



INGENIERÍA

Trabajo de Grado

Songo

Profesora: Mariana Falco

Director: Ignacio Berdiñas

Alumnos: Facundo Bocalandro, Facundo Rivas

1 - Índice

| | |
|--|-----------|
| 1 - Índice | 2 |
| 2 - Resumen ejecutivo | 3 |
| 3 - Introducción | 4 |
| 3.1 - Motivación | 4 |
| 3.2 - Problemática | 5 |
| 3.3 - Objetivo | 6 |
| 4 - Marco teórico | 7 |
| 5 - La aplicación | 24 |
| 6 - Pruebas y validaciones | 26 |
| Resumen de experimentos: | 40 |
| 7 - Tecnologías | 41 |
| 7.1 - Front End | 41 |
| 7.2 - Back End | 42 |
| 7.3 - Inteligencia Artificial | 43 |
| 8 - Arquitectura | 45 |
| 9 - Modelo de datos | 50 |
| 10 - Usuarios y funcionalidades | 53 |
| 10.1 - Tipos de usuarios | 53 |
| 10.2 - Definición de funcionalidades | 53 |
| 10.3 - Especificación de user stories | 54 |
| 11 - Interfaz de usuario | 62 |
| 12 - Dificultades encontradas y soluciones propuestas | 68 |
| 13 - Propuestas de mejora | 71 |
| 14 - Validación del mercado | 75 |
| 15 - Conclusiones | 77 |
| 16 - Bibliografía | 78 |

2 - Resumen ejecutivo

Songo es un sistema creado con el objetivo de ser una herramienta de gran utilidad para aquellas personas que se dedican o tienen como hobby alguna actividad relacionada con la música. DJs, productores, vendedores, son algunos de los potenciales usuarios de la aplicación.

La principal funcionalidad de Songo es la clasificación de géneros de música mediante algoritmos de machine learning. Esto es, dando como input al sistema un track, luego de analizarlo da como output a cuál género pertenece. La versión actualmente desarrollada clasifica tracks pertenecientes a la música electrónica en subgéneros de la misma.

Además de esta funcionalidad el sistema cuenta con herramientas que agregan mucha eficiencia al trabajo diario de los usuarios anteriormente mencionados. Por ejemplo, dado un directorio donde el usuario posee gran cantidad de tracks, el sistema es capaz de ordenar en subdirectorios los archivos de música, clasificándolos y agrupándolos según el subgénero al cual pertenecen.

Para el desarrollo de este proyecto fue necesario realizar un proceso extensivo de iteración con respecto a los algoritmos de machine learning. En primer lugar fue creado el set de datos con el cual se entrenó el modelo, obteniendo tracks de diferentes sitios de descarga de música, luego comenzaron las pruebas e iteraciones con los algoritmos de machine learning hasta llegar al final y en último lugar se desarrolló la aplicación desktop.

3 - Introducción

3.1 - Motivación

La música es un elemento común en la vida de las personas, ya que es un componente que acompaña a las personas a lo largo de su vida, y en diferentes momentos de la misma sirve de motivación, consuelo, distracción, etc. Es por ello que la industria musical es cada vez más grande. La gente consume música todo el tiempo, y para estos momentos diferentes es necesario que existan diferentes géneros de la misma, que acompañen y estén en sintonía con las emociones y vivencias que están presentes. A algunos les gusta más el Rock, a otros el Jazz, a otros la música Pop, y así la lista puede extenderse largamente.

En la industria existen numerosos tipos de trabajos y hobbies que la gente realiza, algunos de estos pueden ser: productores, DJs, vendedores, entre otros. Estas personas que interactúan con la música a diario, ya sea escuchando, descargando, produciendo, comprando, o sea la actividad que sea, están en contacto con diferentes tipos de géneros musicales constantemente.

Ahora bien, sea cual sea el nivel de conocimiento musical, es muy difícil describir qué es un género musical. ¿Por qué el jazz suena como jazz? ¿Cómo se puede distinguir el country o la electrónica? Como no existe una definición sistemática para los géneros, es imposible clasificarlos programáticamente con simples declaraciones if/else.

3.2 - Problemática

Debido a lo anteriormente mencionado, a la hora de decir que de género es una canción o track sin tener información previa del mismo se obtienen diferentes opiniones. En algunos casos es más sencillo distinguir los géneros, por ejemplo, cuando se compara una canción de Rock con una de Jazz, o Hip Hop, es relativamente sencillo identificar los géneros. Dentro de todos los géneros musicales existen subgéneros de música. Por ejemplo en el caso del Rock, existen: alternativo, progresivo, indie, entre otros. Cuando se trata de distinción o clasificación de subgéneros, se pone un poco más complicado.

Es acá donde se encuentra una parte de la problemática, pero antes de continuar, vale la pena aclarar que el trabajo va a estar centrado en el género de la electrónica y los subgéneros de la misma. Muchas veces los DJs o productores no saben de qué géneros son los tracks con los que trabajan. Por ejemplo, cuando un DJ prepara un set para tocar en algún lugar, se ve involucrado en un proceso de escucha y selección de música que puede durar mucho tiempo, y en algunos casos no tener una respuesta clara sobre qué es lo que está escuchando.

Otra parte de la problemática es natural del mismo proceso de selección y escucha. Es mucho el tiempo que una persona tiene que invertir en repasar las librerías de música que se van formando con el pasar del tiempo. La falta de una herramienta que ayude en el proceso de clasificación es parte del problema.

Entonces, tenemos una problemática formada por dos componentes: la primera, nace de la falta de un definición concreta de que es un género y cómo se identifica, y la segunda del proceso de selección y clasificación de música.

3.3 - Objetivo

El objetivo del proyecto es desarrollar una aplicación de escritorio que permita plantear una solución a las problemáticas anteriormente mencionadas. Por un lado, poder obtener un algoritmo de machine learning el cual sea capaz de clasificar con un grado aceptable de precisión un archivo musical que se le de como input. De esta manera, y con el pasar del tiempo utilizando el sistema, dicho algoritmo podría perfeccionarse siendo entrenado en diferentes géneros y variantes, reduciendo el error generado por las personas al clasificar un track. Acompañando esto, también se busca desarrollar dentro de la aplicación una funcionalidad que permita dado un directorio, ordenar los archivos que se encuentran dentro del mismo en subdirectorios dependiendo del género clasificado por el algoritmo. Esto ahorraría mucho tiempo a aquellas personas que trabajan a diario con la música y manejan grandes volúmenes de canciones.

4 - Marco teórico

Para la implementación del modelo predictivo, se realizó el análisis de 100 tracks de 8 subgéneros distintos con la ayuda del Deep Learning.

Inteligencia artificial

Se conoce como Inteligencia artificial a toda demostración de inteligencia (percepción, síntesis e inferencia de información) por parte de máquinas, en oposición a la inteligencia desarrollada por humanos y animales.

El objetivo principal de la inteligencia artificial es desarrollar máquinas capaces de pensar y actuar como humanos. Para ello, existen distintos tipos de inteligencia artificial:

- Máquinas reactivas: Sistemas que sólo reaccionan, no utilizan experiencias pasadas para tomar decisiones y por ende no forman una memoria.
- Memoria limitada: Sistemas que referencian al pasado, y agregan información que se mantiene por un periodo de tiempo limitado.
- Teoría de la mente: Sistemas capaces de entender las emociones humanas y cómo estas afectan en las decisiones.
- Autoconciencia: Sistemas diseñados y creados para ser conscientes de sí mismos. Entienden sus propios estados internos, así como también pueden predecir los sentimientos de otras personas y actuar acordemente.

Una aplicación de la Inteligencia Artificial es el Machine learning.

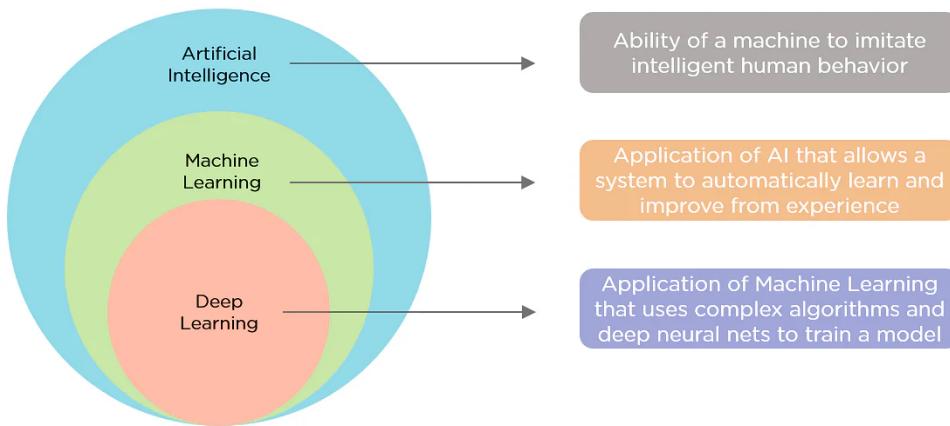


Figura 4.1. Relación entre Inteligencia artificial, Machine Learning y Deep Learning.

Fuente: Shruti M. (Nov. 2022). *Discover the Differences Between AI vs. Machine Learning vs. Deep Learning*.

<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning#:~:text=Artificial%20Intelligence%20is%20the%20concept.algorithms%20to%20train%20a%20model>

Machine learning

Machine learning es una disciplina de las ciencias de la computación que utiliza algoritmos y datos analíticos para construir modelos predictivos capaces de resolver problemas de negocio.

Traduciendo la definición de la empresa Nvidia, "Machine Learning es la práctica de utilizar algoritmos para analizar datos, aprender de ellos y luego hacer una determinación de predicción acerca de algo en el mundo."

Traduciendo la definición de McKinsey & Co, "Machine learning está basado en algoritmos que pueden aprender por datos sin la necesidad de la programación basada en reglas".

A pesar de haber múltiples definiciones para el concepto de Machine Learning, en todos los casos funciona de la misma manera. Se utilizan enormes cantidades de datos, tanto estructurados como no estructurados, los cuales

permiten a la máquina aprender sobre un tópico particular para luego predecir el futuro. El aprendizaje se realiza a través de diferentes técnicas y algoritmos.

Hay tres categorías principales de Machine Learning:

- **Aprendizaje supervisado:** Los datos de entrada ya están etiquetados por una persona humana, es decir, se conocen los valores de la variable de salida. Los problemas a resolver por el aprendizaje supervisado se separan en dos tipos: Clasificación, aquellos que deben clasificar un objeto en diversas clases predeterminadas; Regresión, aquellos que deben predecir un valor numérico.
- **Aprendizaje no supervisado:** Los datos de entrada no están etiquetados, por lo que el objetivo del sistema es buscar patrones y relaciones entre los datos por su cuenta. Los sistemas son capaces de identificar características ocultas. Existen dos tipos de algoritmos utilizados para este propósito: Clustering, clasifica en grupos según similitud de características; Asociación, descubre reglas dentro del conjunto de datos.
- **Aprendizaje por refuerzo:** Tiene como objetivo entrenar a un agente de software para completar una tarea en un entorno desconocido. El agente envía acciones al entorno y recibe observaciones y una recompensa a cambio. La recompensa mide qué tan exitosa fue la acción con respecto al objetivo de completar la tarea.

Esta disciplina comprende siete pasos, los cuales son:

1. Recolectar datos
2. Preparar los datos

3. Elegir el modelo
4. Entrenar el modelo
5. Evaluar el modelo
6. *Parameter tuning*
7. Predicción o inferencia

El problema a resolver por Songo es supervisado, ya que los tracks están etiquetados previo al análisis, y de clasificación, ya que queremos obtener una de varias clases como resultado (género musical). Algunos algoritmos conocidos para clasificación multiclasé son:

- *Logistic Regression*: Utiliza una función sigmoidea y umbrales para determinar la clase de los datos de entrada.
- *Support Vector Machines*: Tiene como objetivo encontrar planos en el espacio n-dimensional (siendo n el número de características) que tengan la mayor distancia entre los *data points* de las distintas clases.

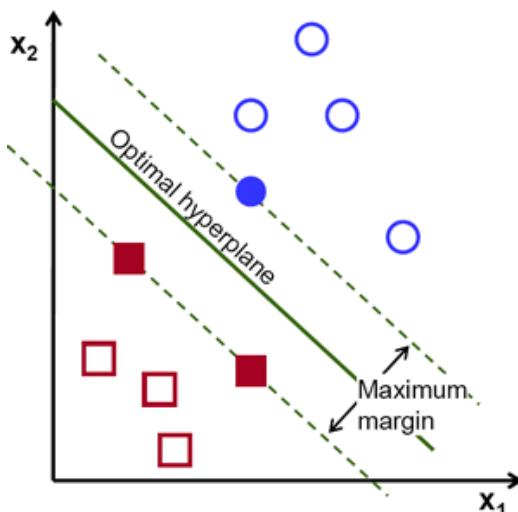


Figura 4.2. Posibles hiperplanos.

Fuente: Rohith Gandhi (Jun. 2018). *Support Vector Machine – Introduction to Machine Learning Algorithms*.

<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

- **K-Nearest Neighbors:** Toma los k (número natural) “vecinos” más cercanos, es decir, los puntos de datos con menor distancia al dato siendo analizado, y luego calcula la moda de las clases de dichos vecinos. La moda de las clases, es decir la clase que aparece con mayor frecuencia entre esos puntos de datos, será el resultado predictivo.
- **Decision Trees:** Es un modelo de decisiones en forma de árbol. Para su correcto funcionamiento, se deben elegir las mejores características a utilizar, así como también las condiciones para separar y el momento en qué se debe terminar la separación.

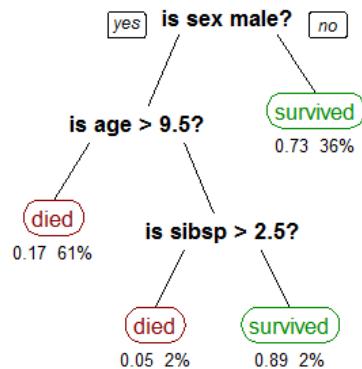


Figura 4.3. Ejemplo de Decision tree.

Fuente: Prashant Gupta (May. 2017). Decision Trees in Machine Learning.
<https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>

- *Random Forest*: Consiste de una gran cantidad de *decision trees*, que actúan de manera individual escupiendo cada uno una clase como resultado. La clase con mayor frecuencia entre los resultados se elige como el resultado total.

A ellos se suma la familia de algoritmos de *Deep Learning*.

Deep Learning

El *Deep Learning* es un subconjunto de *Machine Learning*. Es una familia de algoritmos inspirados por la estructura y funcionamiento del cerebro humano. Estos algoritmos pueden trabajar con enormes cantidades de datos estructurados y no estructurados.

La base del *deep learning* se encuentra en las redes neuronales artificiales (*Artificial Neural Networks - ANNs*). Estas arquitecturas fueron inspiradas por el procesamiento de la información y los nodos de comunicación distribuida en sistemas biológicos, descrito inicialmente en el libro "*Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*" de Frank Rosenblatt.

Las ANNs son básicamente colecciones de unidades interconectadas conocidas como neuronas artificiales, análogas a las neuronas biológicas del cerebro humano. Cada conexión (sinapsis) entre neuronas puede transmitir una señal de una neurona a otra. La neurona receptora puede procesar la señal y retransmitirla a neuronas subsecuentes. Además, las neuronas y sinapsis pueden tener pesos específicos, que incrementan o decrementan la señal emitida.

