ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

PREDICCIÓN DE EMOCIONES A PARTIR DE CARACTERÍSTICAS DE AUDIO MUSICAL UTILIZANDO TÉCNICAS DE APRENDIZAJE MÁQUINA

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

A mis padres, que se dejaron la vida.

A Pedro y Aida, que ayudaron a que fuera posible.

A Yaiza, porque la mitad es suyo.

A lo compañeros, por compartir apuntes y horas de biblioteca.

A mi tutora Rebeca, por darme la oportunidad de hacer este trabajo juntos.

A todos los que en algún momento me echaron una mano para seguir estudiando: familia, amigos, compañeros de trabajo y jefes.

Resumen

Este Trabajo de Fin de Grado tiene como objetivo desarrollar una aplicación de reconocimiento de emociones en la música (*Music Emotion Recognition, MER*) utilizando técnicas de aprendizaje máquina (*Machine Learning*). La aplicación está diseñada para predecir las emociones musicales a través de las características acústicas de las canciones.

Antes de comenzar con el experimento, se introduce el campo de estudio de MER comentando el marco teórico y su estado actual y futuro, para también poner el foco en las emociones. Para la realización del trabajo es importante conocer qué son las emociones, qué tipos hay, y por qué se ha decidido utilizar para este trabajo las emociones percibidas representadas en un espacio bidimensional formado por valencia y activación.

Una vez asentadas las bases teóricas del trabajo, se da paso a comentar las diferentes técnicas de aprendizaje automático empleadas en el experimento. Se usarán cuatro algoritmos diferentes de *Machine Learning*: árbol de decisión, bosque aleatorio, regresión lineal y regresión *ridge*. Para mejorar la precisión de los modelos de predicción, se han ajustado los hiperparámetros mediante el uso de validación cruzada (*cross-validation*).

Para elegir los modelos más adecuados para el sistema se han calculado las tasas de error de los algoritmos y se ha elegido la menor de ellas. El resultado del trabajo es un sistema de MER capaz de predecir (con margen de error aceptable para este propuesto) los valores bidimensionales de los que se componen las emociones, y por lo tanto, capaz de predecir las emociones percibidas de las canciones.

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 [4].

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 [19].

Tradicionalmente, la clasificación de música ha estado basada en catálogos de metadatos (artista, album, título de la canción, ...) [19]. 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 [19].

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 de MER.
- Examinar técnicas de Aprendizaje Máquina (*Machine Learning*) para la detección de emociones.
- Usar lenguage Python para:
  - procesar 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 cinco 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 al campo de estudio de MER,
- una descripción de *Machine Larning* y los diferentes tipos de algoritmos que hay,
- y una explicación de los modelos de regresión y la definición de los usados en el trabajo.

3. El capítulo de *Experimentos* dividido en:
   - descripción de los datos utilizados,
   - y 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 [9].

MER es una de las disciplinas de alto nivel más desafiantes en *Music Information Retrieval* (MIR). MIR es un campo de investigación que se centra en la extracción y el manejo de características significativas de la música (a partir de la señal acústica), 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 [14].

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 [4].

Uno de los más comunes usado en psicología es el modelo dimensional propuesto por Russell [13]. 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 energía o activación de la valencia). Cada emoción es una representación lineal de la combinación de estas dos dimensiones.

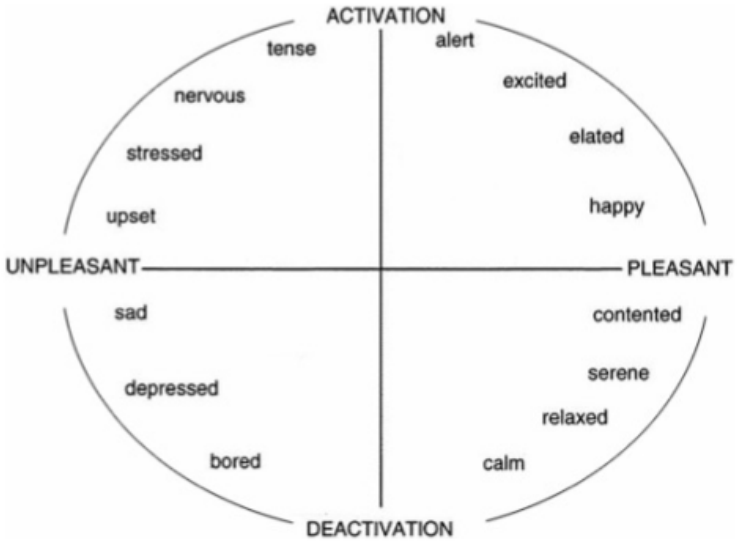

Figura 1: Modelo dimensional de Russell [13]

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.

Yang y Chen en su libro *Music Emotion Recognition* [19], exponen 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 en MER se divide en cuatro bloques [9]:

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 [13]

La elección de la taxonomía es la decisión más importante para diseñar un conjunto de datos. Define el nivel de precisión emocional, por lo que su elección marcará el desarrollo del sistema de MER.

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 de 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 de 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, 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).

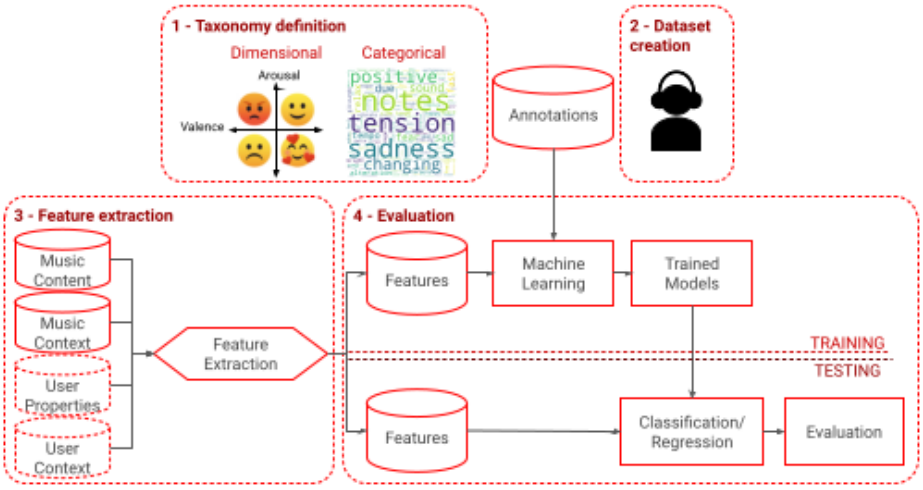

Figura 2: Sistema tradicional MER [9]

2.1.3. Estado futuro MER

En el libro *Music Emotion Recognition* [19] 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, coeficientes cepstrum, etc).

