ESCUELA DE INGENIERÍA DE FUENLABRADA GRADO EN

INGENIERÍA DE SISTEMAS AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

TÍTULO

Autor: Víctor Iglesias Cuevas

Tutora: Rebeca Goya Esteban

Curso académico 2023/2024

"Las emociones no son buenas ni malas, son simplemente nuestras respuestas a la vida." (Shari Y. Manning)

Agradecimientos

aquí van los agradecimientos

Resumen

Esto es un resumen

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 Introducción

1.1 Motivación y contexto

La música juega un rol importante en la vida de la personas, aun más en la era digital. El uso de Internet ha contribuido enormemente en el crecimiento de librerías digitales de música, y con ello, la información y metadatos disponibles de cada pista de audio. El poder emocional de la música es el motivo de su aplicación en áreas tan diversas como la industria del juego, la industria cinematográfica, marketing y musicoterapia, pero los conocimientos científicos sobre este fenómeno están lejos de ser completos o reveladores [3].

La relación entre música y emoción ha sido objeto de muchos debates académicos e investigaciones empíricas en muchas disciplinas diferentes donde están incluidas la filosofía, la musicología, la psicología, la biología, la antropología y la sociología. Sin embargo, no fue hasta la década de los 2000, cuando el estudio de este tema desde el punto de vista de la ingeniería, comenzó a desarrollar un modelo computacional de emoción musical para la recuperación y organización de la música basada en emociones [15].

Tradicionalmente, la clasificación de música ha estado basada en catálogos de metadatos (artista, album, título de la canción, ...) [15]. Esta clasificación puede no ser suficiente para el usuario. Tener acceso a una categorización de la música en función de emociones o estados de ánimo puede ser de gran utilidad para muchas aplicaciones. Imaginemos un reproductor de música que, además de recomendarte canciones según un grupo o un estilo musical, sea capaz de hacerlo según el estado de ánimo que tengas.

Aquí es donde entra en juego el Reconocimiento Musical de Emociones (Music Emotion Recognition). MER categoriza emociones y aplica técnicas de Machine Learning (Aprendizaje Máquina) para clasificarlas usando diferentes características extraídas de la señal acústica de la canción [15].

En este trabajo blablablablabla.....

1.2 Objetivos del trabajo

El objetivo principal del trabajo es diseñar un sistema para el reconocimiento de emociones en pistas de audio. Los objetivos específicos que se plantean son los siguientes:

- Estudiar el Estado del Arte de los sistemas MER
- Examinar técnicas de Aprendizaje Máquina (Machine Learning) para la detección de emociones
- Usar lenguage Python para:
  - procesar de datos del banco de datos
  - desarrollar algoritmos de Machine Learning
  - entrenar y probar los sistemas para ajustar parámetros
  - evaluar las técnicas de Machine Learning según su capacidad para predecir los valores requeridos

1.3 Metodología y estructura de la memoria

La memoria está compuesta por cuatro apartados:

1. Una introducción de la memoria donde se expone el contexto y la motivación del trabajo.
2. El capítulo de *Estado del arte* donde se expone:
   - una introducción a la campo de estudio MER,
   - una descripción de Machine Larning y los diferentes tipos de algoritmos que hay,

- y una explicación de los modelos de regresión y algunos de los ejemplo más comunes

3. El capítulo de *Experimentos* dividido en:

   - descripción de los datos utilizados,
   - y los algoritmos de Machine Learning empleados en el trabajo

4. Los resultados detallados de los experimentos realizados en el capítulo anterior.

5. Las conclusiones obtenidas del trabajo, y líneas futuras en las que seguir profundizando en MER.

2 Estado del arte

2.1 Music Emotion Recognition

Music Emotion Recognition (MER) es un campo de estudio que se enfoca en identificar y clasificar las emociones que la música evoca en los oyentes. Para ello, se extraen las características relevantes de las pistas de audio, se procesan, se evalúan, y posteriormente se asocian con determinadas emociones [8].
MER es una de las disciplinas de alto nivel más desafiantes en Music Information Retrieval (MIR). MIR un campo de investigación que se centra en la extracción e inferencia de características significativas de la música (a partir de la señal de audio), la clasificación de la música utilizando estas características, y el desarrollo de diferentes esquemas de búsqueda y recuperación. Tiene como objetivo poner a disposición de los individuos el vasto almacén de música del mundo [13]

2.1.1 Emociones

Los modelos teóricos de la emoción adoptados por los estudios se dividieron en cuatro clases: discretos, dimensionales, misceláneos y específicos de la música [3].
Uno de los más comunes usado en psicología es el modelo dimensional propuesto por Russell [12]. El modelo consiste en la representación de las emociones a través de dos dimensiones: valencia (nivel de agrado o desagrado) y activación (representa el nivel de energia o activación de la valencia). Cada emoción es una representación linea de la combinación de estas dos dimensiones

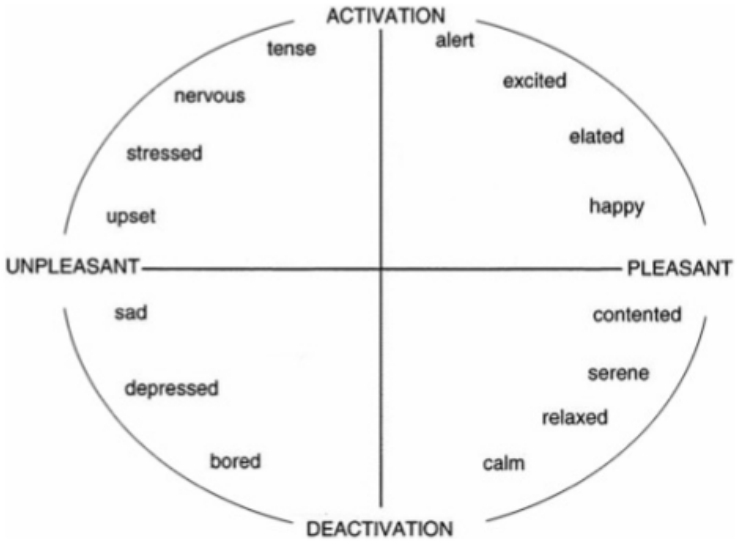

Figure 1: Modelo dimensional de Russell

Un aspecto importante en MER es la categorización de las emociones según su naturaleza. Se distinguen dos tipos:

- Percibidas: son aquellas que hacen referencia a las características musicales canción: tono, escala, ritmo, etc.
- Inducidas: son las emociones que hacen referencia al contexto individual de cada persona

