso de digitalización de señales, que toma muestras de una señal analógica a una tasa de muestreo constante.

Nota fundamental: En un acorde, es la nota base con la cual se construye un acorde, suele ser predominante al oído por lo que suele llamársele a un acorde por el mismo nombre de la nota fundamental.

PCA: Por sus siglas en inglés *Principal Component Analysis*, es una técnica utilizada para describir un conjunto de datos N dimensional, en términos de nuevas variables llamadas componentes. Las componentes resultantes no son correlacionadas entre ellas y se suelen ordenan en término de cantidad de varianza original que describen, por lo que son útiles como método de selección de variables.

Sistema de temperamento igual: Es el sistema de normalización música occidental creado por la ISO, por el cual se crean y estandarizan las frecuencias de una nota musical. La nota fundamental está basada en una frecuencia de 440Hz, y 12 semitonos. La relación existente entre un semitono y el siguiente es de $F_n = F_{n-1} * \sqrt[12]{2}$, es decir dado un semitono, se puede obtener el siguiente multiplicando por la raíz decimosegunda de 2.

Tasa de bits: Define el número de bits que se transmiten por unidad de tiempo, es la velocidad de transferencia de datos, la unidad de medida suele presentarse como kbit/s, es decir en cientos de bits por segundo, para archivos de audio.

Timbre: Cualidad del sonido de la voz de una persona o de un instrumento musical permite distinguirlo de otro sonido del mismo tono, está asociado a la combinación de diferentes frecuencias y efectos acústicos como el eco y la reverberación.

Anaconda: Es una plataforma para ejecutar lenguajes de programación como *Python* y *R* que facilita la administración de paquetes y el desarrollo y entrenamiento de algoritmos de *machine learning*.

6.3 Índice de imágenes

6.4 Índice de tablas

6.5 Código

Para la consulta del código de los procesos descritos anteriormente se puede consultar el repositorio: https://github.com/ludwigrubio/afi-tfm-mir

Para consultar el proceso relacionado a algoritmos de *machine learning*, se puede consultar la carpeta **tfm-mir/fma-outputs/,** que contiene el siguiente contenido:

6.5.1 Scripts

1. **webapi.py:** Archivo de prueba para utilizar la API FMA.

2. **creation.py:** Es el archivo que crea los archivos de metadatos utilizados.

3. **features.py:** Es el archivo que calcula y organiza la información recuperada a través de técnicas de recuperación musical (*MIR*).

4. **utils.py:** Contiene un conjunto de funciones que son utilizadas de forma horizontal en todos los archivos de código.

6.5.2 Notebooks

1. ***Exploration and cleaning:*** Se puede consultar la exploración de los archivos de metadatos construidos a través de las llamadas a distintos APIs, además del proceso de limpieza de *missing values* para los campos de género musical.

2. ***Descriptive analysis***: Es una análisis descriptivo detallado de los metadatos y el *feature engineering* obtenido en los archivos de audio.

3. ***Machine learning:*** Se muestran el conjunto de exploraciones realizadas para la selección del mejor modelo, así como 2 archivos con la hiperparametrización y generación de *ensembles* con y sin balanceo de clases.

Además, se incluye un archivo **environment_mir.yml** que contiene la configuración del ambiente *Anaconda* para la configuración de dependencias necesarias para ejecutar el código con éxito.

6.5.3 Deep Learning

El código de los 3 modelos de *deep learning* y sus variantes puede ser consultado en la carpeta **app-deep-learning/** como:

- **train_model.py:** Entrenamiento del modelo *CNN – LSTM*.
- **train_model_deepsound.py:** Entrenamiento del modelo *Deepsound*.
- **train_model_spotify.py:** *Entrenamiento del modelo Spotify***.**

Como parte de la visualización dinámica e integrando el mejor modelo de *deep learning*, se realizó la adaptación de la aplicación *CNN for Live Music Genre Recognition* la cual permite visualizar en tiempo real la clasificación de un archivo de audio.

Las modificaciones realizadas consisten en un cambio visual, entrenamiento de modelo propio, exportación a formato *h5* para su utilización del lado del cliente, así como cambios relacionado a los 16 géneros musicales a visualizar.

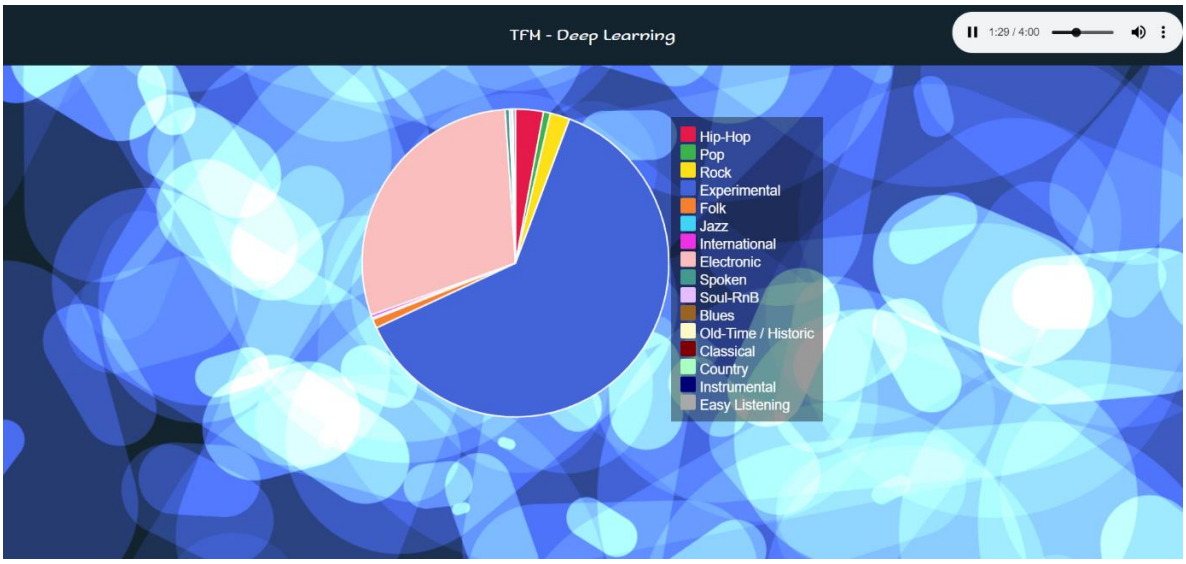


Imagen 33 - APP de visualización dinámica para clasificación de género musical en tiempo real.

Como parte de las tecnologías utilizadas se encuentran:

- HTML
- CSS
- Javascript
- Python
    - h5py
    - librosa
    - numpy
    - scipy
    - tensorflow
    - tensorflowjs

La aplicación puede ser consultada en el dominio:

https://datalud.com/music/genre/