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 *Machine Learning*

El *Machine Learning* es la ciencia de programar ordenadores para que estos puedan aprender de los datos [8]. 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.

2.2.1. Tipos de *Machine Learning*

Los tipos de Machine Learning se clasifican en diferentes categorías según su naturaleza [8]:

- 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
- Regresión lineal
- Regresión logística
- 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 dimensionalidad

    - 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 no 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 alta 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 *Machine Learning* 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 del 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) que usaríamos para entrenar y validar [7].

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 (ver Figura 3), de las cuales se usarán *k-1* para entrenamiento y la restante para la validación. Este proceso se repite de forma iterativa, de manera que todas las particiones se han usado una vez para validar y *k-1* veces para entrenar [10].

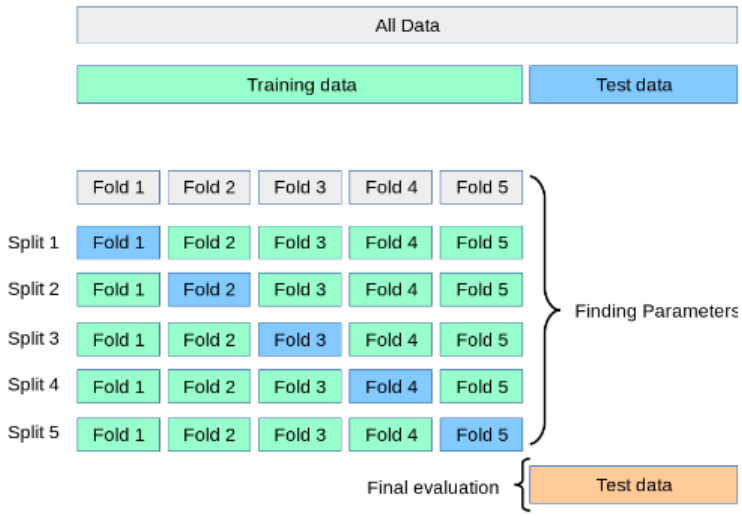

Figura 3: Ejemplo de funcionamiento de *cross-validation* para $k = 5$ [1]

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 [6].

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 utilizan 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 objetivo 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 $(x_1, x_2, x_3, ...)$ 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 de 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 de las 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 [7].

En el libro *The elements of statistical learning: data mining, inference, and prediction* [10] 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 división para lograr el mejor ajuste. Luego, una o ambas regiones se dividen en dos regiones más. El proceso continua hasta que se aplica alguna regla de detención.

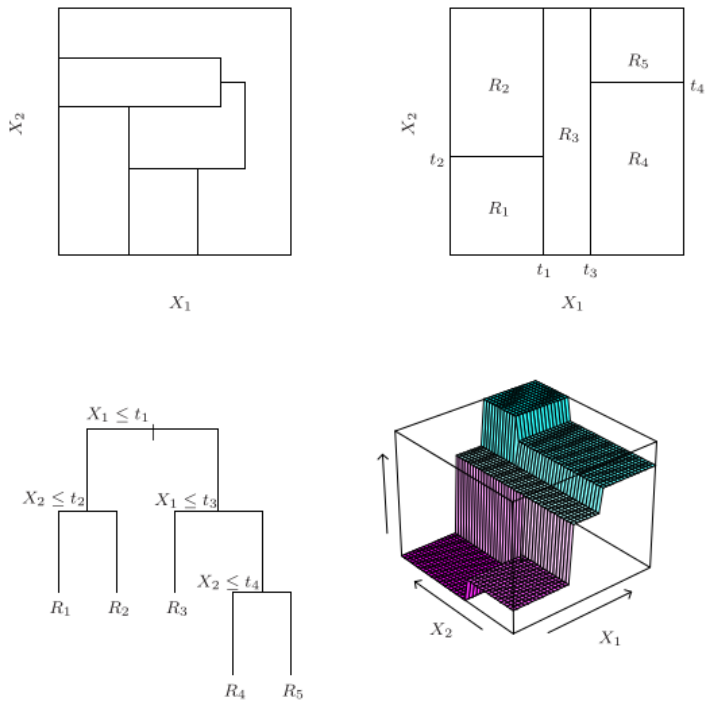

Figura 4: Funcionamiento de un árbol de decisión [7]

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 [7].

2.3.2. Bosque aleatorio

Bosque aleatorio (*random forests*) es 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 [10].

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. 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 $f(x_1, ..., x_k)$. Esta función no es exacta al estar afectada por el ruido ϵ. El objetivo es estimar la función desconocida f, es decir, separar el componente sistemático de f del ruido aleatorio ϵ, asumiendo en este caso un acercamiento lineal [6]. 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 representa la intersección (cuando el valor de x es 0).