[15] expone un ejemplo claro para distinguir entre los dos tipos: el Cumpleaños Feliz. Si analizamos la canción por sus características, se podría decir que es una canción alegre (emoción percibida). Pero es posible que para algunas personas les produzca tristeza al asociar un recuerdo con esta canción (emoción inducida)

2.1.2 Estado actual MER

El flujo de trabajo se divide en cuatro bloques [8]:

2.1.2.1 Taxonomía y propiedades musicales de la emoción

Es la forma en que se clasifican y categorizan las emociones. Hay dos taxonomías predominantes en el marco actual de MER:

- Enfoque categórico/discreto presenta diferentes clases: feliz, triste, etc. Este enfoque tiene una resolución mala y resulta ambiguo
- Enfoque dimensional. Como explica el apartado anterior, se conceptualiza la emoción como un elemento bidimensional (valencia y activación). Este enfoque puede resultar más exacto que el anterior, pero también más complejo a la hora de mapear las emociones en el marco que propone Russell [12]

La elección de la taxonomía es la decisión más importante para diseñar un conjunto de datos, y define el nivel de precisión emocional.

2.1.2.2 Creación del conjunto de datos y recopilación de anotaciones subjetivas

La forma más utilizada para la recopilación de emociones musicales es la subjetiva. Los anotadores suelen escuchar extractos de canciones (de unos 30 segundos) para posteriormente emitir juicios emocionales. Los datos son anotados siguiendo la taxonomía elegida para la creación del conjunto de datos. Estos anotadores son expertos en música, pero sin conocimiento alguno de teoría musical (musicólogos, productores, investigadores, etc.).
Todas las calificaciones son promediadas para llegar a una "verdad universal". Es decir, los valores finales son un promedio de las anotaciones de cada experto.

2.1.2.3 Extracción de las características

En los conjuntos de datos se encuentran características de dos tipos: propiedades acústicas a bajo nivel y los niveles musicales a alto nivel. Estos dos tipos de características no tienen por qué estar relacionadas: las propiedades acústicas a bajo nivel se obtienen de la señal acústica de la canción, mientras que los niveles musicales a alto nivel están relacionados con la melodía, ritmo o el tono.
A esta diferencia de características se la denomina "brecha semántica". La tendencia actual está enfocada en trabajar para reducir esta brecha: se ha descubierto que el ritmo está relacionado con la estructura métrica y la duración de las notas, o que la dinámica de la canción está relacionada con el volumen, la energía media cuadrática o la intensidad de las notas.

Las herramientas más utilizadas para la extracción de características son:

- MIRToolbox. Un conjunto de herramientas en Matlab.
- OpenSMILE. Desarrollada en C++.
- Essentia 4. Desarrollado en C++, posee clasificadores previamente entrenados.
- PsySound. Librería para uso en Matlab basada en algoritmos psicoacústicos
- Librosa. Biblioteca de código abierto para Python. Especializada en el análisis y procesamiento de audio y música. Es ampliamente utilizada en la comunidad de investigación en musicología, así como en aplicaciones prácticas que involucran el procesamiento y análisis de señales de audio en MIR.

2.1.2.4 Evaluación

La evaluación de los sistemas MER está directamente relacionada con la taxonomía de las emociones: la taxonomía elegida para trabajar en el sistema MER

afecta a las características extraídas y los valores a predecir, y estos a su vez a la evaluación de los sistemas. Hay dos enfoques de sistema MER:

- Sistema de clasificación si la taxonomía elegida sigue un enfoque categórico/discreto. Los parámetros de evaluación de estos sistemas son los mismos que en modelos de clasificación de otras disciplinas: precisión, f-score, precision, etc.
- Sistema de regresión si el enfoque es dimensional/continuo. Los parámetros comúnmente usados para evaluar estos sistemas son: error cuadrático medio, coeficiente de determinación (R2) o coeficiente de correlación de Pearson (ρ).

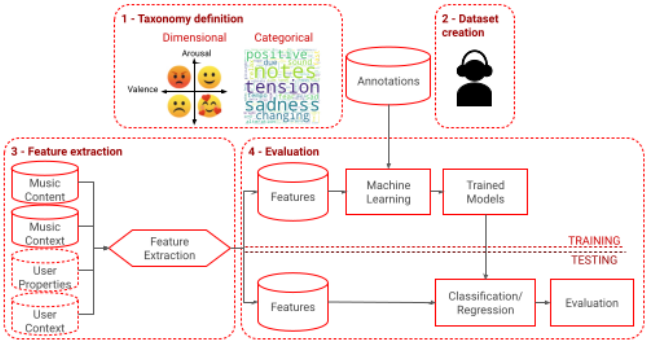

Figure 2: Sistema tradicional MER

2.1.3 Estado futuro MER

En el libro *Music Emotion Recognition* se presentan varias vías de trabajo donde MER puede profundizar:

2.1.3.1 Timbre vocal

Actualmente, en el análisis emocional de una canción, MER pone el foco en la música y en las letras. Sin embargo, la voz contiene muchos matices que pueden resultar especialmente valiosos en los modelos de predicción. Los timbres de voz

(alegre, rasgado, agudo, agresivo, etc.) afectan a la percepción que tiene el oyente de la canción.
Existen distintas formas de estudiar el timbre vocal basadas en el estudio de la señal acústica de la voz y la extracción de descriptores (pitch, energía, coficientes cepstrum, ...).

2.1.3.2 Factores de situación

En el apartado 2.1.1 donde se describen las emociones, se hace referencia a las emociones inducidas. La situación del oyente (momento del día, si está acompañado o solo, etc) afecta directamente a las emociones inducidas. El estudio de la situación puede ser de ayuda para dar contexto acerca del entorno que tiene el oyente, y modificar (o no) la predicción de emociones.

2.1.3.3 Conexión entre enfoques dimensional y categórico

Como se describe en el apartado 2.1.2.1 hay dos enfoques bien diferenciados en el estudio de MER: dimensional/continuo y categórico/discreto. una combinación de los dos enfoques puede resultar eficaz.

2.2 Técnicas de Aprendizaje Máquina

El aprendizaje máquina (Machine Learning) es la ciencia de programar ordenadores para que estos puedan aprender de los datos [7]. Se enfoca en la idea de que los ordenadores puedan "aprender" a partir de la experiencia y los datos: identifican patrones en los datos y predicen o deciden en función de esos patrones. La idea es partir del sistema de etiquetado de emociones (enfoque categórico) y encontrar su lugar en el espacio valencia-activación que propone el enfoque dimensional. Esto

ayudaría a la creación de conjuntos de datos desde el punto de vista del oyente: es más fácil identificar una emoción que situarla dentro del espacio valencia-activación.