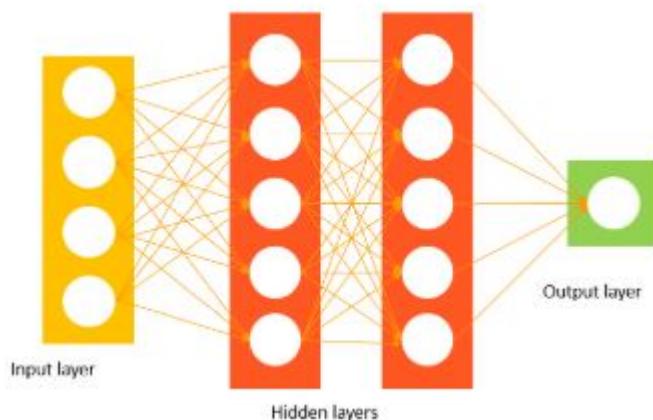


Figura 4.4. Esquema simple de una ANN.

Fuente: Shruti M. (Nov. 2022). *Discover the Differences Between AI vs. Machine Learning vs. Deep Learning*.

<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning#:~:text=Artificial%20Intelligence%20is%20the%20concept.algorithms%20to%20train%20a%20model>

Comúnmente, las neuronas se organizan en capas, una de entrada, una de salida y una o más entre ellas (conocidas como *hidden layers*). Cada capa puede llegar a tener hasta millones de neuronas, las cuales se encargan individualmente

de sumar todos los valores recibidos (teniendo en cuenta el peso de cada sinapsis), aplicar una función de activación (la cual es particular de cada neurona) y en base al resultado decidir si una señal debe ser emitida o no.

Al llegar la señal a la capa de salida, esta última da la información de una manera coherente para el problema que se desea solucionar con la red neuronal. La red utiliza una función de costo, la cual compara el resultado obtenido con el esperado, y en base a eso calcula el desempeño de la red. En todos los casos, independientemente de la función de costo utilizado, se intenta minimizar el error total de la red. Con los resultados de la función de costo, la información regresa hacia "atrás", y los pesos de las sinapsis se ajustan de manera tal que disminuyan el error total. Este proceso, que puede ocurrir múltiples veces, se conoce como *backpropagation*. Los pesos no se actualizan de forma "bruta" (intentando todas las combinaciones), sino que se hace usando una técnica llamada *stochastic gradient descent*. Este algoritmo hace uso del gradiente para encontrar el mínimo global de una función.

En general, hay dos clases de ANNs, las *feedforward networks* y las *feedback/recurrent networks*. Las *feedforward* transmiten la señal en un único sentido: hacia "adelante", es decir, desde la capa de entrada hasta la capa de salida, pasando por las capas entre ellas. En cambio, las *feedback* pueden tener caminos de retroalimentación, o dicho de otra manera, bucles que transmiten señales en ambos sentidos (hacia "adelante" y hacia "atrás").

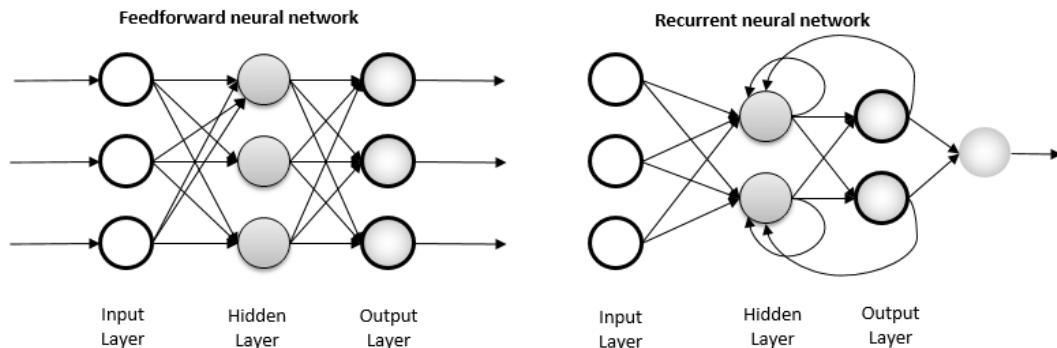


Figura 4.5. Diferencia entre redes feedforward y feedback.

Fuente: Engin Pekel (Mar. 2017). A COMPREHENSIVE REVIEW FOR ARTIFICAL NEURAL NETWORK APPLICATION TO PUBLIC TRANSPORTATION.

https://www.researchgate.net/publication/315111480_A_COMPREHENSIVE REVIEW_FOR_ARTIFI_CAL_NEURAL_NETWORK_APPLICATION_TO_PUBLIC_TRANSPORTATION

Las arquitecturas de Deep Learning han sido aplicadas para la resolución de problemas en diversas áreas. Algunos casos son:

- Filtrado en redes sociales
- Reconocimiento de imágenes y voz
- Detección de fraudes financieros
- Visión artificial
- Procesamiento de imágenes médicas
- Procesamiento de lenguajes naturales
- Procesamiento de artes visuales
- Diseño y descubrimiento de drogas

Convolutional Neural Network

Las Convolutional Neural Network (CNNs) son un tipo de algoritmo de Deep Learning utilizado mayormente para el análisis predictivo de imágenes, ya que fue inspirado por la organización de la corteza visual del cerebro de mamíferos. Una CNN es capaz de capturar las dependencias espaciales y temporales en una imagen. Su rol es simplificar las imágenes a una forma fácil de procesar, sin perder las características más críticas para obtener buenas predicciones.

Las CNNs están compuestas por una capa de entrada, una capa de salida y *hidden layers*.

Como primer paso, siguiendo a la capa de entrada, se encuentra una capa *convolutional*. Esta capa tiene como objetivo principal extraer características de alto nivel de la imagen de entrada. Para eso, se utiliza un filtro (matriz de pesos), el cual se desliza a través de los datos de entrada, moviéndose de a n posiciones/píxeles, definido por el *stride*. En cada posición de la imagen de entrada, se realiza la multiplicación matricial con el filtro, dando como resultado una nueva matriz de menor dimensión, llamado *feature map* o *activation map*. Las CNNs aprenden los valores de los filtros en el proceso de entrenamiento.

Otro paso de las CNNs son las capas ReLU (*Rectified Linear Unit*), las cuales aplican una función de activación a los *feature maps* para aumentar la no linealidad de la red. Las imágenes de por sí son no lineales, por lo cual es importante prevalecer esta propiedad en la red.

Para reducir el *overfitting* se hace uso de las capas *pooling*. Dichas capas convierten los *feature maps* en *pooled feature maps*, partiendo la imagen en grupos de áreas que no se solapan y tomando el valor máximo de cada área (*max pooling*), el promedio (*average pooling*) o la suma (*sum pooling*). Esto permite reconocer las características de la imagen, sin importar su ubicación en la imagen,

su sentido, su tamaño, su color, etc. Esto hace posible la flexibilidad espacial o *spatial variance*.

Luego de las tres capas definidas anteriormente, se hace uso de una capa *flattening*. Dicha capa simplemente aplana el *pooled feature map*, obteniendo como resultado un vector de números que podrá utilizarse como entrada de una red neuronal artificial (ANN) para continuar el procesamiento.

Por último, la CNN utiliza una capa *fully connected*, la cual recibe su nombre del hecho que conecta cada neurona de la capa anterior a cada neurona de la capa siguiente. Esta capa toma en cuenta los pesos de cada sinapsis (que se irán ajustando con *backpropagation*) para decidir el resultado final y compartirlo con la capa de salida.

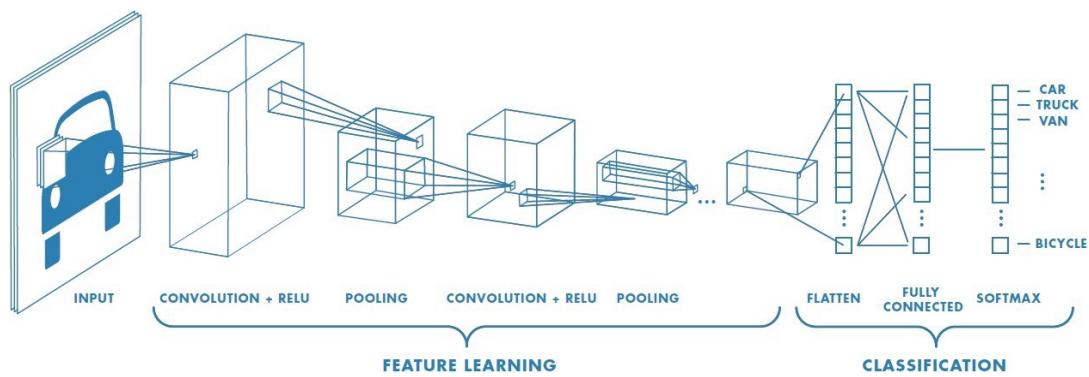


Figura 4.6. Estructura de una CNN.

Fuente: Sumit Saha (Dic. 2018). A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way.

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Para el caso de Songo, en el cual se debe analizar audio, es importante extraer características importantes del track, para poder lograr una mayor precisión a la hora de predecir el género.

Audio

Una señal de audio es una representación del sonido. Codifica toda la información necesaria para reproducir sonido. Viene en dos tipos básicos: analógica y digital.

La señal analógica se refiere al sonido representado utilizando métodos que replican la onda de sonido original, mientras que la señal digital es un conjunto de muestras de la onda sonora original tomadas a una velocidad determinada, según el *sampling rate*.

A nivel general, las características de una señal de audio se pueden categorizar según los siguientes aspectos:

- Nivel de abstracción: Pueden ser de alto nivel (entendidas y disfrutadas por humanos, como el ritmo, género o melodía), nivel medio (pueden ser percibidas por el humano, como el pitch, beat o MFCCs) o bajo nivel (sólo útil para las máquinas, como el zero crossing rate, spectral centroid o la energía).
- Alcance temporal: Pueden ser instantáneas (tomadas a partir de muestras menores a 10 milisegundos), a nivel de segmento (muestras de segundos) o globales (describen el sonido completo).
- Aspecto musical: Propiedades acústicas como el beat, ritmo, timbre, pitch, etc.
- Dominio de la señal: Pueden encontrarse en el dominio del tiempo, en el dominio de la frecuencia o en ambos.

- Enfoque de ML: Características usadas para modelos tradicionales de Machine Learning o para redes de Deep Learning.

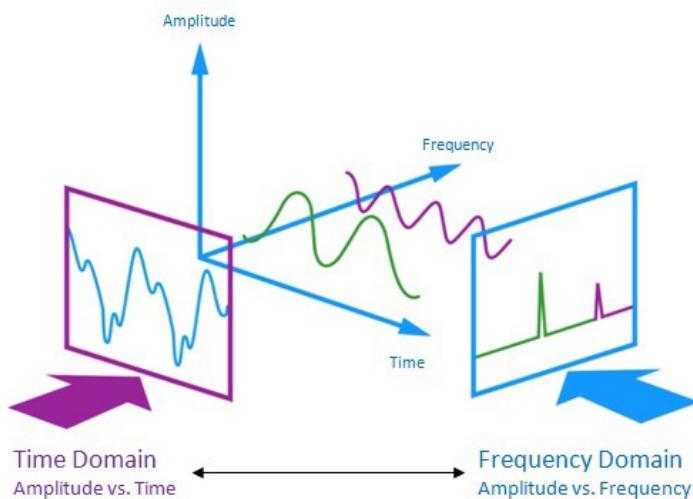


Figura 4.7. Señal de audio.
Fuente: Sanket Doshi (Dic. 2018). Music Feature Extraction in Python.
<https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d>

Transformada de Fourier

La transformada de Fourier, denominada así por Joseph Fourier, es una transformación matemática empleada para transformar señales entre el dominio del tiempo y el dominio de la frecuencia.

Transformada de Fourier de Tiempo Reducido

La Transformada de Fourier de Tiempo Reducido (Short-time Fourier Transform, STFT) es una secuencia de Transformadas de Fourier, que permite visualizar la variación de frecuencias con respecto al tiempo en una señal.

Root Mean Squared

La media cuadrática o *root mean squared* (RMS) es la magnitud total de una señal, es decir, el parámetro de energía o volumen de un audio.

Zero Crossing Rate

La Tasa de Cruces por Cero o Zero Crossing Rate (ZCR) es la tasa a la cual una señal cambia de positiva a negativa o viceversa, es decir, la cantidad de veces que cruza el valor cero.

Spectral Centroid

El Spectral Centroid indica dónde se encuentra el centro de masa de un sonido, y se calcula como la media ponderada de las frecuencias presentes en dicho sonido.

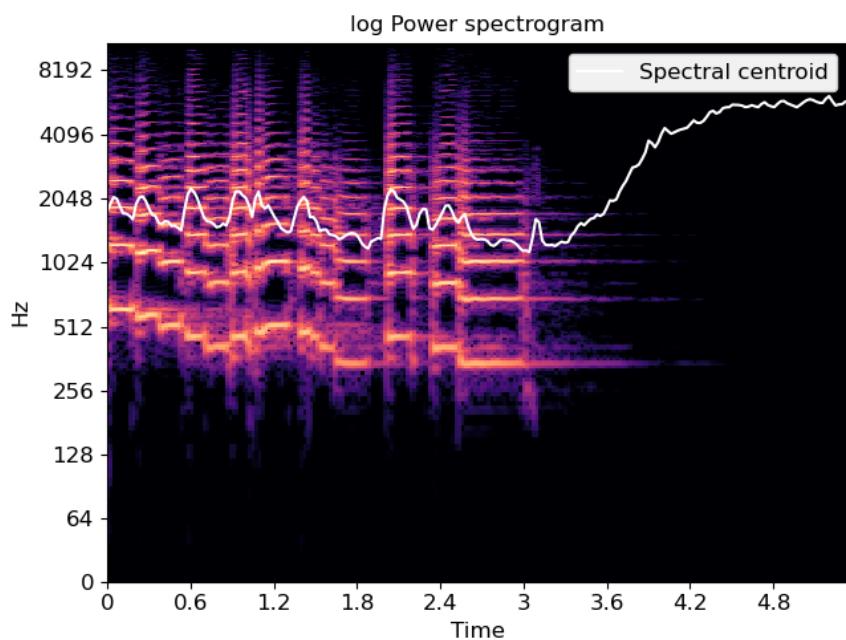


Figura 4.8. Representación del Spectral Centroid.
Fuente: Saranga Mahanta (May. 2021). *Audio Feature Extraction*.
<https://devopedia.org/audio-feature-extraction>

Spectral bandwidth

El *Spectral Bandwidth* deriva del *Spectral Centroid*, y es básicamente un rango de interés cercano a dicho centro de masa, en otras palabras, la varianza del *Spectral Centroid*.

Spectral Rolloff

El *Spectral Rolloff* es la frecuencia por debajo de la cual se encuentra un porcentaje específico de la energía espectral total (por ejemplo el 85%).

Chroma Feature

La visualización de las *Chroma Features* permite saber cuán dominante son las características de un tono específico.

Tempo

El *tempo* es la velocidad de una pieza de audio, generalmente medida en bits por minuto (*beats per minute - BPM*).

Tempogram

Un *Tempogram* es una representación visual del *tempo* de una pieza de audio con respecto al tiempo.

Spectrogram

Un *spectrogram* es una representación visual del espectro de frecuencias de una señal según varía con el tiempo. Se obtiene aplicando la Transformada de Fourier de Tiempo Reducido sobre la señal.