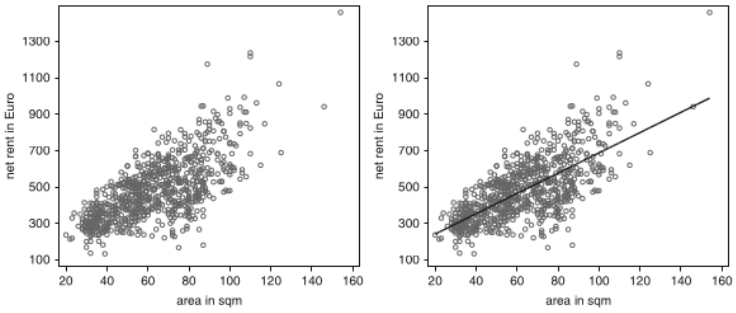

Figura 5: Regresión lineal [6]

El objetivo de la regresión lineal es encontrar la mejor línea recta que minimice 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 [10]:

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

Donde:

- $\hat{\beta}^{ridge}$ es el término de regularización *ridge*,
- y_i es la variable respuesta para la observación i,
- N es el número de observaciones,
- x_{ik} es el valor de la variable independiente k para la observación i,

- β_k son los coeficientes del modelo,
- p es el número de coeficientes de la observación.

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 [6].

3. Experimentos

En este capítulo se describe el proceso del desarrollo de la aplicación de MER. Primero, se describe el conjunto de datos utilizado en el experimento, y cómo los datos han sido manipulados para ser empleados 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* [3]. 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* [5]. 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 [2]:

- Energía.
- Intensidad de fotograma (energía de un segmento corto de la señal de audio).

- Coeficientes cepstrales Mel (MFCC) y coeficientes cepstrales Bark. Las escalas Mel y Bark son escalas perceptuales de frecuencias que aproximan a la forma en que los humanos perciben los sonidos. Se calculan los coeficientes de estas escalas para parametrizar las pistas de audio de forma que se refleje la percepción humana.
- Espectros de banda crítica (Mel/Bark/Octave y filtros de enmascaramiento triangulares).
- Espectros auditivos.
- Sonoridad (medida de cómo los humanos perciben la intensidad de los sonidos) aproximada a partir de espectros auditivos.
- Coeficientes predictivos lineales (LPC). Estos parámetros representan la señal de audio como la salida de un filtro lineal predictivo.
- Coeficientes perceptivos lineales predictivos (PLP): similares a los LPC, pero incorporando aspectos perceptuales basados en la audición humana.
- Coeficientes cepstrales predictivos lineales perceptivos (PLP-CC). Son iguales que los anteriores pero combinados con la representación cepstral.
- Pares espectrales de líneas (LSP) que representan frecuencias resonantes de la señal de audio.
- Frecuencia fundamental obtenida mediante la función de autocorrelación (ACF) y mediante sumación subarmónica SHS (suma de los armónicos de la señal).
- Probabilidad de emisión de voz a partir del pico del espectro ACF y SHS.
- Calidad de voz mediante el *Jitter* y el *Shimmer*. El *Jitter* mide la variabilidad temporal de la señal de audio, mientras que el *Shimmer* mide la variabilidad de la amplitud.
- Frecuencias de formantes y anchos de banda.
- Tasa de cruce por cero y media.

- Características espectrales como la energía de bandas arbitrarias, puntos de caída, centroide, entropía, máximos, mínimos, varianza, asimetría, curtosis (concentración pronunciada de energía en torno a ciertas frecuencias) o pendiente.
- Armonía espectral (presencia de componentes armónicos) para representar la nitidez psicoacústica.
- CROMA, para representar la distribución de energía espectral en semitonos. Y funciones CENS (CROMA suavizado) para obtener una representación suavizada y robusta de las características tonales de la señal.
- Funciones derivadas de CROMA para reconocimiento de acordes.
- 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 [3] 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 de 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. 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
- Bosque aleatorio
- Regresión lineal
- Regresión *ridge*

La taxonomía 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 de 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 [3] organiza los datos de la siguiente forma:

- Documentos CSV (uno por cada pista de audio) con los valores de las características para 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 [11] para su uso en lenguaje 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 es el documento *recap-data.csv*, el cual contiene todas las variables necesarias para construir los 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

Figura 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 de 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
```

Figura 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 [12]. Esta función permite la división de datos de forma aleatoria para que haya 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 de entrenamiento: 90,
  - Porcentaje de datos de entrenamiento: 10.
- Sistema para la predicción de activación (Figura 9):
  - Variables de entrenamiento: *X_train_ar*, *y_train_ar*
  - Variables de test: *X_test_ar*, *y_test_ar*
  - Porcentaje de datos de entrenamiento: 90,
  - Porcentaje de datos de entrenamiento: 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)
```

Figura 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)
```

Figura 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 un 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 [12] 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* correspondiente al modelo que se usa.
- *param grid*: la variable definida previamente con el espacio de datos para cada modelo.
- *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 )
```

Figura 10: Ejemplo de configuración de *GridSearchCV* para 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 [12].

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* [12]).

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

Figura 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]
}
```

Figura 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 *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'}
```

Figura 13: Hiperparámetros del árbol de decisión para la predicción de valencia

- Para activación:

```
]: 1 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'}
```

Figura 14: Hiperparámetros del 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 este 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 capítulo 4.

3.2.4. Bosque aleatorio

Para la creación de bosques aleatorios, se ha utilizado la función *RandomForestRegressor* del módulo *ensemble*, perteneciente a la librería *scikit-learn* de Python [12].

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* [12]).