2.2.1 Tipos de Machine Learning

Los tipos de Machine Learning se clasifican en diferentes categorías según su naturaleza [7]:

- Grado de supervisión humana durante el entrenamiento. Dentro de esta categoría se encuentran los siguientes sistemas:
  - supervisados,
  - no supervisados,
  - refuerzo del aprendizaje (Reinforcement Learning)
- Si pueden o no aprender de forma incremental. se distinguen dos tipos:
  - sistemas en línea (Batch Learning),
  - sistemas por lotes (Online Learning)
- Si detectan o no patrones en los datos de entrenamiento, construyendo un modelo predictivo. Tipos:
  - basados en instancias,
  - basados en modelos

Estos criterios no son exclusivos. Pueden combinarse de la forma que se quiera.

2.2.2 Sistemas supervisados vs. no supervisados

Los sistemas supervisados de Machine Learning son algoritmos que utilizan datos etiquetados (etiquetas) durante la fase de entrenamiento del sistema. Son utilizados

tanto para modelos de clasificación como para modelos de regresión (predicción de un número dado un paquete de características). Algunos de los algoritmos supervisados más importantes son:

- KNN vecinos
- Regresión lineal
- Regresión logarítmica
- Redes Neuronales

Por otro lado nos encontramos los sistemas no supervisados. Se utilizan para encontrar patrones o estructuras ocultas en datos. A diferencia de los supervisados, estos algoritmos no utilizan ningún etiquetado ni requieren de ninguna salida deseada durante el entrenamiento. Tipos de algoritmos no supervisados:

- Agrupamiento (Clustering)
  - K-medias
  - DBSCAN
  - Análisis de agrupamiento jerárquico (HCA)
- Detección de anomalías
  - One-class SVM
  - Isolation Forest
- Reducción de visualización
  - Análisis de componentes principales (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)

- Aprendizaje de reglas de asociación
  - Apriori
  - Eclat

2.2.3 Batch Learning y Online Learning

Los sistemas Batch Learning son incapaces de aprender de forma incremental. El modelo se entrena utilizando la totalidad del conjunto de datos de entrenamiento de una sola vez, en lugar de hacerlo de manera incremental o en pequeñas partes. Dada la forma de entrenar, estos sistemas son capaces de captar mejor las complejidades y las variabilidades de los datos. Sin embargo, son más vulnerables frente a cambios en los datos puesto que sería necesario volver a entrenar el modelo de nuevo.
Los sistemas Online Learning son lo contrario a los anteriores: se entrenan de forma incremental. Se dividen los datos en pequeñas instancias (o mini-batches) y se alimenta el sistema de forma secuencial. Cada aprendizaje es rápido y barato. Se utiliza para sistemas donde se reciben datos de forma continua, o cuando se requiere una gran cantidad de datos para el entrenamiento. En estos modelos es importante el parámetro de tasa de aprendizaje (learning rate): Una tasa de aprendizaje alto hará que el sistema se adapte bien a cambios en los datos, pero también será más susceptible a olvidar más rápido datos antiguos.

2.2.4 Basados en instancias y Basados en modelos

Esta clasificación hace referencia a cómo los sistemas de aprendizaje se generalizan. Por un lado están los basados en instancias: el sistema aprende los ejemplos de memoria y luego generaliza a nuevos casos comparándolos con los ejemplos aprendidos (o un subconjunto de ellos). Es decir, utiliza una medida de similitud. Estos sistemas tiene un buen rendimiento con datos entrenados, pero no es suficiente para un sistema de Machine Learning: el verdadero objetivo es tener un buen rendimiento

en instancias nuevas.
Aquí es donde se encuentran los sistemas basados en modelos, cuya forma de generalizar es a partir de modelos que hagan predicciones. Estos sistemas tienen mucho mejor rendimiento frente a instancias de datos muy ruidosas (gran cantidad de datos aleatorios). Para conseguir buenos resultados con estos sistemas se deben seguir algunos pasos como: selección del modelo (elegir el mejor modelo que se adapte a nuestro sistema), ajuste del modelo (elección correcta de los parámetro o hiperparámetros), o entrenamiento el modelo.

2.2.5 Validación cruzada

La validación cruzada (o *cross-validation*) es una técnica fundamental para la evaluación y selección de modelos en el contexto del aprendizaje estadístico. Se utiliza para evaluar la capacidad predictiva de un modelo, permitiendo estimar cómo de bien funcionará el modelo en datos no vistos. Este procedimiento es crucial para evitar el sobreajuste (*overfitting*) y para comparar de manera objetiva el rendimiento de diferentes modelos o ajustes de hiperparámetros.
Para entender mejor esta técnica, se parte de la diferencia entre la tasa de error de test y la tasa de error de entrenamiento. La tasa de error de test se calcula fácilmente comparando los datos obtenidos en la predicción (se introducen en el modelo entrenado las variables independientes, para obtener la predicción de la variable objetivo) con los datos reales. Sin embargo, la tasa de error de entrenamiento no se puede calcular (sin emplear una técnica precisa para ello), al ser el mismo conjunto de datos (datos de entrenamiento) el mismo que usaríamos para entrenar y validar [6].
Existen diferentes tipos de *cross-validation*, pero el más utilizado es K-Fold. Parte de la idea de dividir el conjunto de datos de entrenamiento en k particiones, de las cuales se usarán *k-1* para entrenamiento y la restante para la validación. Este proceso se repite de forma iterativa, de manera todas las particiones se han usado una vez para validar y *k-1* veces para entrenar [9].

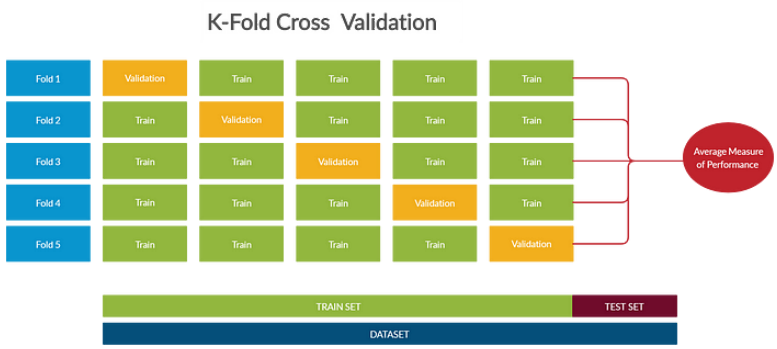

Figure 3: Ejemplo de funcionamiento de *cross-validation* para $k = 5$

En cada iteración se calculan las métricas de rendimiento, para finalmente, hacer un promedio y obtener una estimación total del rendimiento del modelo. A pesar de que esta técnica puede resultar computacionalmente intensiva, proporciona una estimación más fiable del modelo utilizando todos los datos de entrenamiento de una forma más eficiente.