Mel Scale

La Escala de Mel o *Mel Scale* es una transformación logarítmica de la frecuencia de una señal. La idea central de esta transformación es que sonidos a igual distancia en la escala son percibidos a igual distancia por humanos.

Por lógica, podría decirse que los humanos perciben la diferencia entre 100hz y 200hz igual que como perciben la diferencia entre 1000hz y 1100hz. Esto no es así en la realidad, ya que es muy difícil para el oído humano diferenciar entre frecuencias altas, no así ocurre lo mismo con frecuencias bajas. Es por ello que la escala de Mel es fundamental a la hora de analizar audio, ya que permite extraer información del audio de la misma manera en que lo haría el oído humano.

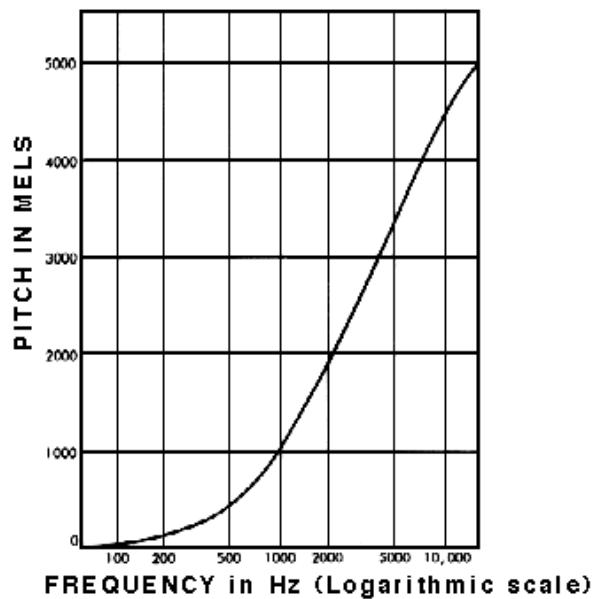


Figura 4.9. Escala mel como función de la frecuencia.

Fuente: Leland Roberts (Mar. 2020). *Understanding the Mel Spectrogram*.
[https://medium.com-analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53](https://medium.com.analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53)

Mel Spectrogram

Un *Mel Spectrogram* es simplemente un *Spectrogram* representado en la escala de Mel, en oposición a la escala de frecuencias tradicional.

Mel Frequency Cepstral Coefficients

Los *Mel Frequency Cepstral Coefficients* (MFCCs) son coeficientes para la representación del habla basados en la percepción auditiva humana. Comúnmente, se calculan de la siguiente forma:

1. Separar la señal en pequeños tramos.
2. A cada tramo aplicarle la Transformada de Fourier discreta y obtener la potencia espectral de la señal.
3. 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.
4. Tomar el logaritmo de todas las energías de cada frecuencia mel
5. Aplicarle la transformada de coseno discreta a estos logaritmos.

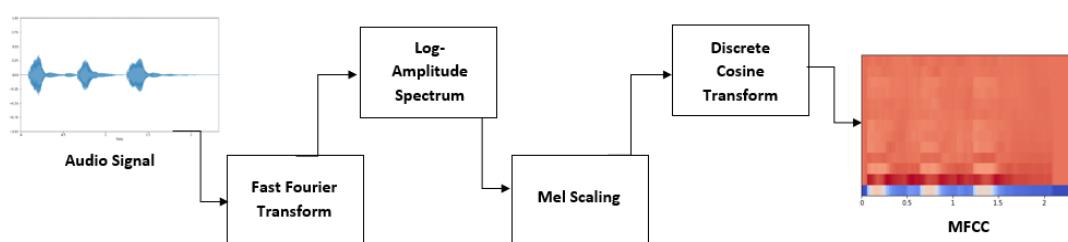


Figura 4.10. Pasos para extraer los MFCC de una señal de audio.
 Fuente: Saranga Mahanta (May. 2021). *Audio Feature Extraction*.
<https://devopedia.org/audio-feature-extraction>

5 - La aplicación

Songo es una aplicación de escritorio, esto quiere decir que no está disponible desde la web ya que es necesario, por la naturaleza de su funcionalidad, tener el control sobre el sistema de archivos de la computadora del usuario. La aplicación consta de 2 funcionalidades principales y otras quality of life implementadas para aprovechar más y hacer más eficientes las funcionalidades principales.

En primer lugar, la funcionalidad más core y básica de la aplicación es la clasificación de un track en los subgéneros de la música electrónica, en especial de Techno (la clasificación se hará con los siguientes géneros: Deep House, Tech House, Melodic Techno, Progressive, Techno Peak Time, Hard Techno, Minimal y Trance). Esta funcionalidad toma como input un path del sistema de archivos que apunta al archivo el cual se quiera clasificar. Mediante el modelo de clasificación entrenado, se obtiene como output el género al cual pertenece ese track y se muestra la información al usuario. Además de esto se guarda en un servidor dicha información para ser reutilizada en el futuro.

En segundo lugar, la aplicación cuenta con una funcionalidad apuntada a simplificar y por ende hacer más eficiente el proceso de clasificación de música que realizan los personajes mencionados. Esta funcionalidad toma como input un directorio seleccionado por el usuario donde se encuentren los archivos de música a clasificar, la aplicación ejecuta los scripts de clasificación sobre el directorio.

El código fuente del proyecto y de sus partes se encuentra en diferentes repositorios de Github, con instrucciones para correr dichas partes y realizar pruebas de las funcionalidades.

6 - Pruebas y validaciones

El éxito de la app Songo depende mayormente de su modelo predictivo, el cual determina el género de cada track a analizar y así permite reubicarlo en la carpeta correspondiente.

Teniendo en cuenta la infinidad de soluciones posibles a la hora de resolver un problema de clasificación, es importante comenzar desde lo más simple a lo más complejo. Por ello, los primeros modelos propuestos deben ser rápidos de construir, entrenar y correr, además de fáciles de entender.

Para la implementación de los primeros modelos, los cuales son básicos de *Machine Learning*, se utiliza la librería scikit-learn de Python. Los archivos de audio se separan en cuatro *samples* de treinta segundos cada uno, equiespaciados. Se analizan las siguientes características de cada *sample*: *Chroma feature*, *Root Mean Squared Energy*, *Spectral Centroid*, *Spectral Bandwidth*, *Spectral Rolloff*, *Zero Crossing Rate*, *Tempo* y *MFCCs*. La siguiente tabla describe la separación de los datos para entrenamiento y validación.

| | |
|------------------------------|------|
| Cantidad de géneros | 8 |
| Tracks por género | 100 |
| Tracks totales | 800 |
| Muestras por track | 4 |
| Muestras totales | 3200 |
| Porcentaje para train | 0.8 |
| Porcentaje para test | 0.2 |
| Muestras para train | 2560 |
| Muestras para test | 640 |

Tabla 6.1. Distribución de los datos.

Fuente: Elaboración propia.

Es importante aclarar en esta instancia de donde provienen los datos. Para la construcción del dataset se utilizaron diferentes fuentes. Estas fuentes son diversos sitios de música que son avalados por la totalidad de la comunidad de DJs y productores de música electrónica. El más utilizado es Muzbase, un sitio de descarga de música en calidad profesional, extensiones .AIFF y .WAV, previamente clasificados por subgéneros. Además de este, se utilizó Beatport, el sitio que ránkea los tracks globalmente por subgénero de música, y el que todos los productores y DJs utilizan para publicar sus tracks y comprar música. También se usaron otros como Soundcloud o Tidal, que son sitios más enfocados en la reproducción que en la venta de tracks, pero que igualmente se pueden conseguir algunos que los productores deciden dar al público de manera gratuita.

Esto es importante aclararlo ya que la construcción del dataset no fue hecha de una manera subjetiva, si no que se utilizaron diferentes fuentes que son avaladas y usadas por la comunidad. Por lo tanto la asignación del subgénero al cual pertenecen cada track no fue hecha a base de opiniones personales de quienes desarrollaron el modelo, si no que fueron asignados a base de datos obtenidos de los sitios más usados de música electrónica.

El primer modelo utiliza el algoritmo *Logistic Regression*. La precisión para este modelo es del 38.9%.

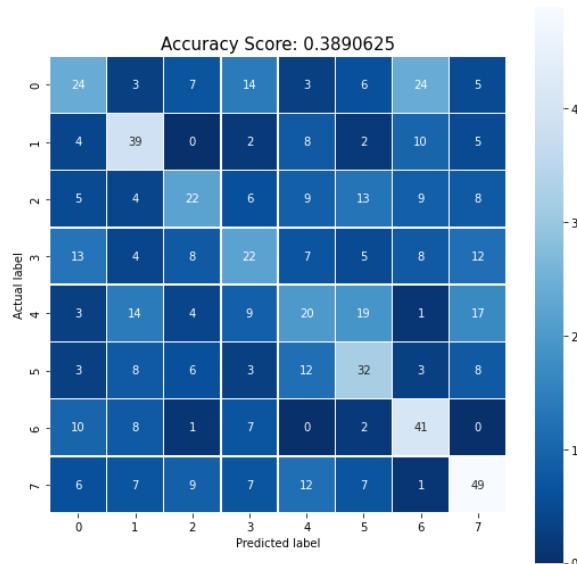


Figura 6.1. Matriz de confusión del algoritmo *Logistic Regression*.
Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0 | 0.35 | 0.28 | 0.31 | 86 |
| 1 | 0.45 | 0.56 | 0.5 | 70 |
| 2 | 0.39 | 0.29 | 0.33 | 76 |
| 3 | 0.31 | 0.28 | 0.3 | 79 |
| 4 | 0.28 | 0.23 | 0.25 | 87 |
| 5 | 0.37 | 0.43 | 0.4 | 75 |
| 6 | 0.42 | 0.59 | 0.49 | 69 |
| 7 | 0.47 | 0.5 | 0.49 | 98 |
| accuracy | | | 0.39 | 640 |
| macro avg | 0.38 | 0.39 | 0.38 | 640 |
| weighted avg | 0.38 | 0.39 | 0.38 | 640 |

Tabla 6.2. Reporte de *sklearn* para Logistic Regression.

Fuente: Elaboración propia.

El segundo modelo utiliza el algoritmo Support Vector Machines. La precisión para este modelo es del 46.8%.

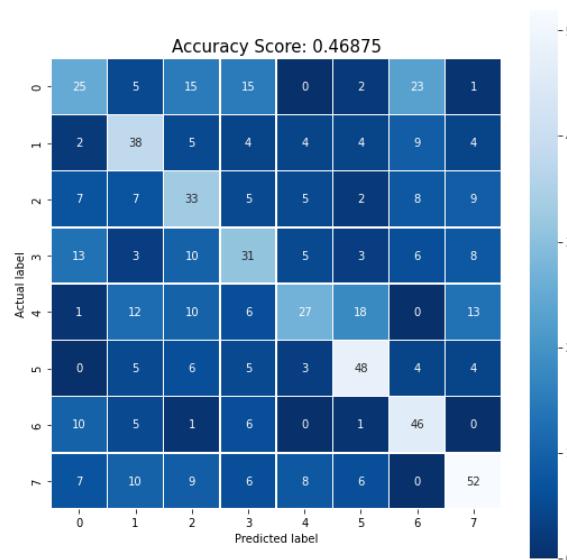


Figura 6.2. Matriz de confusión del algoritmo Support Vector Machines.

Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|----------|-----------|--------|----------|---------|
| 0 | 0.38 | 0.29 | 0.33 | 86 |
| 1 | 0.45 | 0.54 | 0.49 | 70 |
| 2 | 0.37 | 0.43 | 0.4 | 76 |
| 3 | 0.4 | 0.39 | 0.39 | 79 |
| 4 | 0.52 | 0.31 | 0.39 | 87 |
| 5 | 0.57 | 0.64 | 0.6 | 75 |
| 6 | 0.48 | 0.67 | 0.56 | 69 |
| 7 | 0.57 | 0.53 | 0.55 | 98 |

| | | | | |
|---------------------|------|------|------|-----|
| | | | | |
| accuracy | | | 0.47 | 640 |
| macro avg | 0.47 | 0.48 | 0.46 | 640 |
| weighted avg | 0.47 | 0.47 | 0.46 | 640 |

Tabla 6.3. Reporte de *sklearn* para Support Vector Machines.

Fuente: Elaboración propia.

El tercer modelo utiliza el algoritmo K-Nearest Neighbors, con K igual a tres. La precisión para este modelo es del 42.6%.

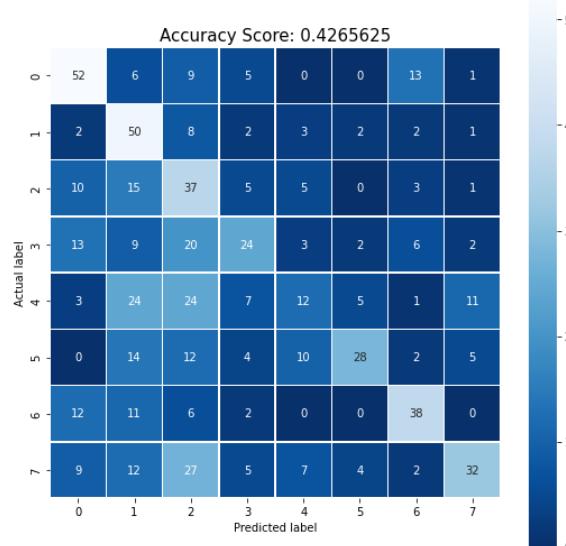


Figura 6.3. Matriz de confusión del algoritmo K-Nearest Neighbors.
Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0 | 0.51 | 0.6 | 0.56 | 86 |
| 1 | 0.35 | 0.71 | 0.47 | 70 |
| 2 | 0.26 | 0.49 | 0.34 | 76 |
| 3 | 0.44 | 0.3 | 0.36 | 79 |
| 4 | 0.3 | 0.14 | 0.19 | 87 |
| 5 | 0.68 | 0.37 | 0.48 | 75 |
| 6 | 0.57 | 0.55 | 0.56 | 69 |
| 7 | 0.6 | 0.33 | 0.42 | 98 |
| | | | | |
| accuracy | | | 0.43 | 640 |
| macro avg | 0.47 | 0.44 | 0.42 | 640 |
| weighted avg | 0.47 | 0.43 | 0.42 | 640 |

Tabla 6.4. Reporte de *sklearn* para K-Nearest Neighbors.

Fuente: Elaboración propia.

El cuarto modelo utiliza el algoritmo Decision Tree. La precisión para este modelo es del 33.5%.