Para este modelo, los valores de entrada de la variable *param_grid* de *GridSearchCV* están contenidos en la la variable *grid_hp_random_forest* (ver Figura 15).

```
grid_hp_random_forest = {
    'n_estimators': [100, 200],
    'criterion': ['squared_error', 'absolute_error'],
    'max_depth': [None, 5],
    'min_samples_split': [2, 4],
    'max_features': [None, 'sqrt', 'log2'],
    'random_state': [None, 3],
    'bootstrap': [True, False]
}
```

Figura 15: Variable *grid_hp_random_forest* para la validación cruzada de *LinearRegression*

La validación cruzada da como resultado los hiperparámetros:

- Para valencia:

```
[ ]: random_forest_model_valence.best_params_

[ ]: {'bootstrap': True,
      'criterion': 'squared_error',
      'max_depth': None,
      'max_features': None,
      'min_samples_split': 4,
      'n_estimators': 200,
      'random_state': None}
```

Figura 16: Hiperparámetros del bosque aleatorio para la predicción de valencia

- Para activación:

```
: random_forest_model_arousal.best_params_

: {'bootstrap': True,
   'criterion': 'squared_error',
   'max_depth': None,
   'max_features': None,
   'min_samples_split': 2,
   'n_estimators': 200,
   'random_state': None}
```

Figura 17: Hiperparámetros del bosque aleatorio para la predicción de 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 este 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 capítulo 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 [12].

Para el ajuste de hiperparámetros del modelo, se ha utilizado *GridSearchCV* (función presente en el módulo *model selection* de *sckit-learn* [12]) para realizar una validación cruzada exhaustiva.

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

Figura 18: Uso de las funciones *LinearRegression* y *GridSearchCV* de *sckit-learn*

Mediante la variable *grid_hp_linear_reg* mostrada en la Figura 19 se ha definido un espacio de valores de configuración 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_linear_reg = {
    'fit_intercept': [True, False],
    'copy_X': [True, False],
    'n_jobs': [None, -1],
    'positive': [True, False]
}
```

Figura 19: Variable *grid_hp_linear_reg* para la validación cruzada de *LinearRegression*

La validación cruzada da como resultado la combinación de parámetros de entrada donde la función *LinearRegression* tiene una mayor precisión. Los hiperparámetros obtenidos son:

- Para valencia:

```
linear_reg_model_valence.best_params_
{'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'positive': True}
```

Figura 20: Hiperparámetros de la regresión lineal para la predicción de valencia

- Para activación:

```
linear_reg_model_arousal.best_params_
{'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'positive': True}
```

Figura 21: 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 este 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 capítulo 4.

3.2.6. Regresión *ridge*

Para la implementación de los modelos de regresión *ridge*, se ha utilizado la función *Ridge*, perteneciente a la librería *scikit-learn* de Python [12].

Para el ajuste de hiperparámetros del modelo, se ha utilizado *GridSearchCV* (función presente en el módulo *model selection* de *sckit-learn* [12]) 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_ridge_reg* (Ver Figura 22).

```
grid_hp_ridge_reg = {
    'alpha': [1.0, 5.0],
    'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs'],
    'max_iter': [None, 1000, 15000],
    'positive': [True, False],
    'random_state': [None, 3, 5]
}
```

Figura 22: Variable *grid_hp_ridge_reg* para la validación cruzada de *Ridge*

Los hiperparámetros elegidos por la validación cruzada para la regresión *ridge* son:

- Para valencia:

```
: 1 ridge_reg_model_valence.best_params_

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

Figura 23: Hiperparámetros de la regresión *ridge* para la predicción de valencia