2.3 Modelos de Regresión

Los modelos de Regresión son técnicas que se utilizan para predecir un valor numérico continuo basado en una o más variables independientes. Para ello, se modela el efecto de un conjunto de variables explicativas sobre una varibale de interés primario [5].
Los conceptos fundamentales de los modelos de Regresión son:

- Variable Dependiente (variable objetivo, respuesta o de interés primario): es la variable que se desea predecir. Esta puede ser continua, binaria, categórica o de recuentos

- Variables explicativas: también llamadas covariables, variables independientes o regresoras. Son las que se utiliza para predecir la variable dependiente. Existen varios tipos de estas variables como continuas, binarias, o categóricas.

En modelos complejos también es posible incluir escalas temporales, variables para describir la distribución espacial o la ubicación geográfica, o indicadores grupales.

- Tipo de modelo: dependerá principalmente del tipo de variable respuesta, y del tipo de variables independientes

Una característica principal de los modelos de regresión es que la relación entre la variable independiente y las variables independientes no es una función determinista, si no que muestra errores. Esto implica que la respuesta y es una variable aleatoria, cuya distribución depende de las variables independientes. Un ejemplo claro sería: sabiendo la altura de los padres no se puede predecir exactamente la altura de los hijos, solo estimar una media y el grado de dispersión. A esta desviación del valor esperado se la denomina ϵ (componente aleatorio o estocástico) (por eso es muy importante estudiar la importancia de las covariables en el valor medio de la variable objetivo). La función resultante es un modelo condicional donde los valores de y (variable objetivo) están condicionados por las covariables $(x_1, x_2, x3, ...)$ y el componente aleatorio ϵ:

$$y = E(y|x_1, ..., x_k) = f(x_1, ..., x_k) + \epsilon \tag{1}$$

2.3.1 Árboles de decisión

Los árboles de decisión son una técnica de Machine Learning y estadística utilizada tanto para casos de clasificación como para casos regresión. Son modelos predictivos que representan decisiones y sus posibles consecuencias, incluidas las probabilidades de diferentes resultados. Son conceptualmente simples, pero poderosos
El principio de los árboles de regresión es segmentar el espacio características en una serie de regiones simples. El conjunto de reglas de división utilizadas para segmentar el espacio se pueden resumir en un árbol, de ahí el término [6].

En el libro *The elements of statistical learning: data mining, inference, and prediction* se explican los modelos de decisión con un ejemplo simple: imaginemos un problema de regresión con una respuesta continua Y y los inputs X_1 y X_2. Primero se divide el espacio en dos regiones. Se elige la variable y el punto de divisón para lograr el mejor ajuste. Luego, una o ambas regiones se dividen en dos regiones más. Continúa el proceso hasta que se aplica alguna regla de detención.

2.3.2 Bosques aleatorios

Los árboles aleatorios (random forests) son un método de aprendizaje para clasificación y regresión que consiste en construir una multitud de árboles de decisión (ver 2.3.1) en la etapa de entrenamiento, y así obtener el modo de las clases (clasificación) o la media de las predicciones (regresión) de los árboles individuales.
La idea de los bosques aleatorios es reducir la varianza del sistema, reduciendo la correlación entre los árboles (sin incrementar en exceso la varianza). Esto es posible gracias a una selección aleatoria de variables independientes durante el proceso de crecimiento de los árboles [9].
La forma en que se construye los bosques aleatorios es la siguiente:

1. Se elige el número B de árboles de decisión que formará el bosque
2. Se hace crecer los árboles del bosque repitiendo de forma recursiva los siguientes pasos
   (a) Del conjunto de datos de entrenamiento, se elige un subconjunto aleatorio de las características (generalmente $\sqrt{\rho}$ variables si hay ρ en total).
   (b) De este subconjunto, se elige la mejor variable .
   (c) Se dividen los nodos en dos nodos hijos.
3. Se obtienen los datos del conjunto de árboles.
4. Se hace la predicción.

2.3.3 Regresión Lineal

La regresión lineal es uno de los métodos más básicos y utilizados de regresión. Aquí tienes una definición más detallada. Como se explica en el apartado anterior, en regresión existe una relación entre la variable objetivo y las variables independientes a través de la función $f(x1, ..., xk)$. Esta función no es exacta al estar afectada por el ruido ϵ. El objetivo es estimar función desconocida f, es decir, separar el componente sistemático de f del ruido aleatorio ϵ, asumiendo en este caso un acercamiento lineal [5]. La ecuación resultante es la siguiente:

$$y_i = \beta_0 + \beta_1 x_i, ..., \beta_k x_i + \epsilon_i = x_i' \beta + \epsilon \tag{2}$$

Donde los parámetros $\beta_0, \beta_1, ..., \beta_k$ son desconocidos y deben ser estimados. El parámetro β_0 respresenta la intersección (cuando el valor de x es 0).

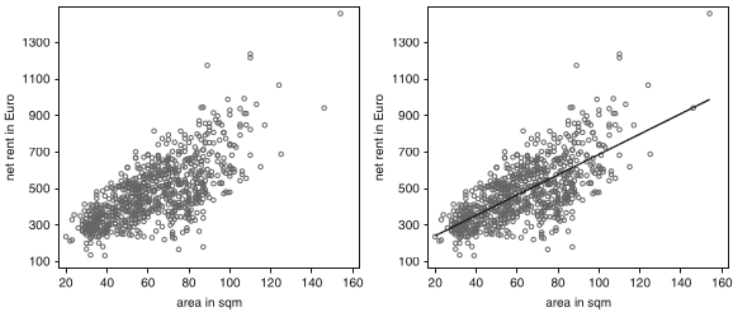

Figure 4: Regresión Lineal

El objetivo de la regresión lineal es encontrar la mejor línea recta que minimiza la suma de los errores cuadrados (diferencias entre los valores reales y los valores predichos).