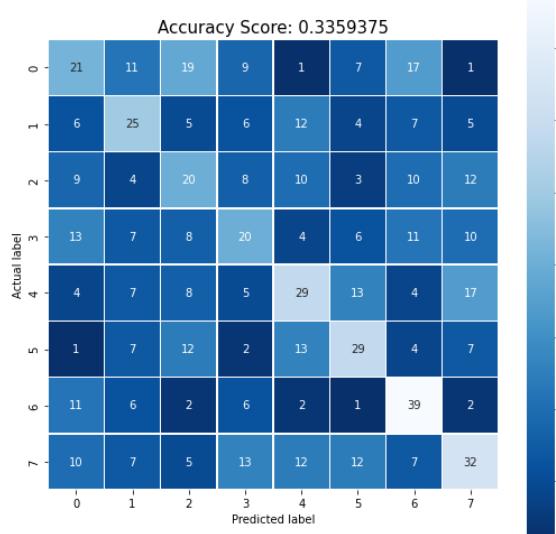


Figura 6.4. Matriz de confusión del algoritmo *Decision Tree*.
Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0 | 0.28 | 0.24 | 0.26 | 86 |
| 1 | 0.34 | 0.36 | 0.35 | 70 |
| 2 | 0.25 | 0.26 | 0.26 | 76 |
| 3 | 0.29 | 0.25 | 0.27 | 79 |
| 4 | 0.35 | 0.33 | 0.34 | 87 |
| 5 | 0.39 | 0.39 | 0.39 | 75 |
| 6 | 0.39 | 0.57 | 0.46 | 69 |
| 7 | 0.37 | 0.33 | 0.35 | 98 |
| | | | | |
| accuracy | | | 0.34 | 640 |
| macro avg | 0.33 | 0.34 | 0.33 | 640 |
| weighted avg | 0.33 | 0.34 | 0.33 | 640 |

Tabla 6.5. Reporte de *sklearn* para *Decision Tree*.
Fuente: Elaboración propia.

El quinto modelo utiliza el algoritmo Random Forest. La precisión para este modelo es del 38.9%.

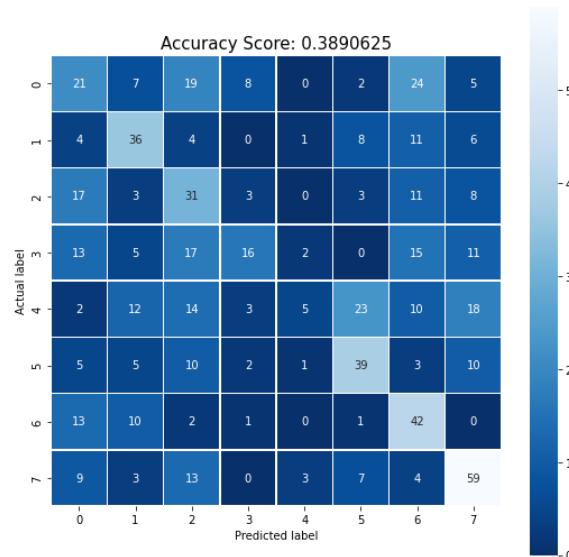


Figura 6.5. Matriz de confusión del algoritmo Random Forest.
Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0 | 0.25 | 0.24 | 0.25 | 86 |
| 1 | 0.44 | 0.51 | 0.48 | 70 |
| 2 | 0.28 | 0.41 | 0.33 | 76 |
| 3 | 0.48 | 0.2 | 0.29 | 79 |
| 4 | 0.42 | 0.06 | 0.1 | 87 |
| 5 | 0.47 | 0.52 | 0.49 | 75 |
| 6 | 0.35 | 0.61 | 0.44 | 69 |
| 7 | 0.5 | 0.6 | 0.55 | 98 |
| accuracy | | | 0.39 | 640 |
| macro avg | 0.4 | 0.39 | 0.37 | 640 |
| weighted avg | 0.4 | 0.39 | 0.36 | 640 |

Tabla 6.6. Reporte de *sklearn* para Random Forest.

Fuente: Elaboración propia.

Definitivamente, los resultados al usar modelos básicos de Machine Learning no son buenos. A pesar de que el problema a resolver es muy complejo, y obtener un buen resultado de precisión es difícil por la amplia similitud que existe entre los distintos subgéneros de techno, es posible mejorarla tomando como base la precisión más alta de los modelos anteriores, el SVM. Para esto, se emplea un modelo de Deep Learning, el cual requiere de un grado de entendimiento más alto del problema y de los conceptos que lo rodean.

El modelo propuesto se basa en una Convolutional Neural Network. Para ello, es necesario usar características del audio que asimilen una imagen.

Para lograr esto, se utilizan los primeros trece coeficientes MFCC de cada frame. El sample rate de los tracks para entrenamiento es de 22050 Hz. Esto

quiere decir que por cada segundo de audio, se toman 22050 muestras o *samples*. Por defecto, en la librería de audio que se usa para analizar la música (librosa), el *hop length* es de 512, es decir que cada *frame* sobre el cual se calculan los coeficientes MFCC contiene 512 *samples*. Es decir, para los treinta segundos de audio de cada *track* se calculan 13 MFCC de 1292 *frames*.

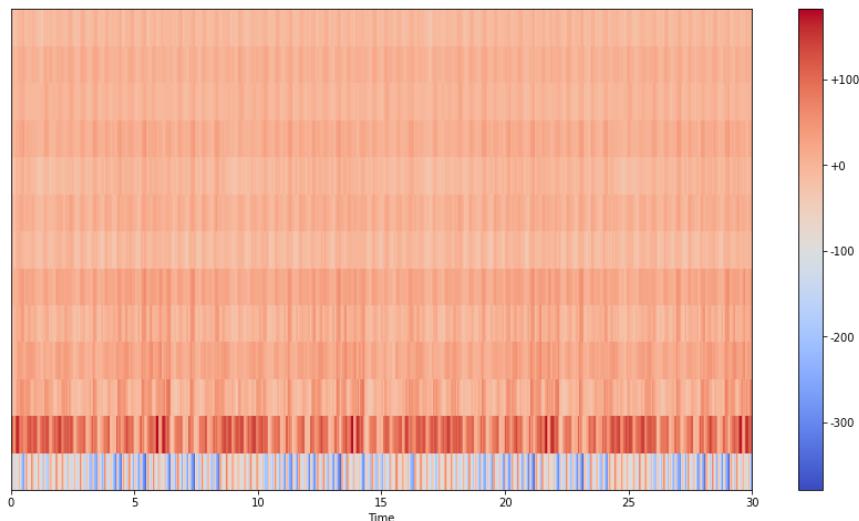


Figura 6.6. MFCC de 13 bandas para un *sample* de 30 segundos de deep house.
Fuente: Elaboración propia

El resultado del análisis de audio haciendo uso de los coeficientes MFCC es una matriz 13x1292 por cada *track*. Esto puede interpretarse como una imagen 2D. De hecho, es lo que se hace para que la red neuronal convolucional funcione adecuadamente.

La CNN tiene tres capas convolutional, cada una de las cuales usa la función de activación ReLU y es seguida por una capa de max pooling y una capa

de *batch normalization*. La secuencia es continuada por una capa *dropout*, la cual simplemente actúa como máscara, invalidando el aporte de algunas neuronas y dejando el resto sin modificar. Esto previene el *overfitting*. El modelo finaliza con una capa *flatten* y dos capas *dense*, la primera con activación ReLU y la última con activación softmax.

El entrenamiento es realizado con lotes de tamaño treinta y dos, es decir se analizan treinta y dos *samples* de treinta segundos antes de actualizar los pesos del modelo. Además, se hacen treinta *epochs*, lo que quiere decir que se recorre el dataset de entrenamiento treinta veces por completo. En cada epoch, la precisión debería aumentar y el error debería disminuir.

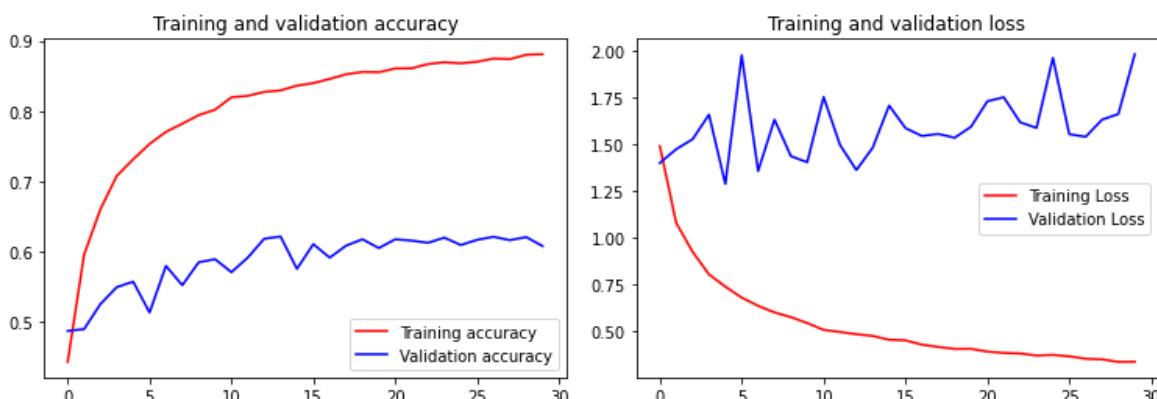


Figura 6.7. Medición de la precisión y el error de validación en función del número de epoch.
Fuente: Elaboración propia.

La precisión total para esta red neuronal es del 72%, significativamente mayor que los métodos iniciales de Machine Learning.

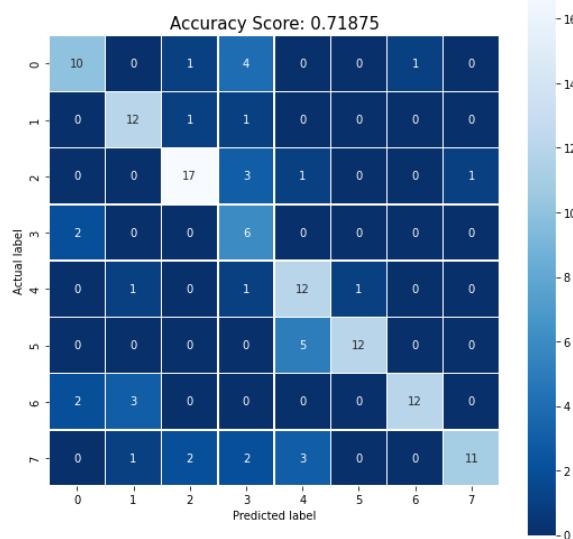


Figura 6.8. Matriz de confusión para la CNN propuesta.
Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0 | 0.71 | 0.62 | 0.67 | 16 |
| 1 | 0.71 | 0.86 | 0.77 | 14 |
| 2 | 0.81 | 0.77 | 0.79 | 22 |
| 3 | 0.35 | 0.75 | 0.48 | 8 |
| 4 | 0.57 | 0.8 | 0.67 | 15 |
| 5 | 0.92 | 0.71 | 0.8 | 17 |
| 6 | 0.92 | 0.71 | 0.8 | 17 |
| 7 | 0.92 | 0.58 | 0.71 | 19 |
| accuracy | | | 0.72 | 128 |
| macro avg | 0.74 | 0.72 | 0.71 | 128 |
| weighted avg | 0.78 | 0.72 | 0.73 | 128 |

Tabla 6.7. Reporte de *sklearn* para CNN.
Fuente: Elaboración propia.

Como métrica alternativa, se toma la precisión del modelo tomando los dos primeros géneros predecidos, es decir, los dos géneros que la red considera más probables. En este caso, el resultado es de 88.2%.

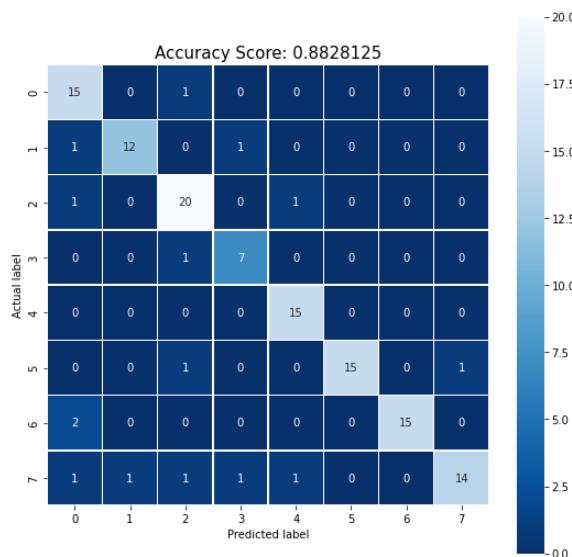


Figura 6.9. Matriz de confusión para la CNN propuesta, utilizando la métrica alternativa.
Fuente: Elaboración propia.

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0 | 0.75 | 0.94 | 0.83 | 16 |
| 1 | 0.92 | 0.86 | 0.89 | 14 |
| 2 | 0.83 | 0.91 | 0.87 | 22 |
| 3 | 0.78 | 0.88 | 0.82 | 8 |
| 4 | 0.88 | 1 | 0.94 | 15 |
| 5 | 1 | 0.88 | 0.94 | 17 |
| 6 | 1 | 0.88 | 0.94 | 17 |
| 7 | 0.93 | 0.74 | 0.82 | 19 |
| accuracy | | | 0.88 | 128 |
| macro avg | 0.89 | 0.89 | 0.88 | 128 |
| weighted avg | 0.89 | 0.88 | 0.88 | 128 |

Tabla 6.8. Reporte de *sklearn* para CNN, utilizando la métrica alternativa.

Fuente: Elaboración propia.

Los valores 0-7 en los distintos reportes corresponden a los géneros, en el siguiente orden: *Deep House, Tech House, Melodic Techno, Progressive, Techno Peak Time, Hard Techno, Minimal, Trance*.

Resumen de experimentos:

| | Algoritmo | Average Precision | Average Recall | Average F1-score |
|--|-------------------------------|-------------------|----------------|------------------|
| Basados en features | Logistic Regression | 0.38 | 0.39 | 0.38 |
| | Support Vector Machine | 0.47 | 0.48 | 0.46 |
| | K-nearest neighbors | 0.47 | 0.44 | 0.42 |
| | Decision tree | 0.33 | 0.34 | 0.33 |
| | Random Forest | 0.4 | 0.39 | 0.37 |
| Basado en imágenes de espectrograma | CNN | 0.74 | 0.72 | 0.71 |

Tabla 6.9. Resumen de las métricas clave de los distintos algoritmos con los que se experimentó.

Fuente: Elaboración propia.

7 - Tecnologías

La aplicación cuenta con diferentes partes, y para cada una de ellas se usaron tecnologías específicas. A continuación, se detallan dichas partes y las tecnologías de las mismas.

7.1 - Front End

React

React es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario. Es mantenido por Facebook y la comunidad de software libre. En el proyecto hay más de mil desarrolladores libres. Intenta ayudar a los desarrolladores a construir aplicaciones que usan datos que cambian todo el tiempo. Su objetivo es ser sencillo, declarativo y fácil de combinar. React sólo maneja la interfaz de usuario en una aplicación, React es la Vista en un contexto en el que se use el patrón MVC (Modelo-Vista-Controlador) o MVVM (Modelo-vista-modelo de vista). También puede ser utilizado con las extensiones de React-based que se encargan de las partes no-UI (que no forman parte de la interfaz de usuario) de una aplicación web.

Electron

Electron es un framework de código abierto creado por Cheng Zhao, y ahora desarrollado por GitHub. Permite el desarrollo de aplicaciones gráficas de escritorio usando componentes del lado del cliente y del servidor originalmente desarrolladas para aplicaciones web: Node.js del lado del servidor y Chromium como interfaz. Debido a esto, permite el uso de diferentes tipos de frameworks de

desarrollo web, como es el caso de React en ese proyecto. Electron es el framework gráfico detrás de muchos proyectos de código abierto importantes, incluyendo a Atom de GitHub y Microsoft Visual Studio Code, la aplicación de escritorio del servicio de streaming Tidal y el IDE Light Table, al igual que el cliente de escritorio freeware del servicio de mensajería instantánea Discord.

7.2 - Back End

Node

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl.