- Para activación:

```
In [ ]: 1 ridge_reg_model_arousal.best_params_

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

Figura 24: Hiperparámetros de la regresión *ridge* para la predicción de 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 este 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 capítulo 4.

4. Resultados

4.1. Análisis de la características

Para analizar los resultados obtenidos, primero se ha puesto el foco en las características. Mediante el uso de la función *feature_importances_* se ha calculado la importancia que han tenido cada una de las variables independientes en los modelos de árbol de decisión y bosque aleatorio. Esta función devuelve un valor entre [0, 1], y la suma de todas las importancias de las características es igual a 1. Una vez calculadas todas las importancias, se han ordenado de mayor a menor para obtener una representación visual de las características más importantes.

En los modelos de árbol de decisión (tanto para la predicción de valencia como de activación), destacan muy pocas características por encima de la mayoría que tienen valor 0. En cada modelo, hay una característica que destaca por encima del resto (Figuras 25 y 26): para valencia es *audspec_lengthL1norm_sma_de_stddev*[1], y para activación *audspec_lengthL1norm_sma_amean*[2]. Ambas características miden aspectos del espectro auditivo.

	Importancia
audspec_lengthL1norm_sma_de_stddev	0.590566
pcm_fftMag_mfcc_sma[1]_amean	0.283756
jitterLocal_sma_de_stddev	0.063279
audspec_lengthL1norm_sma_amean	0.062399
F0final_sma_stddev	0.000000
audSpec_Rfilt_sma_de[13]_amean	0.000000
audSpec_Rfilt_sma_de[13]_stddev	0.000000
audSpec_Rfilt_sma_de[12]_amean	0.000000

Figura 25: Características más importantes para árbol de decisión (predicción de valencia)

[1]Desviación estándar de los coeficientes delta de la media móvil suavizada de la norma L1 del espectro auditivo.

[2]Media aritmética de la media móvil suavizada de la norma L1 del espectro auditivo.

	Importancia
audspec_lengthL1norm_sma_amean	0.577854
pcm_fftMag_spectralEntropy_sma_de_stddev	0.147803
audspec_lengthL1norm_sma_de_stddev	0.147049
logHNR_sma_amean	0.037789
pcm_fftMag_mfcc_sma[1]_amean	0.030927
pcm_fftMag_spectralVariance_sma_amean	0.030360
shimmerLocal_sma_de_stddev	0.028218
audSpec_Rfilt_sma_de[13]_stddev	0.000000
audSpec_Rfilt_sma_de[12]_amean	0.000000

Figura 26: Características más importantes para árbol de decisión (predicción de activación)

Para los modelos de bosque aleatorio se observa un resultado similar al obtenido en los de árbol de decisión (Figuras 27 y 28): unas pocas características tienen valor distinto de 0. En este caso los resultados están más suavizados, es decir, los valores de las características más importantes son similares. Destacan sobre el resto características diferentes en cada modelo: para el modelo de valencia destacan *audspec_lengthL1norm_sma_de_stddev* y *pcm_fftMag_mfcc_sma[1]_amean*[3], y para activación *audspec_lengthL1norm_sma_amean*. Estas características miden aspectos del espectro auditivo y del espectro de frecuencia de Mel.

	Importancia
audspec_lengthL1norm_sma_de_stddev	0.192966
pcm_fftMag_mfcc_sma[1]_amean	0.182091
pcm_fftMag_spectralFlux_sma_de_stddev	0.031868
audspec_lengthL1norm_sma_amean	0.030294
pcm_fftMag_spectralFlux_sma_amean	0.019323
shimmerLocal_sma_de_stddev	0.016924
jitterLocal_sma_de_stddev	0.010951
pcm_fftMag_spectralKurtosis_sma_amean	0.009908
jitterDDP_sma_de_stddev	0.009447

Figura 27: Características más importantes para bosque aleatorio (predicción de valencia)

[3]Media aritmética de los coeficientes cepstrales en la frecuencia Mel (MFCC) suavizados mediante una media móvil, calculados a partir de la magnitud de la FFT de la señal de audio digitalizada.

	Importancia
audspec_lengthL1norm_sma_amean	0.242928
pcm_fftMag_spectralEntropy_sma_de_stddev	0.106660
audspec_lengthL1norm_sma_de_stddev	0.055677
pcm_fftMag_mfcc_sma[1]_amean	0.047834
pcm_fftMag_spectralVariance_sma_amean	0.023260
pcm_fftMag_psySharpness_sma_de_stddev	0.014910
shimmerLocal_sma_de_stddev	0.014439
pcm_fftMag_spectralFlux_sma_amean	0.013743
pcm_fftMag_mfcc_sma[10]_amean	0.008832

Figura 28: Características más importantes para bosque aleatorio (predicción de activación)

Para hacer el análisis en los modelos de regresión lineal y *ridge* se ha empleado la función *coef_* para calcular los coeficientes de las variables independientes. Estos coeficientes representan la relación entre cada variable independiente con la variable dependiente. El rango de los coeficientes pueden ser cualquier número real, por lo que el rango es (-∞, ∞):

- Un coeficiente positivo indica una relación directa: a medida que la característica aumenta, el valor de la variable dependiente también tiende a aumentar.
- Un coeficiente negativo indica una relación inversa: a medida que la característica aumenta, el valor de la variable dependiente tiende a disminuir.
- Un coeficiente de 0 sugiere que la característica no tiene ninguna relación lineal con la variable dependiente.

Para regresión lineal (tanto en valencia como en activación) hay dos variables que destacan por encima del resto (Figuras 29 y 30): *shimmerLocal_sma_de_amean*[4] y *jitterDDP_sma_de_amean*[5]. Estas características miden aspectos relacionados con la variabilidad temporal del audio (*Jitter*) y la variación de la amplitud del audio (*Shimmer*).

[4]Media aritmética de los coeficientes delta de la media móvil suavizada del shimmer local.
[5]Media aritmética de los coeficientes delta de la media móvil suavizada del jitter DDP.

	Importancia
shimmerLocal_sma_de_amean	27340.029645
jitterDDP_sma_de_amean	24311.199612
