EmocIArte:

Predicción de emociones a partir de paisajes sonoros mediante aprendizaje automático

ALUMNO: DANIEL CREMY GRAU

TUTOR: JESÚS ÁLVAREZ HERRERA (IES GERARDO DIEGO)
COTUTORES: INMACULADA MORA Y ROBERTO SAN MILLAN (URJC)

IES GERARDO DIEGO, CURSO 2023-25

Agradecimientos

Quiero expresar mi más sincero agradecimiento a Jesús por su apoyo constante a lo largo de este proceso. A Inmaculada y Roberto les debo un reconocimiento especial por haberme sugerido el tema de este trabajo y por su ayuda inestimable en comprender el funcionamiento del aprendizaje automático, así como en adentrarme en el campo de la inteligencia artificial.

Asimismo, deseo reconocer a mis profesores por su dedicación y esfuerzo. Su empatía y comprensión, especialmente en las últimas semanas, han sido fundamentales para mantenernos motivados y comprometidos en la recta final de este proyecto. También agradezco a mis compañeros, cuya colaboración y compañerismo han hecho que este desafío sea más llevadero.

Finalmente, un agradecimiento especial a mi familia por su apoyo constante y por los ánimos que me brindaron ante las dificultades de este reto. Su respaldo ha sido fundamental a lo largo de este camino.

Índice

Índice de figuras

Índice de tablas

Resumen

El estudio de los paisajes sonoros, que analiza la relación entre las personas y los entornos acústicos, ha cobrado importancia en los últimos años, especialmente en el campo de la acústica ambiental. Los paisajes sonoros no solo afectan el bienestar físico, sino que también influyen en las emociones de los individuos. Este trabajo se enfoca en la **predicción de emociones percibidas** en estos entornos empleando **técnicas de aprendizaje automático**, una de las patas fundamentales de la **Inteligencia Artificial (IA**). Para ello, las respuestas emocionales se caracterizarán en términos de activación y valencia (AV), permitiendo una representación bidimensional comprensible que facilita el análisis de los paisajes sonoros.

El objetivo principal es identificar las **características acústicas** más relevantes para predecir las emociones en paisajes sonoros, utilizando el modelo de **k vecinos más próximos (k-NN)**. En este proceso, se analizan distintas configuraciones del modelo, como el uso de salidas simultáneas o separadas para la predicción de activación y valencia, y la **selección de parámetros libres** que maximicen las prestaciones del modelo. También se realiza una evaluación de la **calidad del modelo,** comparando los resultados predichos con datos reales para asegurar su efectividad en diferentes contextos acústicos.

Una parte del trabajo explora de manera preliminar la **automatización en la extracción de características acústicas,** con el fin de mejorar la eficiencia en la modelización, aunque este aspecto es secundario en relación con los principales objetivos del estudio.

En resumen, este trabajo ofrece una aproximación innovadora a la **modelización del reconocimiento de emociones en paisajes sonoros,** utilizando herramientas aprendizaje automático. Sus resultados buscan ser una contribución al estudio de la planificación urbana y la gestión acústica, facilitando el análisis de entornos sonoros que influyan positivamente en el bienestar emocional.

Abstract

The study of soundscapes, which analyzes the relationship between people and acoustic environments, has gained importance in recent years, particularly in the field of environmental acoustics. Soundscapes not only affect physical well-being but also influence individuals' emotions. This work focuses on **predicting perceived emotions** in these environments using **machine learning techniques**, one of the key pillars of **Artificial Intelligence (AI)**. Emotional responses are characterized in terms of arousal and valence (AV), allowing for a comprehensible two-dimensional representation that facilitates the analysis of soundscapes.

The main objective is to identify the most relevant **acoustic features** for predicting emotions in soundscapes, using the **k-nearest** neighbors (k-NN) model. In this process, different configurations of the model are analyzed, such as the use of simultaneous or separate outputs

for predicting arousal and valence, as well as the selection of **free parameters** that maximize the **model's performance**. A model quality evaluation is also carried out by comparing predicted results with real data to ensure its effectiveness in various acoustic contexts.

A portion of the work preliminarily explores **the automation of acoustic feature extraction**, aiming to improve efficiency in modeling, although this aspect is secondary to the primary goals of the study.

In summary, this work offers an innovative approach to e**motion recognition modeling in soundscapes** using machine learning tools. Its results aim to contribute to the study of urban planning and acoustic management, facilitating the analysis of sound environments that positively influence emotional well-being.

1. Introducción

El ser humano vive rodeado de sonidos. El entorno sonoro que nos rodea es una parte importante del día a día y tiene influencia sobre el bienestar físico y emocional. En este contexto, el ruido se ha convertido en un fenómeno que afecta a millones de personas. Tradicionalmente, la preocupación más importante se ha centrado en la monitorización y el control del nivel sonoro, sin embargo, en los entornos de las grandes ciudades, el ruido es algo que nos acompaña y está presente en todo momento.