2.3.4 Regresión ridge

La regresión ridge es una variante de la regresión lineal estándar. Modifica la estimación de los coeficientes del modelo para mejorar la precisión: introduce una penalización que reduce los coeficientes de las variables independientes. Para lograr esta penalización se agrega un término de regulación que es proporcional a la suma de los cuadrados de los coeficientes. La fórmula de la regresión ridge es la siguiente [9]:

$$\hat{\beta}^{ridge} = argmin \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j^2) \tag{3}$$

Donde:

- $\hat{\beta}^{ridge}$ es el término de regularización ridge
- y_i es la variable respuesta para la observación i
- x_{ij} es el valor de la variable independiente j para la observación i
- β_j son los coeficientes del modelo

La regresión ridge se utiliza cuando hay muchas variables independientes y se quiere mejorar la precisión del modelo lineal: los coeficientes puede estar mal determinados y mostrar una varianza muy alta. Al imponer una restricción de tamaño de los coeficientes se soluciona este problema [5].

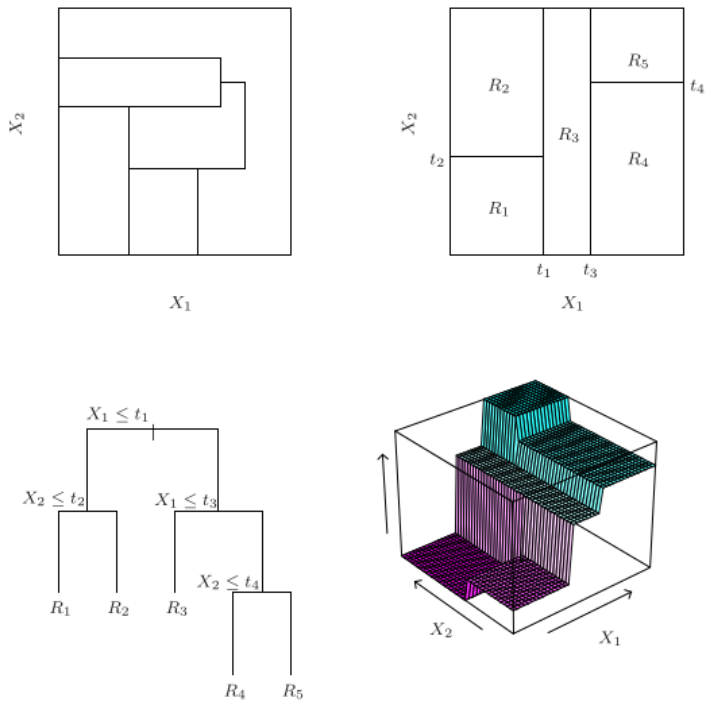

Figure 5: Funcionamiento de un árbol de decisión

Los métodos basados en árboles de decisión son simples, y útiles para la interpretación. Sin embargo, normalmente no son competitivos con los mejores enfoques de aprendizaje supervisado en términos de predicción [6].

3 Experimentos

En este capítulo se describe el proceso del desarrollo de la aplicación MER. Primeramente se describe el conjunto de datos utilizado en el experimento, y cómo han sido manipulados los datos para que sean utilizados por el sistema. Por último, se describen las técnicas de Machine Learning utilizadas en la aplicación.

3.1 Conjunto de datos

El conjunto de datos elegido para el experimento es DEAM dataset - The MediaEval Database for Emotional Analysis of Music [2]. El conjunto de datos DEAM consta de 1802 extractos y canciones libres de derechos que provienen de varias fuentes: freemusicarchive.org (FMA), jamendo.com, y el conjunto de datos medleyDB.
Existen tres tipos de información claramente diferenciables por cada pista de audio del conjunto de datos: características, valores de valencia y activación, y metadatos.

3.1.1 Características

Los datos presentan un conjunto de características extraídas de la librería openSMILE [4]. Cada pista de audio está representada en un fichero CSV, donde las pistas se dividen en ventanas de 500ms. En cada ventana se muestran dos valores de cada una de las características: la media y la desviación estándar Las características presentes en el conjunto de datos son [1]:

- Energía
- Intensidad de fotograma/sonoridad (aproximación)

- Espectros de banda crítica (Mel/Bark/Octave, filtros de enmascaramiento triangulares)
- Mel-/Bark-Frequency-Cepstral Coefficients (MFCC)
- Espectros auditivos
- Sonoridad aproximada a partir de espectros auditivos.
- Coeficientes perceptivos lineales predictivos (PLP)
- Coeficientes cepstrales predictivos lineales perceptivos (PLP-CC)
- Coeficientes predictivos lineales (LPC)
- Pares espectrales de líneas (LSP, también conocido como LSF)
- Frecuencia fundamental (mediante el método ACF/Cepstrum y mediante sumación subarmónica (SHS))
- Probabilidad de emisión de voz a partir del pico del espectro ACF y SHS
- Calidad de voz: Jitter y Shimmer
- Frecuencias de formantes y anchos de banda.
- Tasa de cruce por cero y media
- Características espectrales (energías de banda arbitrarias, puntos de caída, centroide, entropía, maxpos, minpos, varianza (= dispersión), asimetría, curtosis, pendiente)
- Nitidez psicoacústica, armonía espectral.
- CROMA (espectros de semitonos deformados en octavas) y funciones CENS (CROMA suavizado y normalizado de energía)
- Funciones derivadas de CHROMA para reconocimiento de acordes y claves
- F0 Relaciones de armónicos

3.1.2 Valencia y activación

Los datos de valencia y activación se muestran en el conjunto de datos como valores continuos dentro del rango [1, 9].
Para la extracción de estos valores, se eligieron 45 segundos de cada canción de forma aleatoria (posteriormente se descartaron los 15 primeros segundos debido a la inestabilidad de las anotaciones al inicio de los clips). Los 45 segundos se dividieron en ventanas de 500ms, y por cada ventana se anotaron 10 valores para de valencia y activación.
En el conjunto de datos están presentes todas estas anotaciones, junto con los valores de media y desviación estándar de cada canción. Serán estos últimos lo que se usarán en el desarrollo del trabajo.

3.1.3 Metadatos

En el conjunto de datos de DEAM [2] también hay información acerca de las canciones: título de la canción, artista, álbum, géneros musicales y etiquetas.