jitterDDP_sma_de_stddev	22115.540627
shimmerLocal_sma_de_stddev	7369.382856
voicingFinalUnclipped_sma_de_stddev	4621.678970
jitterLocal_sma_de_amean	1136.452039
pcm_fftMag_spectralSlope_sma_de_amean	598.417183
pcm_fftMag_spectralHarmonicity_sma_de_amean	156.402630
pcm_fftMag_spectralFlux_sma_de_amean	135.478156

Figura 29: Características más importantes para regresión lineal (predicción de valencia)

	Importancia
shimmerLocal_sma_de_amean	6200.551602
jitterLocal_sma_de_amean	1012.118046
pcm_fftMag_spectralSlope_sma_de_amean	581.074969
pcm_fftMag_fband1000-4000_sma_de_amean	141.952809
pcm_fftMag_fband250-650_sma_de_amean	134.747001
pcm_fftMag_spectralEntropy_sma_de_amean	65.647901
pcm_fftMag_spectralHarmonicity_sma_de_amean	37.686683
logHNR_sma_de_stddev	31.364247
pcm_fftMag_mfcc_sma_de[2]_amean	25.779844

Figura 30: Características más importantes para regresión lineal (predicción de activación)

En los modelos de regresión *ridge*, probablemente debido al efecto de la normalización, se aprecia un decrecimiento gradual en la importancia de las características (Figuras 31 y 32). Esto indica que todas las características influyen de forma similar en el resultado de la variable objetivo.

	Importancia
pcm_fftMag_fband250-650_sma_de_stddev	0.640207
audspec_lengthL1norm_sma_amean	0.626606
audspec_lengthL1norm_sma_stddev	0.583462
pcm_fftMag_mfcc_sma_de[4]_stddev	0.575928
pcm_fftMag_spectralFlux_sma_stddev	0.550687
audspec_lengthL1norm_sma_de_stddev	0.493948
pcm_fftMag_spectralFlux_sma_de_stddev	0.313460
pcm_fftMag_spectralHarmonicity_sma_de_stddev	0.305019
pcm_fftMag_spectralEntropy_sma_amean	0.298272

Figura 31: Características más importantes para regresión *ridge* (predicción de valencia)

	Importancia
audspec_lengthL1norm_sma_amean	0.643724
audspec_lengthL1norm_sma_de_stddev	0.533155
audspec_lengthL1norm_sma_stddev	0.505169
pcm_fftMag_mfcc_sma_de[4]_stddev	0.480647
pcm_fftMag_mfcc_sma_de[2]_stddev	0.472288
pcm_fftMag_mfcc_sma_de[1]_stddev	0.425346
pcm_fftMag_spectralFlux_sma_stddev	0.422188
pcm_fftMag_spectralSkewness_sma_de_stddev	0.380496
pcm_fftMag_spectralFlux_sma_de_stddev	0.361670

Figura 32: Características más importantes para regresión *ridge* (predicción de activación)

4.2. Comparativa de modelos

La evaluación de modelos predictivos se realiza mediante la comparación de las variables objetivo (predichas por el sistema) con los valores reales. En este trabajo, al tener dos variables objetivo, han de compararse los resultados obtenidos para la valencia y para la activación por separado.

La salida de este sistema de MER es la predicción de emociones representadas por dos magnitudes. En decir, un punto en un espacio bidimensional. En la siguiente Figura 33 se muestra una comparativa entre el valor real y el predicho de cinco canciones del conjunto de variables de test.

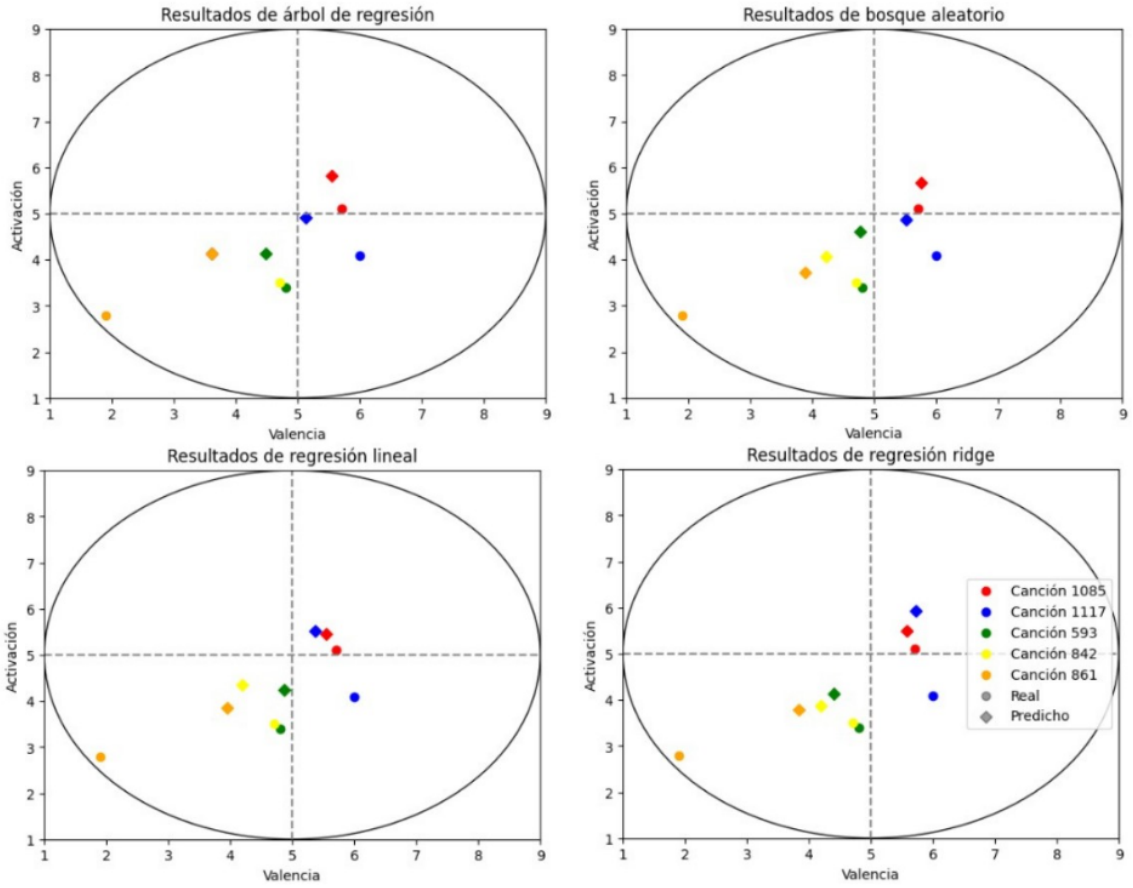

Figura 33: Comparativa de modelos

A simple vista, no es posible evaluar cómo de eficientes son los modelos empleados en el sistema. A diferencia de los problemas de clasificación (donde la precisión de los modelos se mide a partir de los fallos y aciertos del modelo), los modelos de regresión precisan de diferentes métricas para evaluar la efectividad. Los valores a predecir y los reales son continuos, y mediante las métricas de evaluación se mide lo cerca (o lejos) que están los datos reales de los predichos. Las métricas elegidas para este experimento son:

- *Mean absolute error* (MAE)
- *Mean squared error* (MSE)
- Coeficiente de determinación (R2 *Score*)
- *Mean absolute percentage error* (MAPE)

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

4.2.1. 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 del MAE es [16]:

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

donde

- N es el número de observaciones,
- y_i es el valor real de la observación,
- $\hat{y}_i$ es el valor predicho.

Los resultados del MAE obtenidos durante el experimento son:

- Para valencia:

Cuadro 1: MAE de valencia

Algoritmo	**MAE**
Árbol de decisión	0.811
Bosque aleatorio	0.680
Regresión Lineal	0.649
Regresión *ridge*	0.696

- Para activación:

Cuadro 2: MAE de activación

Algoritmo	**MAE**
Árbol de decisión	0.842
Bosque aleatorio	0.718
Regresión Lineal	0.795
Regresión *ridge*	0.852

Si se observa la fórmula (4), MAE mide la diferencia media entre el valor real y el valor predicho. Por lo tanto, el mejor modelo respecto a esta métrica será aquel con el menor MAE. En las tablas 1 y 2 se puede ver que los modelos con mejor puntuación son: regresión lineal para valencia, y bosque aleatorio para activación.

La interpretación del MAE se hace en función de los valores posibles de la variable dependiente, que para este caso, tanto valencia como activación están entre 1 y 9 (esta relación se puede ver mejor en el apartado 4.2.4 donde se analiza la métrica MAPE). Observando los resultados, los modelos con mejor puntuación MAE predicen con un error de 0.649 y 0.718 respectivamente.

4.2.2. Mean squared error (MSE)

El *mean squared error* (o error cuadrático medio) es similar al MAE, con la diferencia de que mide el promedio de los errores al cuadrado. La fórmula del MSE es [17]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5}$$

donde

- N es el número de observaciones,
- y_i es el valor real de la observación,
- $\hat{y}_i$ es el valor predictivo.

Los resultados del MSE obtenidos son:

- Para valencia:

Cuadro 3: MSE de valencia

Algoritmo	**MSE**
Árbol de decisión	0.904
Bosque aleatorio	0.688
Regresión Lineal	0.646
Regresión *ridge*	0.708

- Para activación:

Cuadro 4: MSE de activación

Algoritmo	**MSE**
Árbol de decisión	1.035
Bosque aleatorio	0.806