Express

Express es un entorno de trabajo para aplicaciones web para Node.js, de código abierto y con licencia MIT. Se utiliza para desarrollar aplicaciones web y APIs. El autor original es TJ Holowaychuk y la primera versión se lanzó el 2010. Express.js forma parte de MEAN, juntamente con MongoDB, Angular.js y Node.js.

Postgresql

PostgreSQL, también llamado Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL, similar a la BSD o la MIT.

7.3 - Inteligencia Artificial

Python

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Librosa

Librosa es un paquete de python para análisis de música y audio. Proporciona los componentes básicos necesarios para crear sistemas de recuperación de información musical.

TensorFlow

TensorFlow es una biblioteca de código abierto para aprendizaje automático, fue desarrollada por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y

descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Actualmente es utilizado tanto en la investigación como en los productos de Google, frecuentemente reemplazando el rol de su predecesor de código cerrado, DistBelief. TensorFlow fue originalmente desarrollado por el equipo de Google Brain para uso interno en Google antes de ser publicado bajo la licencia de código abierto Apache 2.0 el 9 de noviembre de 2015.

Scikit-Learn

Scikit-learn es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python. Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.

Keras

Keras es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano. Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje Profundo. Sus fuertes se centran en ser amigable para el usuario, modular y extensible. Inicialmente fue desarrollada como parte de los esfuerzos de investigación del proyecto ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).

Para el desarrollo del módulo de inteligencia artificial fueron utilizadas más librerías para cuestiones como el manejo y procesamiento de datos, e interacciones que era necesario realizar con la computadora del usuario.

8 - Arquitectura

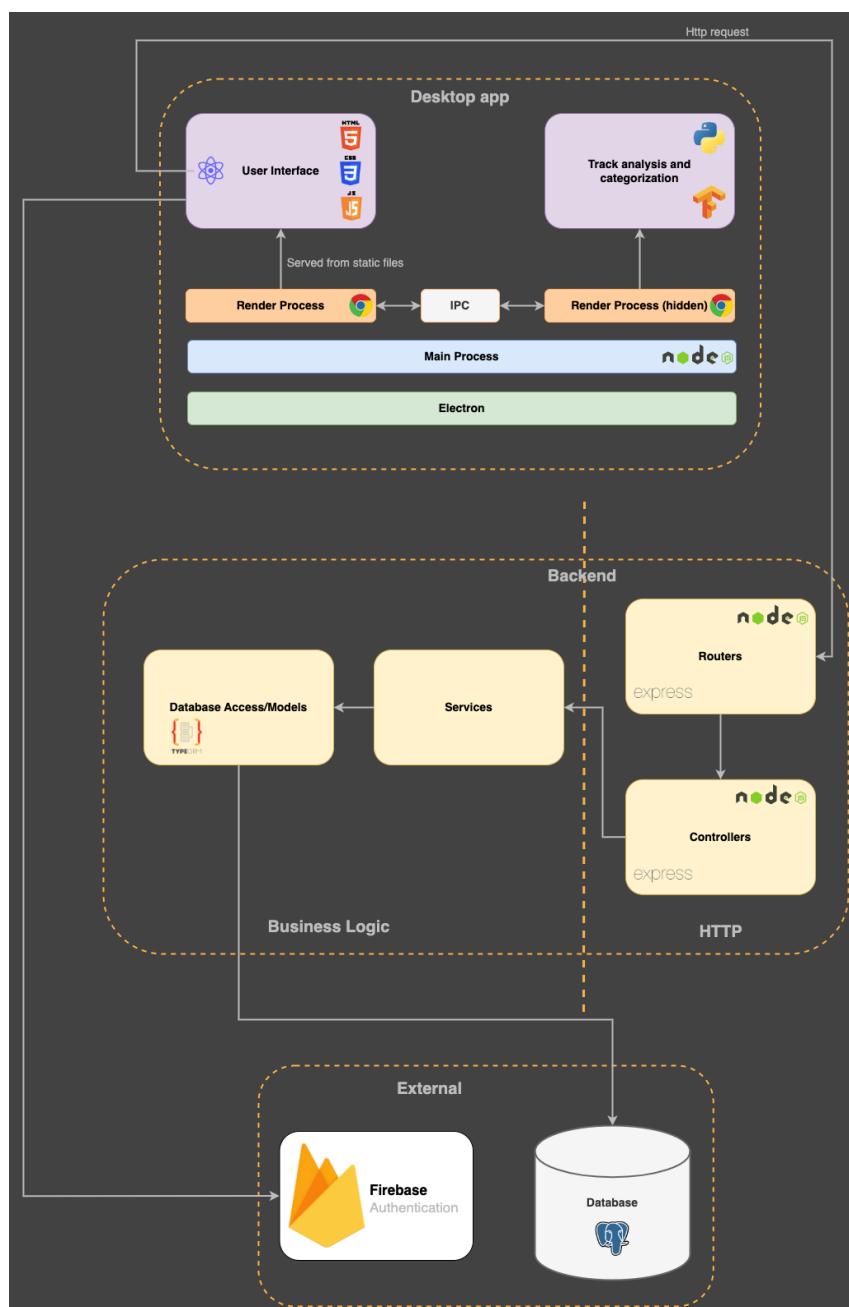


Figura 8.1. Arquitectura de Songo.
Fuente: Elaboración propia.

La aplicación se divide en tres componentes principales: La aplicación de escritorio, el *backend* y el módulo externo. Cada uno de ellos cumple un rol único y fundamental para el correcto funcionamiento de Songo.

Comenzando por la aplicación de escritorio, la misma está basada en el framework Electron. Dicho entorno de trabajo permite utilizar las tecnologías comunes de desarrollo web para aplicaciones de escritorio, sin la necesidad de escribir código nativo. Además, permite la integración con herramientas como React y Angular.

El proceso *main* se encarga de correr la lógica de la aplicación y a su vez organizar el ciclo de vida de la misma. Además, crea procesos *renderer*, a través de la instancia de ventanas o *Browser Windows*. El proceso *main* actúa como un *backend*, y está construido sobre el entorno de Node.

En Songo, existen dos procesos *renderer*. Uno de ellos correrá siempre que la aplicación esté abierta, el proceso visible que se encarga de manejar la interfaz de usuario. El mismo hace uso de la librería React, y de las tecnologías básicas Javascript, CSS y HTML.

Por otro lado, existe un proceso *renderer* secundario que corre solamente cuando el usuario desea analizar *tracks* de su directorio de archivos. Dicho proceso crea una ventana invisible para el usuario. Esta ventana hace uso de python-shell para correr los tres scripts de Python necesarios para el análisis predictivo.

Para lograr el flujo completo, se hace uso del módulo IPC (*Inter-Process Communication*) introducido por Electron. Como su nombre en inglés lo indica, el módulo permite la comunicación entre los procesos *renderer* a través de mensajes. Estos mensajes, opcionalmente, pueden contener información.

El flujo completo de análisis predictivo del género de los *tracks* dentro de un directorio es el siguiente:

1. El proceso *renderer* principal recibe un evento (click del botón “Clasificar” por parte del usuario).
2. El proceso *main* crea una nueva instancia invisible de *BrowserWindow*.
3. Dicha ventana comienza a correr el primer *script*, el cual analiza los metadatos de los *tracks*.
4. Al finalizar, el proceso *renderer* secundario envía un mensaje al proceso principal (UI), incluyendo el resultado del análisis de metadatos.
5. El proceso principal realiza una *request* HTTP al backend con la información recibida del *renderer* secundario, con el objetivo de encontrar *tracks* similares que ya han sido analizados por otro usuario (o por el mismo).
6. El resultado de la *request* será una lista con todos los *tracks* para los cuales se les encontró un *match*. El proceso principal envía un mensaje con esta lista al secundario.
7. El proceso secundario corre un segundo *script*, el cual simplemente ordena los *tracks* de la lista en las carpetas correspondientes a su género. Ej. si un *track* fue predecido como *melodic_techno* previamente, el *script* moverá dicho *track* a la carpeta *melodic_techno* (si no existe, la creará).

8. El proceso secundario envía un mensaje al principal, con el objetivo de anunciar que ya finalizó el proceso del segundo script.
9. El proceso principal envía un mensaje al secundario, sin ninguna información adicional.
10. El proceso secundario corre el tercer y último script, el cual predice el género de todos los tracks que no han sido analizados previamente, haciendo uso del modelo predictivo ya guardado. Una vez hechas las predicciones, mueve los archivos de música a las carpetas correspondientes, al igual que lo hace el segundo script.
11. Al finalizar, el proceso secundario envía un mensaje al principal, incluyendo la lista de tracks con su género predicho.
12. El proceso principal envía una request HTTP al backend con el resultado de las predicciones, de modo tal que el backend guarde esta información para ser utilizada en un futuro por el mismo usuario u otros (paso 5).

Hasta aquí, el backend funciona como un sistema *black box*. Internamente, este componente de la aplicación se divide en tres capas como cualquier backend común en node.

La primera capa contiene las rutas y el controlador. En conjunto, se encargan de recibir requests HTTP y delegarlas al servicio correspondiente.

La segunda capa contiene todos los servicios específicos de la aplicación. Cada uno de los servicios contiene la lógica de negocios para un propósito específico. Por ejemplo, el servicio *Tracks* contiene la lógica para el *matching* entre archivos musicales.

La tercera capa contiene los modelos de datos. Se hace uso de TypeORM para simplificar el manejo de entidades y sus relaciones en la base de datos. Esta capa no contiene lógica de negocios, simplemente define la información que será guardada en la base de datos y la manera de acceder a ella.

Por último, el componente externo de la aplicación comprende la base de datos y todos los servicios externos. La base de datos utiliza el sistema de manejo de bases de datos relacionales PostgreSQL. TypeORM se encarga de la comunicación con PostgreSQL a la hora de crear, actualizar, leer o eliminar datos. Por otro lado, Songo utiliza Firebase Authentication para el registro, login y logout de usuarios, el cual es invocado a través de un SDK directamente desde la aplicación de escritorio.

9 - Modelo de datos

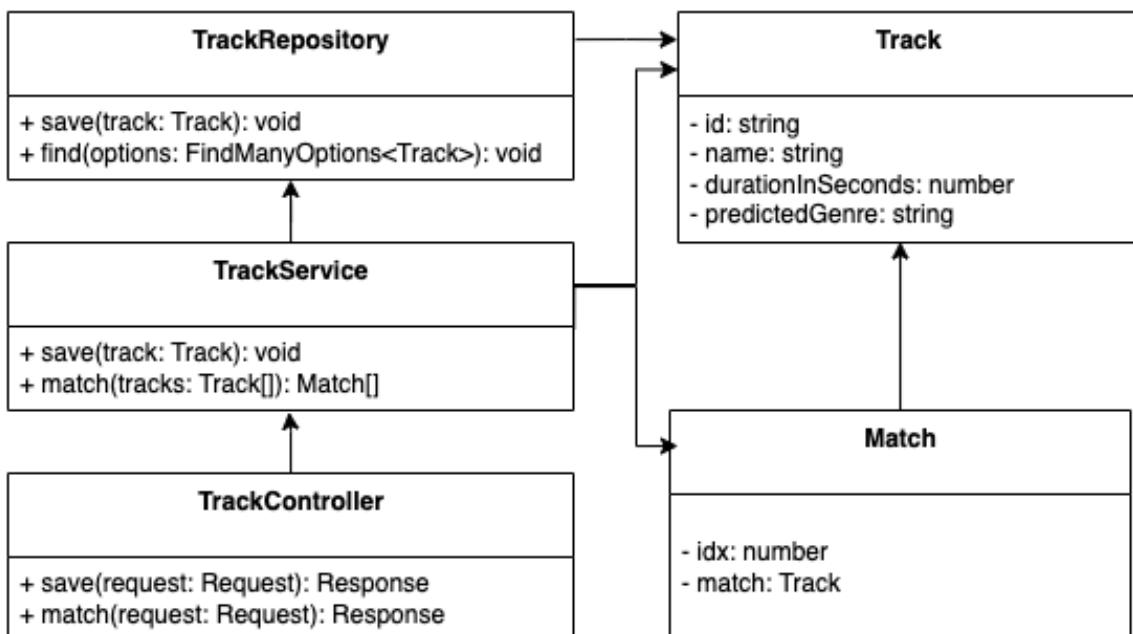


Figura 9.1. Modelo de datos de Songo.

Fuente: Elaboración propia.

El modelo de datos de la plataforma es muy simple. Simplemente, necesita soportar el guardado y *match* de los tracks según sus metadatos.

La primera clase es **TrackController**, la cual tiene solamente dos métodos: *save* y *match*. El método *save* recibe una *request* HTTP, la cual contiene un **Track** en su *body*, y retorna una *response* HTTP sin ninguna información adicional. El segundo método, *find*, recibe una *request* HTTP, la cual contiene una lista de **Track** en su *body* y retorna una *response* HTTP con una lista de **Match** en su *body*. Las clases **Request** y **Response** son propias del framework Express.

TrackController hace uso de *TrackService*, que contiene la lógica de negocio. Los métodos coinciden en nombre con los del *TrackController*, pero difieren en que no dependen de *Request* y *Response*, es decir, están aislados de la capa HTTP del sistema.

A su vez, *TrackService* usa *TrackRepository*, de modo tal que esta última clase sea la única consciente de la capa de datos del sistema. Esta última clase tiene dos métodos: *save* y *find*. *Save*, como lo indica su nombre, se encarga del guardado de *tracks* en la base de datos. Al igual, *find* es responsable de encontrar *tracks* en la base de datos, en base a ciertas restricciones definidas por *options*. El parámetro *options* es del tipo *FindManyOptions<Track>* y está definido por TypeORM.

La clase *Track* contiene cuatro propiedades:

- *id*: Propiedad del tipo *string*, sirve como identificador para el *track*.
- *name*: Propiedad del tipo *string*. Nombre del *track*, el cual será utilizado para el *match* con otros *tracks*.
- *durationInSeconds*: Propiedad del tipo *number*. Duración del *track* en segundos, la cual será utilizada para el *match* con otros *tracks*.
- *predictedGenre*: Propiedad del tipo *string*. Género predecido del *track*, el cual será utilizado por la aplicación para ordenar un archivo local en caso de hacer un *match*.

En cuanto a la clase *Match*, esta tiene sólo dos propiedades:

- *idx*: Propiedad del tipo *number*. El índice del *track* que encontró un *match* en la lista de entrada.

- *match*: Track guardado con el cual hizo match el track de la lista de entrada.

Notar que la clase Match no contiene la propiedad *id*, ya que no se guarda en la base de datos, simplemente se usa como respuesta al método *match*.

10 - Usuarios y funcionalidades

10.1 - Tipos de usuarios

Songo es una aplicación diseñada para un único uso, el análisis predictivo del género de *tracks* de música electrónica. Por eso, cuenta con un único rol: Usuario.

Con dicho rol, la persona utilizando la aplicación de escritorio puede acceder al funcionamiento completo. Esto significa que tiene la capacidad de analizar sus propios *tracks*, eligiendo la carpeta fuente y haciendo uso de la librería de los *tracks* previamente clasificados y del modelo de *Deep Learning*.

10.2 - Definición de funcionalidades

Las funcionalidades de la app son:

- Registro de usuario: Registro en Songo proveyendo nombre completo, correo electrónico y una contraseña. El registro usa *Firebase Authentication*.
- Login y logout de usuario: Login a la plataforma usando email y contraseña, datos que deben coincidir con los del registro. El login usa *Firebase Authentication*. El logout permite borrar los datos del usuario logueado.
- Navegación de carpetas y archivos: Navegar archivos locales, con el objetivo de seleccionar la carpeta fuente que contiene los archivos a analizar y ordenar.