3.2 Desarrollo

En este apartado se describen los pasos a seguir en el desarrollo del sistema MER. El sistema está compuesto, en primer lugar, por un procesado de datos: se toman los archivos proporcionados por el conjunto de datos y se manipulan para crear las variables de las que están formados los modelos.
Por último, se documenta todo el desarrollo seguido para el diseño de la aplicación MER de predicción de emociones. En los siguientes apartados se procede a explicar cómo se han diseñado y entrenado los modelos para cada uno de los algoritmos. Los algoritmos de Machine Learning elegidos son:

- Árbol de decisión
- Bosques aleatorios
- Regresión lineal
- Regresión ridge

La taxonomía de emociones elegida para este experimento es la continua/dimensional (ver apartado 2.1.2.1), donde las emociones están representadas por dos valores: valencia y activación. Visto desde el punto de vista de Machine Learning, se presenta un sistema de predicción donde hay dos variables objetivo. Por lo tanto, es necesario dividir el problema en dos sistemas diferentes. El resultado son dos modelos de predicción diferentes para cada algoritmo (uno para valencia y otro para activación). Tanto la memoria como el código están disponibles en este repositorio Github `https://github.com/viglescue/tfg`. El código fuente de la aplicación MER está dividido en dos archivos: *data-recap.ipynb* con todo el desarrollo correspondiente al procesamiento de datos y archivos del conjunto de datos, y *MER-ML-models.ipynb* con el desarrollo de los diferentes algoritmos para la predicción de emociones. El experimento al completo está desarrollado en lenguaje Python, utilizando cuadernos Jupyter en el entorno de Anaconda.

3.2.1 Extracción y manipulación de los datos

El conjunto de datos DEAM [2] organiza los datos de la siguiente forma:

- Documentos CSV (uno por cada pista de audio) donde se encuentran los valores de las características de cada canción. En cada documento CSV aparece el valor de cada característica en intervalos de 500ms.
- Un archivo CSV con los valores de media y desviación para valencia y activación.

La idea es recopilar todos los datos en una tabla para manipularlos mejor. Para ello se ha empleado la librería Pandas [10] para su uso en lenguage Python.
Para empezar a construir la tabla, primeramente, se obtienen los nombres de las características a partir de uno de los archivos CSV que las contienen. Dado que los valores de características están disponibles por intervalos de 500ms, se ha calculado la media de los valores de cada característica (por cada pista de audio) con el fin de obtener un solo valor por característica y canción. Estos valores serán las variables independientes que se usarán posteriormente en los modelos de ML.
De otro archivo, se han recopilado los valores de media y activación de cada pista de audio. Estos serán las variables dependientes (o variables objetivo) de los modelos. El resultado ha sido el documento *recap-data.csv*, el cual contiene todas las variables necesarias para construir los sistemas MER a partir de los diferentes modelos de ML.

pcm_fftMag_mfcc_sma_de[13]_stddev	pcm_fftMag_mfcc_sma_de[13]_amean	pcm_fftMag_mfcc_sma_de[14]_stddev	pcm_fftMag_mfcc_sma_de[14]_amean	valence	arousal
1.848157	0.000444	1.709386	0.000424	5.5	4.6
1.238022	0.000212	1.116202	0.000100	2.7	3.7
2.053561	0.001272	1.702499	0.001127	5.0	4.3
1.802331	-0.000547	1.680737	-0.001378	3.5	2.8
1.909060	0.001739	1.753582	-0.001443	5.1	6.1
...	...	...	...	...	...
1.671897	-0.000347	1.513882	-0.000573	4.8	6.5
1.501074	-0.002797	1.444453	-0.005020	3.9	2.5
2.564815	-0.000453	2.606689	-0.003860	4.0	6.1
2.023299	0.002849	1.714624	0.000608	5.5	5.1
1.759458	-0.003256	1.899548	-0.003618	6.2	6.8

Figure 6: Recopilación de datos

Una vez recopilados los datos en la tabla que se muestra en la Figura 6, se definen las variables independientes y las variables objetivo del sistema. Como se comenta en el apartado anterior, esta aplicación MER se compone de dos variables objetivo. Por lo tanto, las variables se dividen en: un conjunto de variables independientes, una variable objetivo para la valencia, y otra variable objetivo para la activación.

```
1 explicative = matrix.drop(columns=['valence', 'file', 'Unnamed: 0', 'arousal'])
2 objective_valence = matrix.valence
3 objective_arousal = matrix.arousal
4
```

Figure 7: Definición de variables explicativas y variables objetivo

Un último procesado de datos antes de empezar con el diseño de los modelos es la división del conjunto de datos entre datos de entrenamiento y datos de test. Para

ello se ha utilizado la función *train_test_split* de la librería *scikit-learn* de Python [11]. Esta función permite la división de datos de forma aleatoria, de forma que hay todo tipo de datos tanto en el conjunto de entrenamiento como en el de test. Los porcentajes elegidos para este trabajo son: 90% para el conjunto de datos para entrenamiento, y 10% para test.
En resumen, los sistemas quedan definidos de la siguiente forma:

- Sistema para la predicción de valencia (Figura 8):
  - Variables de entrenamiento: *X_train_val*, *y_train_val*
  - Variables de test: *X_test_val*, *y_test_val*
  - Porcentaje de datos: 90
- Sistema para la predicción de activación (Figura 9):
  - Variables de entrenamiento: *X_train_val*, *y_train_val*
  - Variables de test: *X_test_ar*, *y_test_val*
  - Porcentaje de datos: 10

```
1 from sklearn.model_selection import train_test_split
2 X_train_val, X_test_val, y_train_val, y_test_val = train_test_split(explicative, objective_valence, test_size=0.10, random_state=42)
```

Figure 8: Definición de variables para valencia

```
1 from sklearn.model_selection import train_test_split
2 X_train_ar, X_test_ar, y_train_ar, y_test_ar = train_test_split(explicative, objective_valence, test_size=0.10, random_state=42)
```

Figure 9: Definición de variables para activación

3.2.2 Validación cruzada y *GridSearchCV*

En el apartado 2.2.5 se describe el concepto de *cross-validation* y su importancia en el diseño de modelo de *Machine Learning*. En este experimento se ha implementado *cross-validation* mediante el uso de la función *GridSearchCV* de la librería *scikit-learn* de Python [11] en todos los modelos.Se ha utilizado esta técnica

para encontrar la mejor combinación de hiperparámetros en los modelos.
GridSearchCV obtiene como entrada un espacio de valores para cada parámetro del modelo, para posteriormente, evaluar todas las combinaciones de valores posibles. La salida de la función será el valor de cada parámetro del modelo que tenga mejor precisión (*score*).
Los parámetros de entrada para *GridSearchCV* son:

- *estimator*: algoritmo de Machine Learning utilizado.
- *param_grid*: espacio de valores para los parámetros del algoritmo.
- *scoring*: estrategia para evaluar el desempeño del modelo.
- *n_jobs*: número de trabajos que se ejecutarán en paralelo.
- *refit*: reajuste del estimador.
- *cv*: divisiones para la validación cruzada (*k-folds*, ver 2.2.5).
- *verbose*: controla los mensajes de salida.
- *pre_dispatch*: controla el consumo de CPU.
- *error_score*: valor a asignar a la puntuación si se produce un error en el ajuste del estimador.
- *return_train_score*: si se incluyen o no las puntuaciones de entrenamiento.