Regresión Lineal	0.946
Regresión *ridge*	1.094

Al igual que MAE, cuanto menor sea el valor obtenido, mejor será el modelo. Los resultado obtenidos de esta métrica son (ver tablas 3 y 4): modelo de regresión lineal para valencia con una MSE de 0.646, y modelo de bosque aleatorio para activación con un valor de 0.806. Ambos valores son menores que 1, por lo que el resultado para los modelos se puede considerar bueno.

4.2.3. Coeficiente de determinación (R2 *score*)

El coeficiente de determinación (R2 *score*) mide la varianza en la variable objetivo a partir de las variables independientes. Su fórmula es [15]:

$$R2 = 1 - \frac{\sigma_r^2}{\sigma_y^2} \tag{6}$$

donde

- σ_r^2 es la varianza de la variable residual (diferencia entre lo real y lo predicho),
- σ_y^2 es la varianza de la variable independiente.

Los valores obtenidos son los siguientes:

- Para valencia:

Cuadro 5: Coeficiente R2 para valencia

Algoritmo	**R2**
Árbol de decisión	0.303
Bosque aleatorio	0.469
Regresión Lineal	0.502
Regresión *ridge*	0.454

- Para activación:

Cuadro 6: Coeficiente R2 para activación

Algoritmo	**R2**
Árbol de decisión	0.363
Bosque aleatorio	0.504
Regresión Lineal	0.417
Regresión *ridge*	0.326

Los valores de R2 varían entre 0 y 1 (aunque hay casos donde puede ser negativo si el modelos es muy inadecuado). Si $R2 = 1$ significa que el modelo explica toda la variabilidad de la variable independiente, mientras que si $R2 = 0$ no lo hace. En los resultados (tablas 5 y 6) se observa que los modelos con mejor R2 son: regresión lineal para valencia, y bosque aleatorio para activación. Ambos valores están alrededor de 0.5, lo que quiere decir que los modelos explican el 50 % de la variabilidad de los datos. Para un caso como el que presenta este trabajo donde hay muchas variables independientes, se puede decir que el valor de R2 conseguido es generalmente bueno.

4.2.4. Mean absolute percentage error (MAPE)

El *mean absolute percentage error* (error porcentual absoluto medio) mide la precisión de la predicción como una proporción entre el valor real y el valor predicho. Su fórmula es [18]:

$$MAPE = 100\frac{1}{N}\sum_{t=1}^{N}|\frac{A_t - F_t}{A_t}| \tag{7}$$

donde

- N es el número de observaciones,
- A_t es el valor real,
- F_t es el valor predicho.

Los resultados obtenidos del MAPE son:

- Para valencia:

Cuadro 7: MAPE de valencia

Algoritmo	**MAPE**
Árbol de decisión	0.170
Bosque aleatorio	0.143
Regresión Lineal	0.137
Regresión *ridge*	0.149

- Para activación:

Cuadro 8: MAPE de activación

Algoritmo	**MAPE**
Árbol de decisión	0.181
Bosque aleatorio	0.160
Regresión Lineal	0.174
Regresión *ridge*	0.186

Esta métrica es similar a MAE (ver apartado 4.2.1), con la diferencia de que representa el error en forma de porcentaje. Los modelos con menor porcentaje de

error son los modelos más precisos. Según los resultados observados en el experimento (ver tablas 7 y 8), los modelos más precisos son: regresión lineal para valencia, con un 13.7 % de error; y bosque aleatorio para activación con 16 % de error. En general, los modelos pueden considerarse como buenos según esta métrica.

5. Conclusiones

En este trabajo se ha realizado una comparativa de algoritmos de regresión para la predicción de emociones musicales. Para ello se ha realizado un estudio previo de la disciplina de MER (marco teórico, estado actual y futuro). También se han definido los problemas de regresión de *Machine Learning* y las técnicas empleadas en el experimento. Finalmente, se han aplicado los distintos modelos y se han calculado métricas para valorar su rendimiento.

A continuación se detallan las conclusiones obtenidas de los resultados del experimento. Se analizan también las competencias adquiridas durante el estudio del Grado que fueron empleadas en el trabajo. Por último, se proponen las posibles líneas futuras que seguiría el trabajo.

5.1. Conclusiones

El primer objetivo propuesto para el trabajo es el estudio de la disciplina de MER. Este objetivo se ha cumplido: para la elaboración del capítulo de *Estado del arte* se ha llevado a cabo una investigación sobre este campo mediante la lectura de libros, artículos y trabajos.

La primera conclusión obtenida del trabajo, es la complejidad del campo de estudio de MER. Partiendo de la propia naturaleza de las emociones, no es posible construir un sistema que prediga la totalidad de las emociones que un oyente extrae de la escucha de una canción. El apartado 2.1.1 describe dos tipos de emociones (percibidas e inducidas). Mediante el análisis de las características acústicas de la canción solo se pueden estimar las emociones percibidas, al estar relacionadas directamente con el momento de la escucha de la canción. Un sistema en el que se introduzcan variables o elementos que analicen las emociones inducidas sería demasiado complejo.

Los experimentos se han desarrollado en su totalidad usando el lenguaje Python, cumpliendo así los objetivos marcados para el trabajo. El uso de los cuadernos Jupyter ha resultado de gran ayuda para la ejecución del código. Al igual que las librerías como *scikit-learn* o *Pandas*, muy útiles para el desarrollo de algoritmos y el manejo de datos.

Desde el punto de vista del empleo de técnicas de *Machine Learning* se concluye que, los modelos óptimos no son los mismos para predecir valencia que para predecir activación. Los resultado son claros: todas las métricas utilizadas para la comparativa han dado los mismo resultados. En el caso de valencia, el algoritmo óptimo es la regresión lineal. Mientras que para activación es el bosque aleatorio. Al ser un sistema compuesto por dos variables objetivo diferente, se puede usar un algoritmo diferente para cada una.