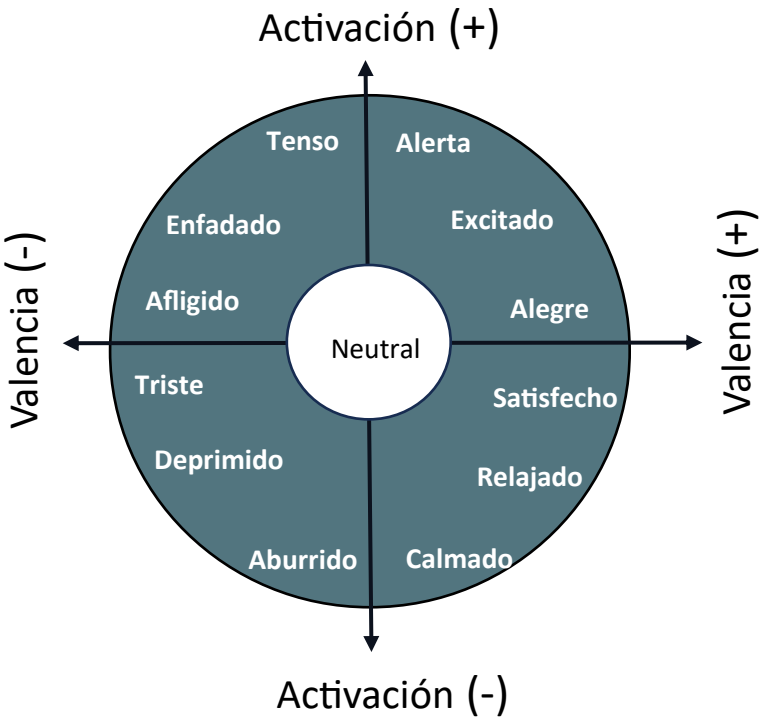
Esto ha sido el origen del concepto de paisaje sonoro, que va más allá del ruido, y considera también la percepción de los individuos en diferentes contextos. La palabra "paisaje sonoro" fue popularizada por Schafer (Schafer, 1977), quien definió el término como "un entorno de sonido (o entorno sonoro) con énfasis en la forma en que es percibido y comprendido por el individuo, o por una sociedad". Schafer exploró cómo los factores humanos y el entorno acústico se entrelazan favoreciendo lo que se ha llamado ecología acústica, que reúne diversas disciplinas, tales como la acústica, la psicoacústica, la percepción humana, la ingeniería de sonido, etc. Otra definición de paisaje sonoro es la de Payne et al. (2009) que definen los paisajes sonoros como " la totalidad de todos los sonidos de un lugar, con especial atención a la relación entre la percepción, la comprensión y la interacción del individuo o la sociedad con el entorno sonoro". Mas recientemente la Organización Internacional de Normalización (ISO) definió el término como "el entorno percibido o experimentado y/o comprendido por una persona o personas, en un contexto" (ISO, 2014). Todas estas definiciones tienen algo en común y es que hacen referencia a los sonidos en un entorno y a la forma en que son percibidos por un oyente en un tiempo y espacio determinado.

En lugar de centrarse únicamente en la reducción del ruido, el análisis de los paisajes sonoros busca entender cómo los sonidos impactan nuestras emociones, moldean nuestras experiencias y contribuyen a la identidad de lugares específicos. Sin embargo, esta investigación enfrenta importantes desafíos metodológicos, tales como la necesidad de recopilar datos en diversas ubicaciones y contar con la participación de un gran número de personas para evaluar las emociones asociadas a los paisajes sonoros.

A pesar de estas dificultades, la modelización de la percepción de los paisajes sonoros es crucial, ya que permite predecir las respuestas humanas a diferentes circunstancias acústicas de manera eficiente. Esto facilita que los procesos de diseño y planificación urbana se

optimicen sin depender exclusivamente de experimentos complejos. Así, la modelización hace que análisis de paisajes sonoros se convierta en una herramienta práctica.

Los estudios sobre el reconocimiento de emociones en paisajes sonoros (SER, por sus siglas en inglés) no solo tienen aplicaciones en la planificación urbana y la acústica ambiental, sino que también son valiosos para el diseño sonoro en medios audiovisuales, como películas y videojuegos (Ekman, 2008). Estos estudios se centran en comprender las emociones que un oyente percibe en un paisaje sonoro específico, ampliando el ámbito de aplicación de la investigación sobre entornos acústicos.