Los parámetros de entrada para *GridSearchCV* se han configurado como:

- *estimator*: la función de *scikit-learn*, *DecisionTreeRegressor()*.
- *param grid*: la variable *grid_hp_tree_reg* definida previamente.
- *scoring*: *None* (valor por defecto).
- *n_jobs*: valor '-1', para utilizar todo el procesador.

- *refit*: '*True*' (valor por defecto).
- *cv*: 5 *k-folds*. Cinco particiones es lo más común para *cross-validation*.
- *verbose*: '10', para obtener toda la información.
- *pre_dispatch*: '*2*n_jobs*' (valor por defecto).
- *error_score*: 'np.nan' (valor por defecto).
- *return_train_score*: '*Falso*' (valor por defecto).

```
15 tree_regr_model_valence = GridSearchCV(
16     estimator = DecisionTreeRegressor(),
17     param_grid = grid_hp_tree_reg,
18     cv = 5,
19     n_jobs = -1,
20     verbose = 10,
21 )
```

Figure 10: Condiguración de *GridSearchCV* para el árbol de decisión

3.2.3 Árbol de decisión

Para la creación del árbol de decisión, se ha utilizado la función *DecisionTreeRegressor* del módulo *tree*, perteneciente a la librería *scikit-learn* de Python [11].
El ajuste de hiperparámetros del modelo se ha realizado mediante una validación cruzada exhaustiva utilizando *GridSearchCV* (ver apartado 3.2.2). Esta función está presente en el módulo *model_selection* de *sckit-learn* [11]).

```
1 from sklearn.model_selection import GridSearchCV
2 from sklearn.tree import DecisionTreeRegressor
3
```

Figure 11: Uso de las funciones *DecisionTreeRegressor* y *GridSearchCV* de *sckit-learn*

Mediante la variable *grid_hp_tree_reg* mostrada en la Figura 12 se ha definido un espacio de valores de condiguración del árbol de decisión. Esta variable será el valor

de entrada del parámetro *param_grid* en el proceso de validación cruzada exhaustiva de *GridSearchCV*.

```
grid_hp_tree_reg = {
    'criterion': ['squared_error', 'friedman_mse', 'absolute_error'],
    'splitter': ['best', 'random'],
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 4],
    'min_samples_leaf': [1, 3],
    'max_features': [None, 'sqrt', 'log2'],
    'random_state': [None, 3, 5],
    'max_leaf_nodes': [None, 3]
}
```

Figure 12: Variable *grid_hp_tree_reg* para la validación cruzada de *DecisionTreeRegressor*

La validación cruzada da como resultado la siguiente combinación de hiperparámetros para el árbol de decisión, la más precisa dentro del espacio de valores de los parámetros contenidos en la variable *grid_hp_tree_reg*:

- Para valencia:

```
tree_regr_model_valence.best_params_
```

```
{'criterion': 'squared_error',
 'max_depth': 3,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'random_state': None,
 'splitter': 'best'}
```

Figure 13: Hiperparámetros del el árbol de decisión para la predicción de valencia

- Para activación:

```
tree_regr_model_arousal.best_params_
```

```
{'criterion': 'squared_error',
 'max_depth': 3,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'random_state': None,
 'splitter': 'best'}
```

Figure 14: Hiperparámetros del el árbol de decisión para la predicción de activación

Una vez obtenidos los hiperparámetros se construye el algoritmo de árbol de decisión. Después se entrena el modelo con las variables de entrenamiento (*X_train_val* y *y_train_val* para valencia, y *X_train_ar* y *y_train_ar* para activación).
En esto punto del experimento, el árbol de decisión está listo para ser utilizado con los datos de test (*X_test _val* y *X_test_ar*). La precisión y la fiabilidad del modelo será evaluada en el apartado 4.

3.2.4 Bosques aleatorios

Para la creación de bosques aleatorios, se ha utilizado la función *LinearRegression* del módulo *linear_model*, perteneciente a la librería *scikit-learn* de Python [11].
El ajuste de hiperparámetros del modelo se ha realizado mediante una validación cruzada exhaustiva utilizando *GridSearchCV* (función presente en el módulo *model_selection* de *sckit-learn* [11]).

Uno de los parámetro de entrada de *GridSearchCV* es *param_grid*. Este parámetro lee una selección de valores para cada hiperparámetro de *DecisionTreeRegressor*. Estos valores se recogen en la variable *grid-hp-tree-reg*.

La validación cruzada dará como resultado la combinación de parámetros de entrada donde la función *DecisionTreeRegressor* tenga una mayor precisión (*score*).
Los hipermetamorfosis elegidos por la validación cruzada son:

- Para valencia:
- Para activación:

Una vez obtenidos los hiperparámetros se construye el algoritmo de bosque aleatorio. Posteriormente, se entrena el modelo con las variables de entrenamiento (*X_train_val* y *y_train_val* para valencia, y *X_train_ar* y *y_train_ar* para activación).
En esto punto del experimento, el bosque aleatorio está listo para ser utilizado con

los datos de test (*X_test_val* y *X_test_ar*). La precisión y la fiabilidad del modelo será evaluada en el apartado 4.