- Ordenamiento de *tracks*: Clasificación y ordenamiento de *tracks* con las siguientes características.
 - Formatos de audio aceptados: MP3, AIFF y WAV.
 - Géneros aceptados:
 1. Deep House (DH)
 2. Tech House (TH)
 3. Melodic Techno (MT)
 4. Progressive (P)
 5. Techno Peak Time (TPT)
 6. Hard Techno (HT)
 7. Minimal (M)
 8. Trance (T)
 - Match de archivos locales con archivos previamente clasificados, según el nombre y la metadata.

10.3 - Especificación de user stories

El trabajo realizado para el diseño y la construcción de Songo fue delimitado por las *user stories* que se detallan a continuación.

01 - Capacitación tecnologías Machine Learning

Descripción:

Como Desarrollador, Quisiera capacitarme en tecnologías de Machine Learning, para poder desarrollar el modelo predictivo.

Criterios de aceptación:

- a. Las tecnologías en las que debe capacitarse el desarrollador son Python, Tensorflow y Keras.
- b. El desarrollador debe alcanzar un entendimiento básico sobre cómo construir una red neuronal convolucional, agregar capas a la red, entrenar el modelo y guardarla.

Dependencias: -**Estimación (USP):** 12**02 - Creación de datasets por género****Descripción:**

Como Product Owner, Quisiera crear los datasets de tracks para cada género, Para poder entrenar el modelo de Deep Learning

Criterios de aceptación:

- a. Debe crearse un dataset por cada género: Deep House, Tech House, Melodic Techno, Progressive, Techno Peak Time, Hard Techno, Minimal, Trance.
- b. Cada dataset debe contener al menos 100 tracks completos.
- c. Los tracks deben estar en formato MP3, AIFF o WAV (uno solo de ellos).
- d. No debe haber tracks repetidos.

Dependencias: -**Estimación (USP):** 24

03 - Investigación modelos de Machine Learning y elección

Descripción:

Como Product Owner, Quisiera investigar los distintos algoritmos de Machine Learning y elegir uno de ellos, Para poder comenzar con el desarrollo del modelo predictivo

Criterios de aceptación:

- a. El algoritmo debe ser de aprendizaje supervisado y de clasificación.
- b. Por cada familia de algoritmos debe realizarse una prueba rápida, con el objetivo de encontrar el más preciso a "simple vista".

Dependencias: 01

Estimación (USP): 24

04 - Investigación y elección de las características del audio a utilizar

Descripción:

Como Product Owner, Quisiera investigar las distintas características presentes en una señal de audio, Para elegir la más adecuada para la predicción de género

Criterios de aceptación:

- a. Elegir características significativas para la predicción de género, teniendo en cuenta experiencias previas de otras personas.
- b. La o las características elegidas deben poder extraerse utilizando librosa.

Dependencias: -

Estimación (USP): 12

05 - Creación y configuración del modelo

Descripción:

Como Desarrollador, Quisiera crear y configurar el modelo de Machine Learning, Para poder entrenarlo en base a los datasets

Criterios de aceptación:

- a. El modelo debe utilizar el algoritmo elegido por el Product Owner.
- b. El modelo debe ser creado con Tensorflow y Keras.
- c. Deben utilizarse las configuraciones por defecto del algoritmo, a menos que rápidamente (en cinco minutos o menos) se pueda identificar una mejora en alguno de ellos.
- d. Tener en cuenta las características del audio elegidas.

Dependencias: 03, 04

Estimación (USP): 36

06 - Entrenamiento y testeo del modelo

Descripción:

Como Desarrollador, Quisiera entrenar y testear el modelo, Para obtener una base de porcentaje de precisión del modelo

Criterios de aceptación:

- a. Los datasets deben separarse en *train*, *test* y *validation*, con un porcentaje de datos de 64%, 16% y 20% del total, respectivamente.

- b. De cada *track*, extraer las características elegidas por el *Product Owner*.
- c. Usar buenas prácticas para evitar el *overfitting*.

Dependencias: 02, 05

Estimación (USP): 24

07 - Iteración sobre las configuraciones del modelo

Descripción:

Como Desarrollador, Quisiera iterar sobre las configuraciones del modelo, Para obtener un porcentaje de precisión del modelo mayor al conseguido como base

Criterios de aceptación:

- a. Iterar sobre todas las configuraciones del modelo, haciendo cambios lógicos y experimentales.

Dependencias: 06

Estimación (USP): 24

08 - Guardado del modelo final con reporte

Descripción:

Como Desarrollador, Quisiera guardar el modelo final con su reporte de precisión, Para poder utilizarlo en el futuro y conocer la probabilidad de acierto

Criterios de aceptación:

- a. El modelo guardado debe ser aquel con las configuraciones que resulten en la mayor precisión.
- b. El modelo debe guardarse en un archivo con formato h5.

Dependencias: 07

Estimación (USP): 12

09 - Registro de usuario

Descripción:

Como Usuario Quisiera poder registrarme en la aplicación, Para poder loguearme y acceder a su funcionalidad

Criterios de aceptación:

- a. El registro se hará con nombre completo, correo electrónico y contraseña.
- b. El registro utilizará *Firebase Authentication* como motor.
- c. Se accede al formulario de registro con un botón desde la pantalla inicial.
- d. Una vez realizado el registro, se redirige al usuario a la pantalla de *home*.
- e. El correo electrónico no necesitará ser verificado.

Dependencias: -

Estimación (USP): 12

10 - Login y logout de usuario

Descripción:

Como Usuario Quisiera poder loguearme y desloguearme de la aplicación, Para poder acceder a su funcionalidad

Criterios de aceptación:

- a. El *login* se realizará con correo electrónico y contraseña.

- b. El correo electrónico y contraseña utilizados deben coincidir con el de un usuario registrado. En caso de no coincidir se mostrará un estado de error.
- c. El *login* y *logout* utilizarán *Firebase Authentication* como motor.
- d. El *logout* se hará clickeando un botón en la barra lateral de la aplicación.
- e. Luego de *loguearse*, el usuario será redirigido a la pantalla de *home*.

Dependencias: 09

Estimación (USP): 12

11 - Visualización de carpetas y archivos locales

Descripción:

Como Usuario, Quisiera poder visualizar mis carpetas y archivos locales, Para poder elegir la carpeta fuente para el análisis predictivo

Criterios de aceptación:

- a. El *file manager* debe permitir la navegación de carpetas y subcarpetas, dando la posibilidad al usuario de ir "hacia atrás" y "hacia adelante".
- b. El usuario debe visualizar el contenido actual de la carpeta seleccionada.
- c. En caso de actualizar archivos o subcarpetas de la carpeta seleccionada por fuera de la aplicación, estos cambios no se verán reflejados hasta que el usuario salga y entre a dicha carpeta nuevamente.

Dependencias: 10

Estimación (USP): 12

12 - Elección de carpeta y análisis de tracks

Descripción:

Como Usuario, Quisiera poder analizar mis tracks, Para poder predecir su género y ordenarlos en base a eso

Criterios de aceptación:

- a. Utilizando el *file manager*, el usuario debe poder elegir la carpeta en la que se encuentra actualmente para hacer el análisis.
- b. El análisis sólo tomará en cuenta los archivos MP3, AIFF y WAV en esa carpeta.
- c. El análisis hará uso de los *tracks* previamente clasificados.
- d. Al finalizar el análisis, los *tracks* analizados serán guardados con su nombre y metadatos.
- e. Al finalizar el análisis, los *tracks* serán reubicados en distintas carpetas según el género predicho.
- f. Las carpetas de géneros necesarias que no existan serán creadas automáticamente.

Dependencias: 11

Estimación (USP): 36

11 - Interfaz de usuario

Login

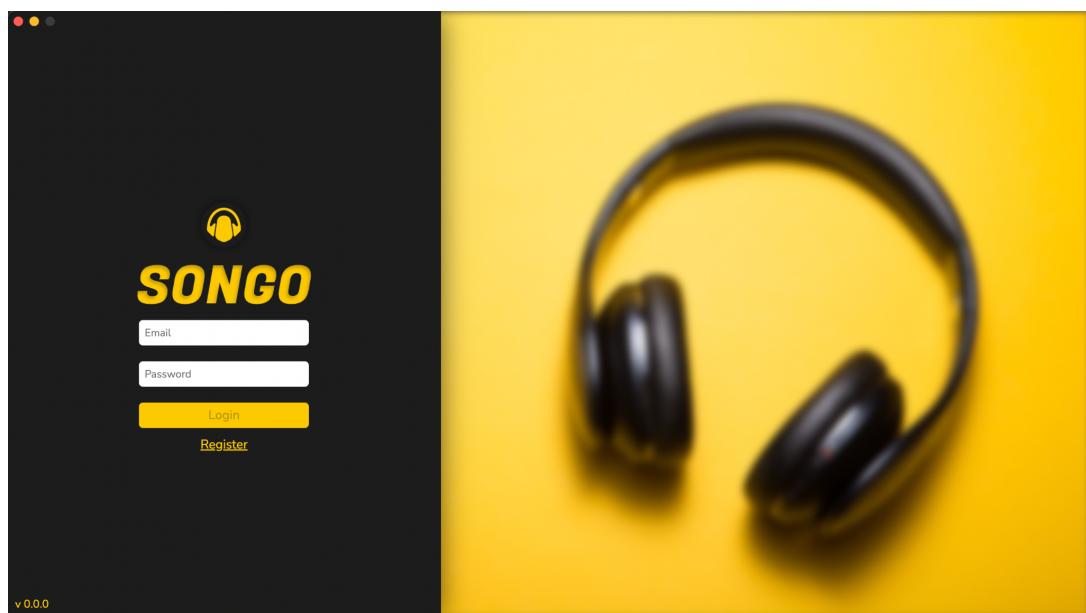


Figura 11.1. Login de Songo.
Fuente: Elaboración propia.

En la pantalla de inicio de sesión se renderiza un formulario que contiene campos de correo electrónico y contraseña, y debajo un botón para ejecutar el inicio de sesión, acorde a los criterios de aceptación.

También hay un link para navegar a la pantalla de registro en el caso que el usuario no se haya creado una cuenta aun.

Registro

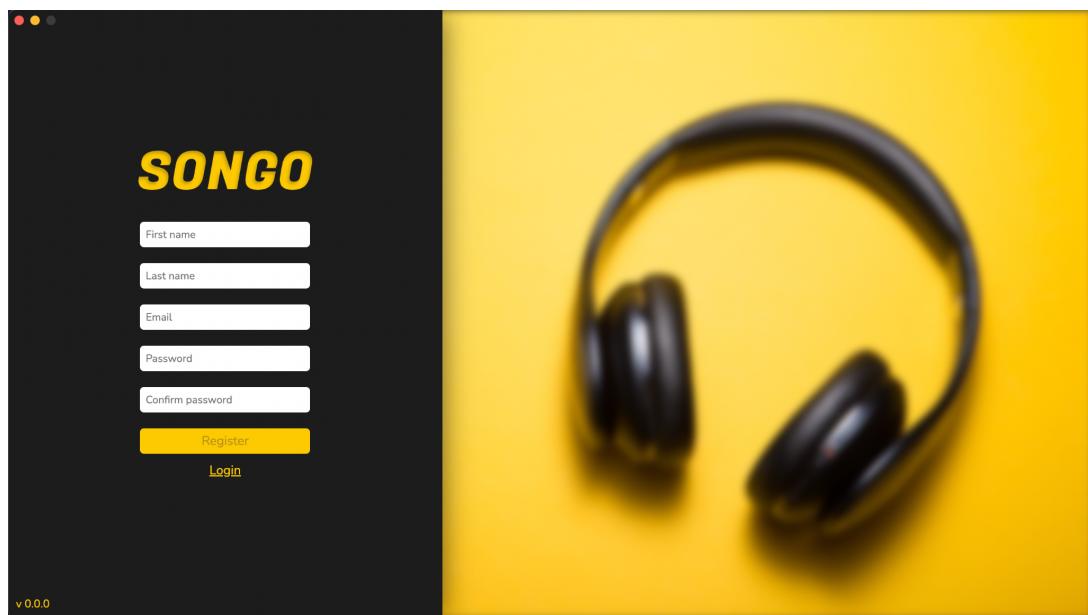


Figura 11.2. Registro de Songo.
Fuente: Elaboración propia.

En la pantalla de registro se renderiza un formulario con campos para introducir nombre, apellido, correo electrónico, contraseña y confirmar la contraseña. Debajo se renderiza un botón para ejecutar el registro y un link para navegar a la pantalla de inicio de sesión en caso de ser necesario.

Selección de directorio

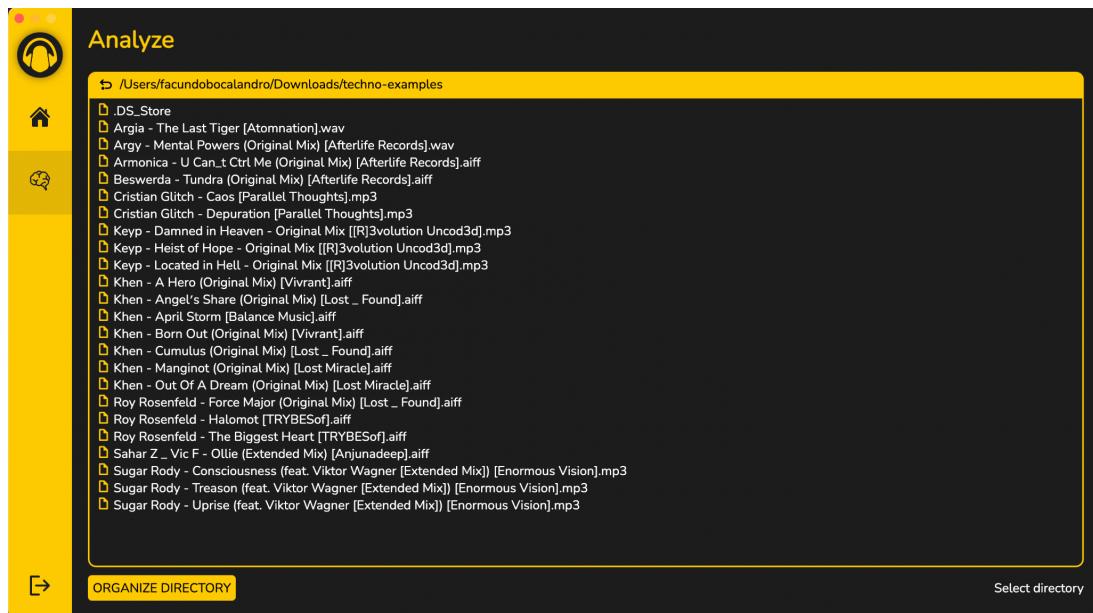


Figura 11.3. Navegador del sistema de directorios de Songo.
Fuente: Elaboración propia.

En la pantalla de análisis el componente principal es el navegador del sistema de directorios que fue hecho desde cero para tener flexibilidad en la funcionalidad que se quiera implementar con el mismo. Debajo se encuentra renderizado un botón que ejecuta el proceso de clasificación en el directorio donde se encuentre ubicado el navegador.