Fuente: Adaptado de Russell (1980)

Figura 1: El modelo circumplexo de la emoción con dos ejes: valencia y activación

Una de las aproximaciones para el análisis SER es el uso de modelos dimensionales. Los modelos dimensionales utilizan un espacio cartesiano con dimensiones continuas para representar las emociones (Russell, 1980) mediante el denominado el modelo circumplexo de la emoción, que representa las emociones con dos dimensiones: valencia (*valence*) y activación (*arousal*). La valencia representa el placer o disgusto afectivo de un estímulo, mientras que la activación indica la energía y la activación provocados por un estímulo. Idealmente, todas las emociones imaginables podrían ser representadas en distintas posiciones dentro de un espacio bidimensional

Es importante señalar que diversos estudios han propuesto modelos dimensionales que incluyen dimensiones adicionales como tensión, potencia y dominancia (Bakker et al., 2014). No obstante, dado que la clasificación de los sonidos es realizada por individuos, estos encuentran difícil evaluar con precisión esas dimensiones adicionales, lo que complica el proceso de calificación. Por esta razón, aunque el espacio valencia-activación presenta limitaciones para distinguir entre emociones como el miedo intenso y la ira, la mayoría de los investigadores optan por utilizar valencia y activación (*arousal*) como las principales dimensiones para la modelización del reconocimiento de emociones en paisajes sonoros (SER).

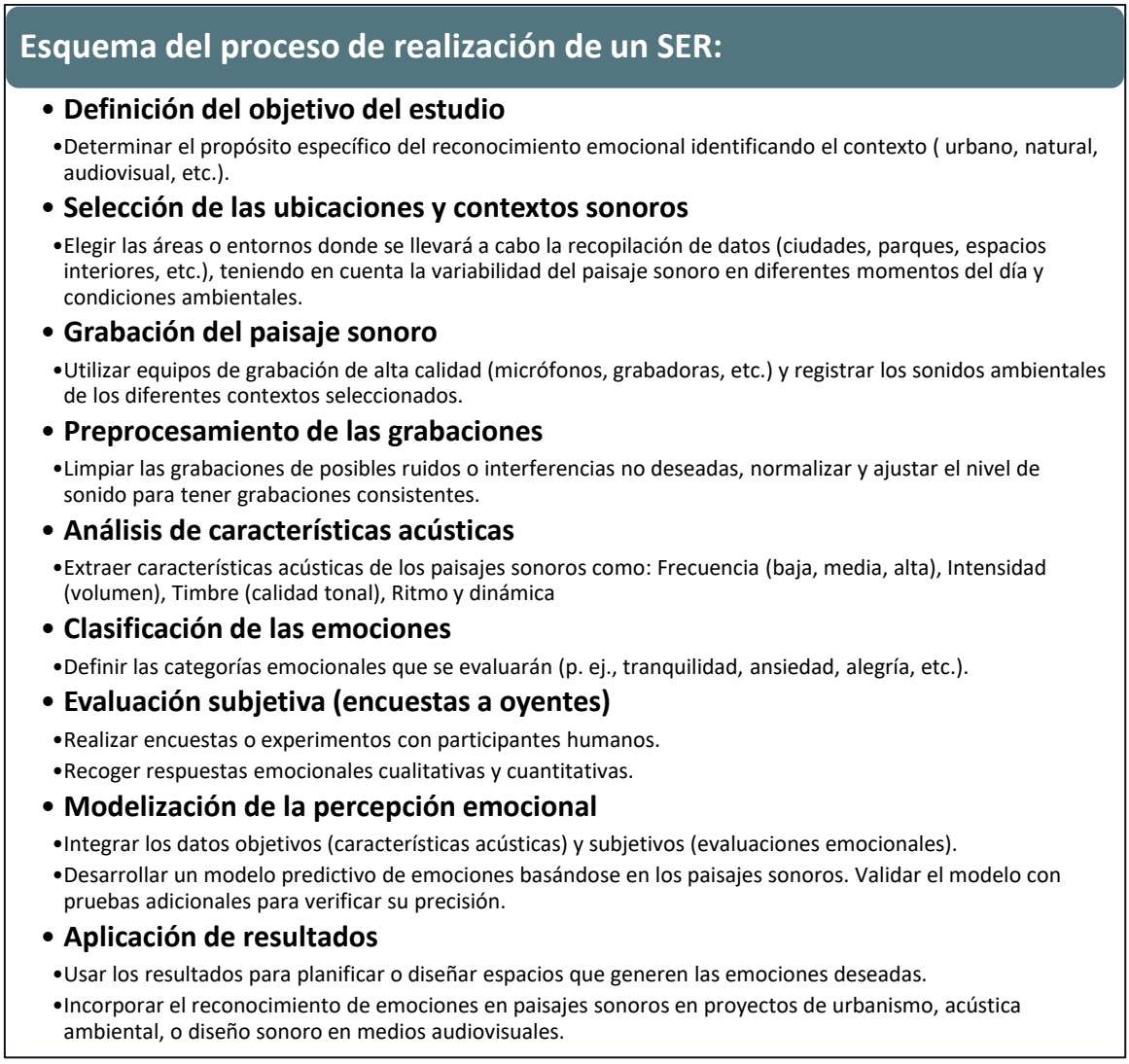
Conviene también añadir que las emociones percibidas a partir del paisaje sonoro son sustancialmente diferentes de las relacionadas con la música o el habla, porque son más sutiles y pasan más desapercibidas, por lo que su tratamiento y modelización son diferentes.