3.2.5 Regresión Lineal

Para la implementación de los modelos de regresión lineal, se ha utilizado la función *LinearRegression* del módulo *linear model*, perteneciente a la librería *scikit-learn* de Python [11].
Para el ajuste de hiperparámetros del modelo, se ha utilizado *GridSearchCV* (función presente en el módulo *model selection* de *sckit-learn* [11]) para realizar una validación cruzada exhaustiva.

```
:  1 from sklearn.model_selection import GridSearchCV
   2 from sklearn.linear_model import LinearRegression
   3
```

Figure 15: Uso de las funciones *LinearRegression* y *GridSearchCV* de *sckit-learn*

Mediante la variable *grid-hp-linear-reg* mostrada en la Figura 16 se ha definido un espacio de valores de condiguración del árbol de decisión. Esta variable será el valor de entrada del parámetro *param_grid* en el proceso de validación cruzada exhaustiva de *GridSearchCV*.

```
 4 grid_hp_linear_reg = {
 5     'fit_intercept': [True, False],
 6     'copy_X': [True, False],
 7     'n_jobs': [None, -1],
 8     'positive': [True, False]
 9 }
10
```

Figure 16: Variable *grid_hp_linear_reg* para la validación cruzada de *LinearRegression*

La validación cruzada da como resultado la combinación de parámetros de en-

trada donde la función *LinearRegression* tiene una mayor precisión. Los hiperparámetros obtenidos son:

- Para valencia:

```
]: 1 linear_reg_model_valence.best_params_

]: {'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'positive': True}
```

Figure 17: Hiperparámetros de la regresión lineal para la predicción de valencia

- Para activación:

```
9]: 1 linear_reg_model_arousal.best_params_

9]: {'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'positive': True}
```

Figure 18: Hiperparámetros de la regresión lineal para la predicción de activación

Una vez obtenidos los hiperparámetros se construye el algoritmo de regresión lineal. Después se entrena el modelo con las variables de entrenamiento (*X_train_val* y *y_train_val* para valencia, y *X_train_ar* y *y_train_ar* para activación).
En esto punto del experimento, el modelo de regresión lineal está listo para ser utilizado con los datos de test (*X_test_val* y *X_test_ar*). La precisión y la fiabilidad del modelo será evaluada en el apartado 4.

3.2.6 Regresión ridge

Para la implementación de los modelos de regresión ridge, se ha utilizado la función *LinearRegression* del módulo *linear_model*, perteneciente a la librería *scikit-learn* de Python [11].
Para el ajuste de hiperparámetros del modelo, se ha utilizado *GridSearchCV* (función presente en el módulo *model selection* de *sckit-learn* [11]) para realizar una validación cruzada exhaustiva.

Uno de los parámetro de entrada de *GridSearchCV* es *param_grid*. Este parámetro

lee una selección de valores para cada hiperparámetro de *LinearRegression*. Estos valores se recogen en la variable *grid_hp_linear_reg*.

La validación cruzada dará como resultado la combinación de parámetros de entrada donde la función *LinearRegression* tenga una mayor precisión (*score*). Los hipermetamorfosis elegidos por la validación cruzada son:

- Para valencia:

```
: 1 ridge_reg_model_valence.best_params_

: {'alpha': 5.0,
   'max_iter': None,
   'positive': False,
   'random_state': None,
   'solver': 'svd'}
```

Figure 19:

- Para activación:

Una vez obtenidos los hiperparámetros se construye el algoritmo de regresión ridge. Después se entrena el modelo con las variables de entrenamiento (*X_train_val* y *y_train_val* para valencia, y *X_train_ar* y *y_train_ar* para activación).
En esto punto del experimento, el modelo de regresión ridge está listo para ser utilizado con los datos de test (*X_test_val* y *X_test_ar*). La precisión y la fiabilidad del modelo será evaluada en el apartado 4.

4 Resultados

En este experimento se presentan algoritmos de regresión de Machine Learning. La evaluación de este tipo de modelos se realiza mediante el cálculo de metricas de evaluación. el principio básico de esta evaluación es comparar las variables objetivo predecidas por el sistema a partir de las variables independientes de test (*X_test_val* y *X_test_ar*), con las variables objetivo reales (*y_test_val* y *y_test_ar*). Las métricas elegidas son:

- Mean absolute error (MAE)
- Mean squared error (MSE)
- R2 Score
- Mean squared log error (MSLE)
- Mean absolute percentage error (MAPE)

Para el obtener estas métricas se ha utilizado la librería de *scikit-learn* de Python, y las funciones correspondientes para cada caso.

4.1 Mean absolute error (MAE)

El *Mean absolute error* (o error absoluto medio) se calcula como el promedio de las diferencias absolutas entre dos variables continuas. La fórmula de MAE es [14]:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - x_i| \tag{4}$$

donde

- n es el número de observaciones,
- y_i es el valor real de la observación,
- x_i es el valor predictivo

4.2 Mean squared error (MSE)

4.3 R2 Score

4.4 Mean squared log error (MSLE)

4.5 Mean absolute percentage error (MAPE)

5 Conclusiones

6 Bibliografía

References

[1] URL: https://audeering.github.io/opensmile/about.html#capabilities.

[2] Anna Alajanki, Yi-Hsuan Yang, and Mohammad Soleymani. "Benchmarking music emotion recognition systems". In: *PLOS ONE* (2016). under review.

[3] Tuomas Eerola and Jonna K Vuoskoski. "A review of music and emotion studies: Approaches, emotion models, and stimuli". In: *Music Perception: An Interdisciplinary Journal* 30.3 (2012), pp. 307–340.

[4] Florian Eyben et al. "Recent Developments in openSMILE, the Munich Open-source Multimedia Feature Extractor". In: *Proceedings of the 21st ACM International Conference on Multimedia*. MM '13. Barcelona, Spain: ACM, 2013, pp. 835–838. ISBN: 978-1-4503-2404-5. DOI: 10.1145/2502081.2502224. URL: http://doi.acm.org/10.1145/2502081.2502224.

[5] Ludwig Fahrmeir et al. *Regression models*. Springer, 2013.

[6] James Gareth et al. *An introduction to statistical learning: with applications in R*. Spinger, 2013.

[7] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* " O'Reilly Media, Inc.", 2022.

[8] Juan Sebastián Gómez-Cañón et al. "Music Emotion Recognition: Toward new, robust standards in personalized and context-sensitive applications". In: *IEEE Signal Processing Magazine* 38 (6 2021), pp. 106–114. DOI: 10.1109/MSP.2021.3106232.

[9] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[10] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[12] Jonathan Posner, James A Russell, and Bradley S Peterson. "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology". In: *Development and psychopathology* 17.3 (2005), pp. 715–734.

[13] Markus Schedl, Emilia Gómez, Julián Urbano, et al. "Music information retrieval: Recent developments and applications". In: *Foundations and Trends® in Information Retrieval* 8.2-3 (2014), pp. 127–261.

[14] Wikipedia. *Error absoluto medio — Wikipedia, La enciclopedia libre*. [Internet; descargado 27-septiembre-2022]. 2022. URL: https://es.wikipedia.org/w/index.php?title=Error_absoluto_medio&oldid=146234007.

[15] Y.H. Yang and H.H. Chen. *Music Emotion Recognition*. Multimedia Computing, Communication and Intelligence. CRC Press, 2011. ISBN: 9781439850473. URL: https://books.google.es/books?id=qnjRBQAAQBAJ.