Comienzo del proceso de clasificación

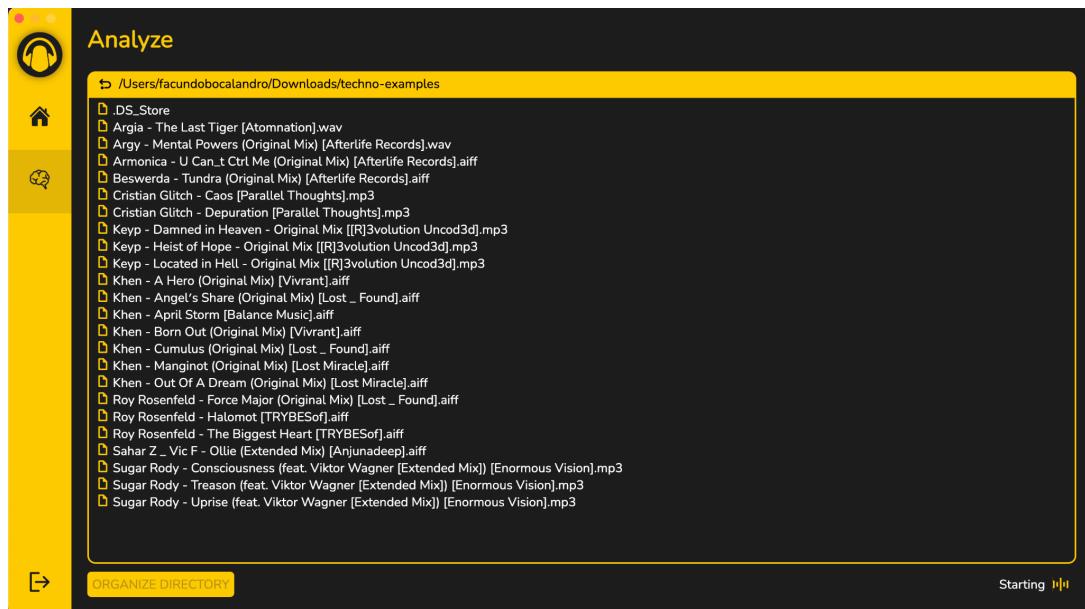


Figura 11.4. Comienzo del proceso de clasificación en Songo.

Fuente: Elaboración propia.

Una vez iniciado el proceso de clasificación se puede observar en la esquina inferior derecha el texto con un ícono animado que indica la etapa y el progreso del proceso. Como se puede observar, está comenzando el proceso y todavía no se clasificó ningún archivo.

Progreso del proceso

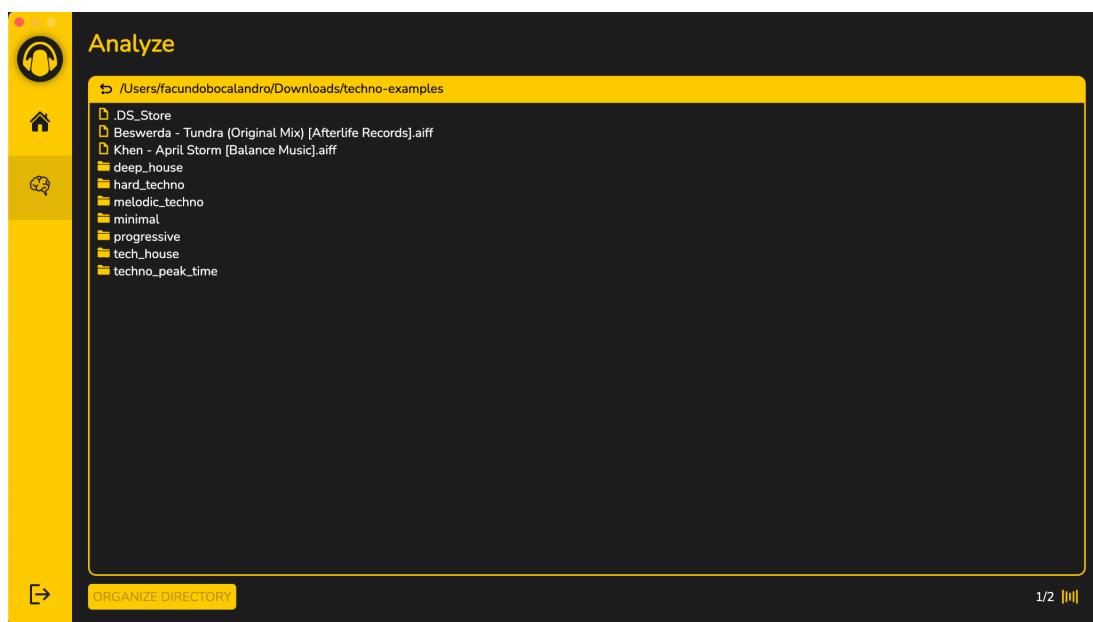


Figura 11.5. Progreso del proceso de clasificación en Songo.
Fuente: Elaboración propia.

La mayoría de los archivos ya fueron ubicados en sus respectivos subdirectores según el género al cual pertenecen. Para los dos restantes, se está ejecutando el proceso de análisis porque son archivos que no se analizaron nunca. Para todos los anteriores el proceso ya identificó que fueron analizados previamente y con esa información género los subdirectorios y movió los archivos.

Proceso finalizado

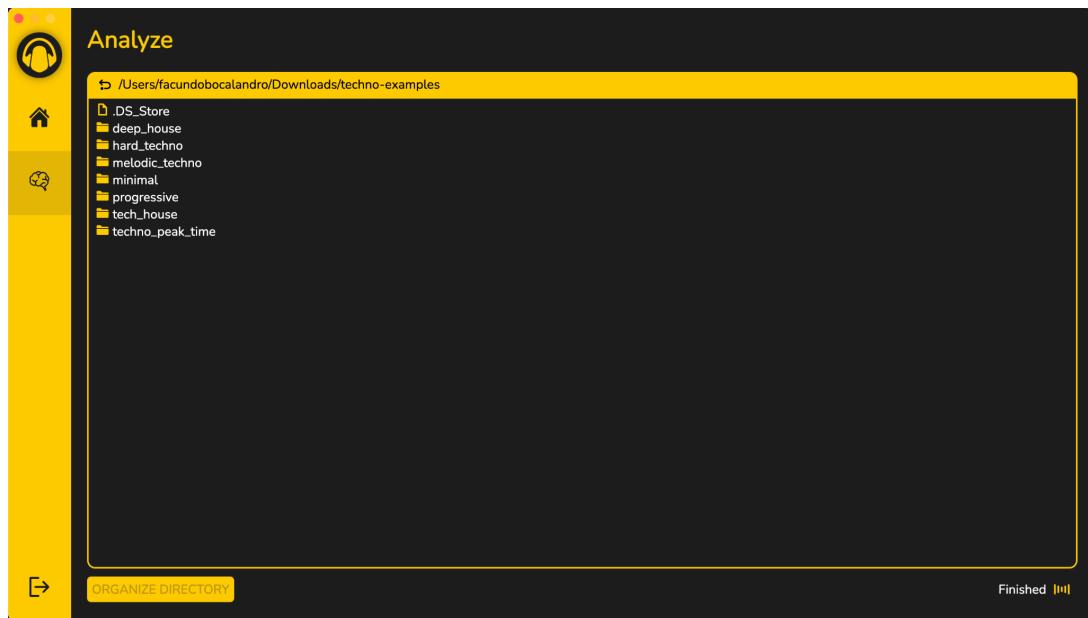


Figura 11.6. Fin del proceso de clasificación en Songo.
Fuente: Elaboración propia.

Finalmente, el proceso de clasificación termina dejando el directorio que se especificó para ordenar, con los subdirectorios creados y los archivos ubicados dentro de los mismos, acorde a los criterios de aceptación.

12 - Dificultades encontradas y soluciones propuestas

Durante el planeamiento, diseño e implementación de Songo, surgieron problemas de distinta índole. A continuación, se detallan aquellos que produjeron más tiempo de investigación en búsqueda de la solución adecuada.

Dificultad 1

El procesamiento de los *tracks* en su totalidad consume muchos recursos (memoria y CPU), así como también puede requerir mucho tiempo.

Como solución, los *tracks*, que comúnmente tienen una duración de entre seis y diez minutos, se dividen en cuatro muestras de treinta segundos equiespaciadas, unificando la duración de procesamiento por *track* a dos minutos.

La idea de tener segmentos de audio equiespaciados surge de que por naturaleza, la música *techno* tiene variaciones muy marcadas a lo largo de un mismo *track*.

Para generar los segmentos de audio se utiliza la librería pydub de Python.

Dificultad 2

A diferencia del análisis predictivo de géneros musicales a nivel general (rock, jazz, blues, etc.), el análisis de subgéneros de *techno* puede ser muy tedioso. Esto tiene razón de ser en el hecho de que los parámetros básicos, como el BPM, tienen valores muy similares incluso entre distintos subgéneros.

Para solucionar esto, se analizan los coeficientes MFCC de cada *sample* de treinta segundos. Estos coeficientes permiten analizar numéricamente cualquier señal de sonido, similar a como lo haría el oído humano.

Así, aunque en algunos casos sigue resultando difícil diferenciar entre *tracks* dos subgéneros (inclusive para el oído humano), se resuelve la mayor cantidad de predicciones erróneas.

Dificultad 3

Existen una infinita cantidad de posibles combinaciones a la hora de crear un modelo de *Deep Learning*. Los tipos de capas a elegir, las funciones de activación y la cantidad de capas son algunas de las preguntas que debe hacerse una persona al diseñar una red neuronal desde cero.

Es por ello que para primeras experiencias con *Deep Learning* es recomendable usar como base una red ya existente, y lentamente ir ajustando las configuraciones. Esta es la solución al desafío mencionado.

Como base, se usa una *Convolutional Neural Network* con tres capas *convolutional*, cuatro capas *ReLU*, tres capas *max pooling* y una capa *flatten*.

Dificultad 4

TensorFlow es una de las librerías de *Machine Learning* para *Python* más populares. Posibilita, entre otras cosas, crear modelos de *Deep Learning*, entrenarlos y guardarlos para su uso en el futuro. Una de sus contrapartes es la compatibilidad únicamente con versiones específicas de *Python*. Esto impide, en algunos casos, instalar *TensorFlow* localmente si no se tiene alguna de esas versiones.

Para evitar este problema y otros similares que puedan ocurrir a la hora de trabajar con *Python*, se crea lo que se conoce como entornos virtuales (*virtual environments*). Para solucionar el problema particular de *TensorFlow*, se usa *Conda*, especificando una versión de *Python* y una de *TensorFlow* que sean compatibles. Esto hace posible no solo usar las versiones específicas de esta

librería y otras, sino también aislar los entornos de otros entornos y del entorno base de la computadora.

Dificultad 5

Es frecuente en el ámbito del Machine Learning y Deep Learning encontrar papers, documentos e inclusive código de personas con profesiones relacionadas a la matemática y la estadística. Como ventaja, estas personas conocen y entienden las bases de los algoritmos predictivos. Sin embargo, como desventaja, generalmente tienen bajo conocimiento de programación, buenas prácticas y principios por lo que se dificulta en gran medida la lectura de su código.

La solución a este problema no es otra que dedicarle más tiempo a la lectura y entendimiento de los casos y los modelos propuestos por estas personas. En muchos casos, puede ser necesario entender la base teórica que sustenta los modelos.

13 - Propuestas de mejora

Songo es una plataforma que permite a sus usuarios clasificar y ordenar tracks de techno según su género. A pesar de cumplir con su funcionalidad, existen múltiples mejoras que podrían incorporarse a la app, de modo tal que sea más eficiente, precisa y escalable.

Propuesta 1

La primera propuesta es con respecto al modelo. A pesar de ser efectivo, con una precisión cercana al 80%, en Deep Learning siempre hay espacio para la mejora.

En este caso, siguiendo el principio de "no reinventar la rueda", puede ser útil usar el paper "*Deep Learning Based EDM Subgenre Classification using Mel-Spectrogram and Tempogram Features*" de investigadores del Research Center for IT Innovation, de la Academia Sinica, en Taipei, Taiwán.

En dicho paper, se hace una investigación sobre distintos modelos de Deep Learning y distintas características del audio posibles para clasificar música electrónica en treinta subgéneros distintos. La conclusión a la que arriban los autores es que las características a utilizar son *Mel-spectrogram*, *Autocorrelation tempogram* y *Fourier tempogram*, y el modelo más óptimo entre los analizados es el *Short-chunk CNN with Resnet*.

La arquitectura propuesta contiene siete copias de capas para la extracción de características, cada una conteniendo dos capas convolution, dos capas batch

normalization y una capa ReLU entre las convolution. El output de este conjunto de capas pasa por una capa max pooling, dos capas dense y una última capa softmax para la clasificación. El modelo está diseñado para procesar segmentos cortos de audio, de dos segundos cada uno.

Haciendo uso de las características del audio y el modelo mencionado podría incrementar notablemente la precisión.

Propuesta 2

Songo puede correrse de manera local a través del código fuente. Esto es muy poco escalable, ya que debe realizarse un trabajo exhaustivo para el setup de la aplicación. Una experiencia de usuario más estética y rápida sería la descarga de la aplicación ya empaquetada a través de una landing page.

Para lograr esto, es necesario proponer una infraestructura simple pero útil acorde a este propósito.

La infraestructura propuesta utiliza dos servicios de Amazon Web Services:

- EC2
- S3

El servicio EC2 permite alojar el backend y base de datos de la aplicación en un servidor cloud. Para ello, a través de Github Actions se publica una imagen de Docker, en DockerHub ,que permite instanciar el backend. La máquina proveída por EC2 instala esta imagen, así como también la imagen de Docker de Postgres, y crea contenedores para correrlas.

El servicio S3 sirve para almacenar tanto los archivos estáticos de la *landing page* de descarga de la app, así como también la aplicación de Electron y Python ya empaquetada.