5.2. Competencias y conocimientos

En el desarrollo del trabajo, se han empleado diferentes técnicas y conocimientos adquiridos durante el estudio del Grado. Algunas de ellas son:

- Uso del lenguaje Python de programación. Conocimiento adquirido a través del estudio de varias asignaturas como *Protocolos para la Transmisión de Audio y Video en Internet* o *Tratamiento Digital del Sonido*. Más específicamente, el uso del entorno de Anaconda y los cuadernos Jupyter. El uso de este entorno para desarrollos de *Machine Learning* es bastante habitual, motivo de su elección para el desarrollo del experimento.
- De la asignatura de *Tratamiento Digital del Sonido* también se han adquirido los conocimientos necesarios para el uso y el manejo de las características acústicas de las canciones.
- Durante el experimento, se emplean técnicas de *Machine Learning*. En la asignatura de *Tratamiento Digital del Sonido* se aprendieron los conocimientos

básicos de este campo, y algunas de las técnicas empleadas durante el trabajo.

- De la asignatura *Estadística para Medios Audiovisuales*, se adquirieron los conocimientos para comprender los conceptos estadísticos de las métricas utilizadas en la evaluación de los modelos.

Además de las competencias adquiridas durante el estudio del Grado, otras nuevas se han aprendido en el desarrollo del trabajo. Como son:

- Empleo del lenguaje Latex para la redacción de la memoria. Su uso ha resultado de gran ayuda para estructurar el texto, o incluir en la composición de la memoria diferentes elementos como referencias, imágenes y tablas.
- Aumentar el conocimiento en *Machine Learning* mediante técnicas y algoritmos no estudiados en el Grado.
- Conocimiento adquirido durante la investigación en el campo de estudio de MER.

5.3. Futuras líneas de trabajo

Una vez realizados los experimentos propuestos para el trabajo se detectan las siguientes líneas de estudio futuras para continuar mejorando y expandiendo esta aplicación MER:

- Implementación de otros algoritmos y técnicas de *Machine Learning* con el objetivo de mejorar las puntuaciones de precisión en la predicción obtenidas en el experimento.
- Hubiera sido interesante contar con una base de datos donde, el espacio bidimensional de Russell, estuviera segmentado en distintas emociones. Como se comentaba en el apartado 2.1.3.3, una combinación discreta-dimensional

de la taxonomía de las emociones puede ayudar en la efectividad del modelo. De esta forma, la aplicación sería mucho más intuitiva puesto que trabajaría directamente con emociones discretas en lugar de su representación mediante valencia-activación. Por ejemplo, si una canción está identificada con la emoción "alegre", se puede probar a simple vista si la aplicación predijo "alegre" de forma satisfactoria u otra emoción.

- La letra de las canciones juega un papel importante en las emociones percibidas por el oyente. Una posible línea de trabajo es la introducción de la letra como una variable independiente más del modelo. Al estudiar solo las características acústicas de la canción, se pierden los posibles matices que pueda tener la letra. Una canción puede tener una letra triste pero sin embargo una música alegre, entonces la emoción percibida se puede ver "contaminada" en gran manera por la letra.

- El complemento más obvio a este trabajo es la integración de este desarrollo en una aplicación final (de interfaz de usuario amigable). Un caso claro sería un reproductor streaming de música, que pueda recomendar o clasificar canciones en función de sus emociones. O un editor de video que, como complemento a la imagen, quiera incluir música específica para evocar en el usuario una emoción precisa.

6. Bibliografía

[1] URL: https://scikit-learn.org/stable/modules/cross_validation.html.

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

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

[4] Tuomas Eerola y Jonna K Vuoskoski. "A review of music and emotion studies: Approaches, emotion models, and stimuli". En: *Music Perception: An Interdisciplinary Journal* 30.3 (2012), págs. 307-340.

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

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

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

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

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

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

[11] Wes McKinney. "Data Structures for Statistical Computing in Python". En: *Proceedings of the 9th Python in Science Conference*. Ed. por Stéfan van der Walt y Jarrod Millman. 2010, págs. 56-61. DOI: 10.25080/Majora-92bf1922-00a.

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

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

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

[15] Wikipedia. *Coeficiente de determinación — Wikipedia, La enciclopedia libre*. [Internet; descargado 22-febrero-2024]. 2024. URL: https://es.wikipedia.org/w/index.php?title=Coeficiente_de_determinaci%C3%B3n&oldid=158364923.

[16] 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.

[17] Wikipedia. *Error cuadrático medio — Wikipedia, La enciclopedia libre*. [Internet; descargado 6-mayo-2024]. 2024. URL: https://es.wikipedia.org/w/index.php?title=Error_cuadr%C3%A1tico_medio&oldid=159951880.

[18] Wikipedia contributors. *Mean absolute percentage error — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-June-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Mean_absolute_percentage_error&oldid=1225070385.

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