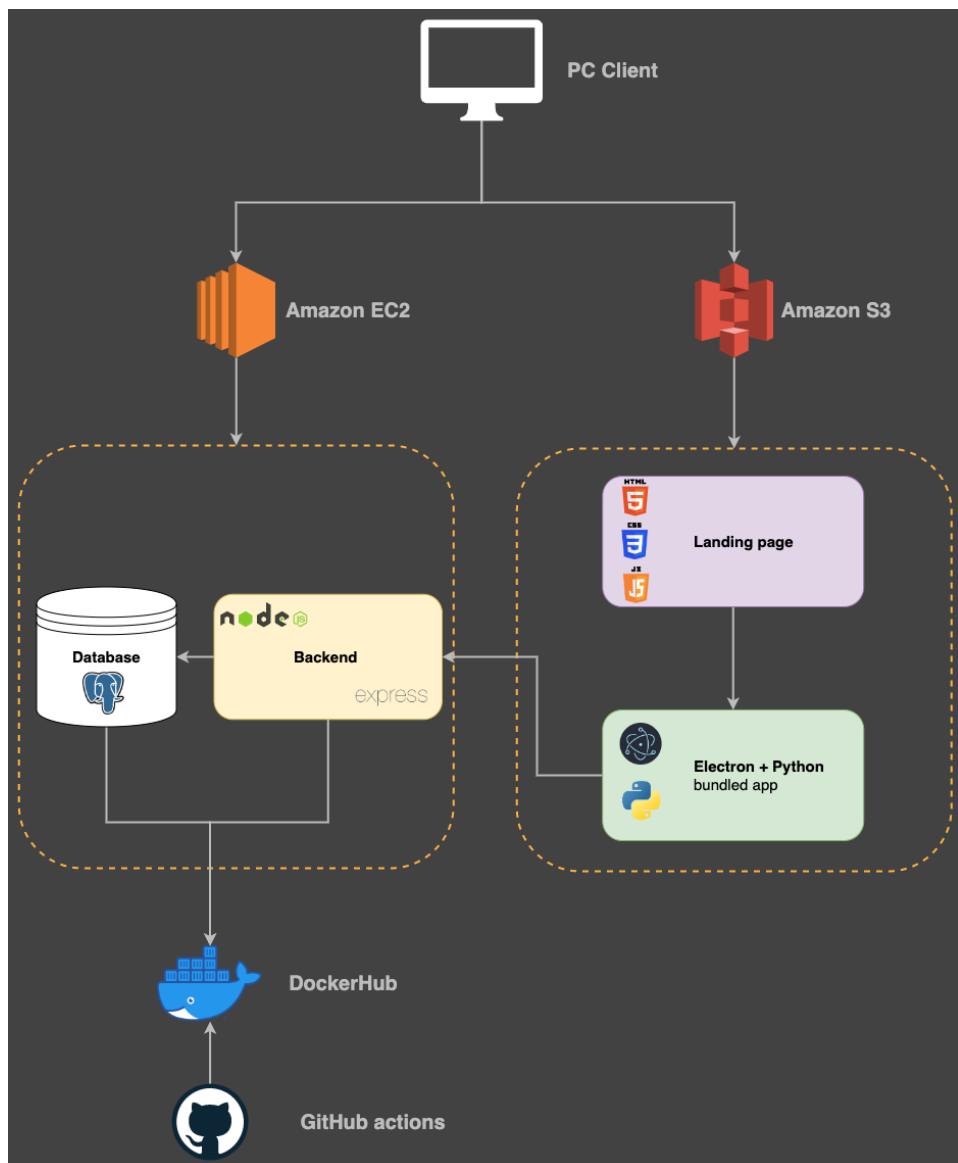


Figura 13.1. Propuesta de infraestructura para Songo.
Fuente: Elaboración propia.

Propuesta 3

La plataforma tiene un enorme potencial dentro de la comunidad de DJs *techno*. Para aprovechar la oportunidad, debe encontrarse un modelo de negocio adecuado que favorezca tanto a los usuarios como a los dueños de Songo.

Un modelo popular en plataformas de Streaming como Spotify es el *freemium*. Básicamente, su nombre indica su naturaleza, parte *free* (gratis) y parte *premium* (pago).

Aplicado a Songo, el *Freemium* se puede lograr aplicando un límite de tracks analizados según el plan comprado. Para ello, cada usuario registrado de la aplicación debe tener un plan asignado (por defecto, el gratuito).

También, los *scripts* de python deben correrse en el *backend*, ya que de manera contraria el código fuente sería accesible para los usuarios. Para lograr esto, los archivos de audio deben subirse a un servicio de almacenamiento (como Amazon S3), para luego ser utilizados por el *backend* para el análisis predictivo.

14 - Validación del mercado

Si bien el desarrollo del proyecto no incluyó una etapa de despliegue del sistema, se pudo validar de diferentes maneras. La validación del sistema a nivel funcional fue el testeo manual del mismo en los equipos propios de los desarrolladores, de manera local. Este testeo manual fue llevado a cabo por algunos potenciales usuarios pertenecientes al entorno de la música electrónica, productores y DJs. Además de esto se validó la idea general del proyecto con potenciales usuarios, siendo estos productores y DJs profesionales de la escena argentina. A continuación se detalla el feedback recibido de las respuestas de los diferentes potenciales usuarios.

"La idea está bárbara, simplifica un montón el laburo a la hora de organizar la música para los sets. No se como funciona la app por atras, pero capaz a los tracks que son difíciles de clasificar en algun subgenero los metes en alguna carpeta general que se llame electrónica como las paginas para bajar tracks de internet." - Pedro Belardo, productor y DJ de Progressive House avalado por figuras mundiales como Guy J.

"Es muy interesante y útil lo que plantean. Sinceramente no se si le agregaría algo, ya con que ordene la música por subgéneros es suficiente. Sin dudas ahorra mucho tiempo de laburo, asi que lo usaría feliz." - Alfonso Pratgay, DJ de Indie y Melodic Techno, residente de la productora Kool Kidz.

"Muy buena idea, más de una vez me puse a buscar apps que reconozcan géneros y nunca encontré nada bueno, me parece más que interesante, lo usaría 100%." - Joaquin Ferri, DJ de Techno Peak Time y Hard Techno, de la escuela Arjaus.

"Lo usaría, lo probaría y vería si me es útil. Aunque en mi caso la música que bajo para tocar es bastante similar, no suelo variar en más de 2 o 3 tipos de subgéneros Techno. No se si identificará esos sub géneros, ahí ya entra la percepción de cada uno." - Inazio, DJ y productor de Hard Techno, avalado por figuras mundiales como Richie Hawtin.

"Dado nuestra forma de trabajo no lo usaríamos. Pero lo encontramos útil, en caso de necesitarlo lo tendríamos en cuenta. Lo mejor hoy por hoy que podrían agregarle es un análisis de tonalidad. Si bien hay otras apps que se encargan de hacerlo bien, lo vemos muy útil en este caso." - Mens of Clam, DJs y productores de Techno Peak Time y Hard Techno, avalados por figuras mundiales como Dubfire.

Como se puede leer en las respuestas obtenidas, hay un mix de opiniones. Si bien la mayoría de los entrevistados piensan que la idea del sistema es buena y que lo utilizarían, hay algunos que no lo encuentran muy aplicable a su trabajo, y esto depende mucho del estilo que tienen los DJs. Los DJs que trabajan enfocados en un solo género o menos, y que no son muy versátiles en los estilos que utilizan, no encuentran de mucha utilidad el producto, ya que al no tener variedad de géneros, no tienen grandes cantidades de tracks para ordenar y clasificar. En estos casos como mencionan los Mens, podría agregarse una análisis de tonalidad o de energía que tienen los tracks, para tener otras clasificaciones además de los géneros al que pertenecen.

En general se puede decir que el producto tuvo un muy buen feedback del lado de los potenciales usuarios a los cuales se entrevistó. Con más testimonios y más conversaciones, son dudas que se puede ampliar el producto y sus funcionalidades para abarcar más mercado, y ofrecer un sistema con más calidad y valor.

15 - Conclusiones

El proceso del proyecto fue muy fructífero. A lo largo del desarrollo fueron surgiendo muchos desafíos los cuales algunos fueron enfrentados y solucionados con éxito y otros no, lo cual no es algo necesariamente malo. Uno de los ejemplos más claros de esto es el proceso de prueba y error que se realizó para hallar que modelo de aprendizaje automático era el que mejor se desempeñaba según los requerimientos del sistema en desarrollo. Muchos de los modelos fracasaron, pero no fue en vano, hubo mucho aprendizaje y crecimiento técnico involucrados en la búsqueda de la solución al problema planteado.

A pesar de estos desafíos y obstáculos, el proyecto fue desarrollado exitosamente, logrando cumplir con los objetivos que se plantearon en un comienzo y proveyendo una solución de utilidad y funcional al problema en cuestión. No solo se desarrolló un modelo de aprendizaje profundo capaz de identificar subgéneros de música (cabe aclarar que la dificultad es mucho más elevada que la clasificación entre diferentes géneros y no subgéneros), sino que también se alcanzó un desarrollo formidable de una aplicación de escritorio con sus diversos módulos que la componen, siendo esta funcional y útil para el problema planteado.

Si bien es posible incrementar la precisión del modelo de aprendizaje, se puede afirmar que se construyeron las bases sólidas para un producto que puede revolucionar el trabajo diario de aquellos que pasan gran parte de sus vidas en contacto con la música. Con solo entrenar el modelo para diferentes géneros, con sets de datos más grandes y diversos, se puede ampliar la utilidad y funcionalidad del sistema exponencialmente.

16 - Bibliografía

- Sanket Doshi (2019, 5 de enero). Classification of Music into different Genres using Keras. <https://medium.com/@sdoshi579/classification-of-music-into-different-genres-using-keras-82ab5339efe0>
- Seldon (2021, 11 de septiembre). How to Build a Machine Learning Model. <https://www.seldon.io/how-to-build-a-machine-learning-model>
- Arsh Chowdhry (2021, 7 de mayo). Music Genre Classification Using CNN. <https://www.clairvoyant.ai/blog/music-genre-classification-using-cnn>
- Derek A. Huang, Arianna A. Serafini, Eli J. Pugh (2018). Music Genre Classification. <http://cs229.stanford.edu/proj2018/report/21.pdf>
- Projectpro (2022, 4 de octubre). Solved Music Genre Classification Project using Deep Learning. <https://www.projectpro.io/article/music-genre-classification-project-python-coded/566>
- Papia Nandi (2021, 24 de marzo). CNNs for Audio Classification. <https://towardsdatascience.com/cnn-for-audio-classification-6244954665ab>
- Kunal Vaidya (2020, 7 de diciembre). Music Genre Recognition using Convolutional Neural Networks (CNN) – Part 1. <https://towardsdatascience.com/music-genre-recognition-using-convolutional-neural-networks-cnn-part-1-212c6b93da76>
- Wei-Han Hsu, Bo-Yu Chen, Yi-Hsuan Yang (2021, 17 de octubre). Deep Learning Based EDM Subgenre Classification using Mel-Spectrogram and Tempogram Features. <https://arxiv.org/abs/2110.08862>

- doug, Johan_A_M (2022, 31 de agosto). How to choose the number of hidden layers and nodes in a feedforward neural network? <https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-network>
- D. STATHAKIS (2009, 20 de abril). How many hidden layers and nodes? http://dstath.users.uth.gr/papers/IJRS2009_Stathakis.pdf
- Warren S. Sarle (2002). Comp.ai.neural-nets FAQ, Part 1 of 7: Introduction. <http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html>
- Saul Dobilas (2021, 27 de diciembre). Feed Forward Neural Networks – How To Successfully Build Them in Python. <https://towardsdatascience.com/feed-forward-neural-networks-how-to-successfully-build-them-in-python-74503409d99a>
- jaleroux (2019). Electronic Music Genre Classification. <https://github.com/jaleroux/music-genre-classification>
- eatsleeprepeat (2019). Electronic music styles Neural Net. https://github.com/eatsleeprepeat/emusic_net
- Keunwoo Choi, Gyorgy Fazekas, Mark Sandler. AUTOMATIC TAGGING USING DEEP CONVOLUTIONAL NEURAL NETWORKS. <https://arxiv.org/pdf/1606.00298.pdf>
- Minz Won (2021, 25 de agosto). State-of-the-art Music Tagging Models. <https://github.com/minzwon/sota-music-tagging-models>
- Minz Won Andres Ferraro Dmitry Bogdanov Xavier Serra (2020, 1 de junio). Evaluation of CNN-based Automatic Music Tagging Models. <https://arxiv.org/pdf/2006.00751.pdf>

- Anne Bonner (2019, 13 de enero). The Complete Beginners Guide to Deep Learning.
<https://towardsdatascience.com/intro-to-deep-learning-c025efd92535>
- Anne Bonner (2019, 19 de enero). The Complete Beginner's Guide to Deep Learning: Artificial Neural Networks.
<https://towardsdatascience.com/simply-deep-learning-an-effortless-introduction-45591a1c4abb>
- Anne Bonner (2019, 2 de febrero). The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification.
<https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>
- Wikipedia (2022, 28 de noviembre). Deep learning.
https://en.wikipedia.org/wiki/Deep_learning
- Engin Pekel (2017, marzo). Feed forward and recurrent ANN architecture.
https://www.researchgate.net/figure/Feed-forward-and-recurrent-ANN-architecture_fig1_315111480
- Sumit Saha (2018, 15 de diciembre). A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way.
<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Jason Brownlee (2019, 27 de noviembre) . How to Choose a Feature Selection Method For Machine Learning.
<https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>
- Ashish (2021, 27 de junio). Feature Selection using Statistical Tests.
<https://www.analyticsvidhya.com/blog/2021/06/feature-selection-using-statistical-tests>
- Aman Gupta (2020, 10 de octubre). Feature Selection Techniques in Machine Learning.

<https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning>

- Vikas Verma (2020, 24 de octubre). A comprehensive guide to Feature Selection using Wrapper methods in Python.
<https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python>
 - Kiran Parte (2020, 3 de noviembre). Dimensionality Reduction: Principal Component Analysis.
<https://medium.com/analytics-vidhya/dimensionality-reduction-principal-component-analysis-d1402b58feb1>
 - derekahuang (2019, 25 de abril). Music-Classification.
<https://github.com/derekahuang/Music-Classification>
-
- Jorge Castañón (2019, 1 de mayo). 10 Machine Learning Methods that Every Data Scientist Should Know.
<https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9>
 - Saishruthi Swaminathan (2018, 15 de marzo). Logistic Regression – Detailed Overview.
<https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>
 - Rohith Gandhi (2018, 7 de junio). Support Vector Machine – Introduction to Machine Learning Algorithms.

<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

- Onel Harrison (2018, 10 de septiembre). Machine Learning Basics with the K-Nearest Neighbors Algorithm.
<https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- Tony Yiu (2019, 12 de junio). Understanding Random Forest.
<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- Prashant Gupta (2017, 17 de mayo). Decision Trees in Machine Learning.
<https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>
- Devopedia (2021). Audio Feature Extraction.
<https://devopedia.org/audio-feature-extraction>
- Sanket Doshi (2018, 30 de diciembre). Music Feature Extraction in Python.
<https://medium.com/towards-data-science/extract-features-of-music-75a3f9bc265d>
- Olivia Tanuwidjaja (2022, 18 de enero). Get To Know Audio Feature Extraction in Python.
<https://towardsdatascience.com/get-to-know-audio-feature-extraction-in-python-a499fdafef42>
- Leland Roberts (2020, 6 de marzo). Understanding the Mel Spectrogram.
<https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>
- mlearnere (2021, 16 de febrero). Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients.
<https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8>

- Wikipedia (2022, 29 de agosto). Spectrogram.
<https://en.wikipedia.org/wiki/Spectrogram>
- Wikipedia (2022, 25 de agosto). Mel-frequency cepstrum.
https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
- Aakash Mallik (2019, 19 de mayo). Electron + React + Python (Part 1) – Introduction.
<https://medium.com/heuristics/electron-react-python-part-1-introduction-b228ccf8e889>
- Andy Bulka (2018, 3 de octubre). Building a deployable Python-Electron App. <https://medium.com/@abulka/electron-python-4e8c807bfa5e>
- Tomasz Wegrzanowski (2021, 14 de agosto). Electron Adventures: Episode 22: File Manager in React.
<https://dev.to/taw/electron-adventures-episode-22-file-manager-in-react-hi2>
- Shruti M (2022, 15 de noviembre). Discover the Differences Between AI vs. Machine Learning vs. Deep Learning.
<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning>
- Alain Perkaz (2021, 14 de julio). Advanced Electron.js architecture.
<https://blog.logrocket.com/advanced-electron-js-architecture/>
- Marc Saint-Félix (2021, 20 de junio). Music Genre Detection With Deep Learning.
<https://towardsdatascience.com/music-genre-detection-with-deep-learning-cf89e4cb2ecc>