Tal y como indican Wold, et al. (1996) la modelización del paisaje sonoro puede servir para predecir la percepción de los entornos acústicos a menor coste. Así, en el ámbito del urbanismo y la acústica ambiental, este proceso se puede resumir en los siguientes pasos:

i. la grabación de paisajes sonoros,
ii. el cálculo de indicadores acústicos y psicoacústicos de las señales,
iii. recogida de otros indicadores de contexto (por ejemplo, información visual),
iv. clasificación de las señales de audio de los paisajes sonoros empleando descriptores emocionales.

A partir de este punto se puede desarrollar un modelo que facilite la evaluación y planificación de los entornos sonoros

En general, la SER relaciona las características de audio extraídas de las grabaciones de paisajes sonoros con las anotaciones de emociones percibidas. Así, el proceso completo típico de elaboración de un SER pasaría por las siguientes fases:



Fuente: Elaboración propia

Figura 2: Ilustración las fases del proceso de estudio sobre el reconocimiento de emociones del paisaje sonoro (SER)

2. Estado de la cuestión

El reconocimiento automático de emociones en paisajes sonoros (SER) se centra en identificar las emociones captadas en un entorno acústico. Aunque el progreso en las técnicas de Aprendizaje Automático (*Machine Learning*) para el procesamiento de audio ha estado en gran medida enfocado en la detección de fuentes sonoras (Barkana y Saricicek, 2010), la modelización de la percepción emocional en paisajes sonoros aún no ha sido explorada exhaustivamente.

Para poder modelizar y evaluar con precisión las emociones percibidas en un paisaje sonoro, es crucial disponer de una cantidad significativa de datos reales que reflejen de manera precisa y consistente las características acústicas del entorno en forma de grabaciones de audio etiquetadas. Una opción prometedora en este sentido es el conjunto de datos conocido como Emo-Soundscapes (Fan et a., 2017), que contiene 1213 clips de audio bajo licencia Creative Commons. El uso de este conjunto de datos puede proporcionar la base necesaria para avanzar en el desarrollo de modelos de SER y comprender mejor cómo las personas perciben y responden emocionalmente a su entorno acústico.

Sin embargo, existen otros estudios que utilizan otras técnicas para evaluar sonidos y emociones. Por ejemplo, Berglund et al. (2007) describe la encuesta para asegurarse de atributos críticos de los sonidos que percibe un ser humano al oír un paisaje sonoro dividido en: tecnológico, natural o humano. En esta encuesta, se les preguntó a 100 oyentes que evaluaran 30 grabaciones de 30 segundos y que colocasen las grabaciones escalas de calificación para 116 atributos perceptivos y emocionales. Mediante el análisis por Componentes principales encuentra que los componentes principales eran el agrado y los acontecimientos, explicando el 50% y el 18% del total de diferencia respectivamente.

Por otro lado, Davies (2014) diseñó una encuesta para evaluar los paisajes sonoros urbanos. La encuesta demostró que los oyentes que califican según escalas lineales de agradable, desagradable, enérgico aburrido, tranquilo agitado, conformado preocupado e informado confundido podrían evaluar con precisión la calidad de los paisajes sonoros urbanos.

Lundén y Hurtig (2016) exploraron un enfoque alternativo para predecir la evaluación de paisajes sonoros utilizando características acústicas. Implementaron un modelo de mezcla gaussiana para agrupar dichas características y, a partir de la matriz de disimilitud generada, entrenaron dos modelos de regresión utilizando máquinas de vectores de soporte. Estos modelos se emplearon para predecir, de manera independiente, el nivel de agrado y la percepción de dinamismo en los paisajes sonoros. Este enfoque permitió un análisis más detallado de las dimensiones clave en la evaluación de los entornos acústicos.

En términos de emociones asociadas a los paisajes sonoros, estudios previos (Kallinen y Ravaja, 2006) identificaron dos tipos principales:

- Emociones percibidas: Emociones que son comunicadas por fuente misma

- Emociones Inducidas: Reacciones emotivas que la fuente provoca al oyente

Esta taxonomía de las emociones asociadas proporciona un marco para comprender cómo los paisajes sonoros pueden influir en nuestras emociones y percepciones.
La emoción percibida es más abstracta y objetiva. Por ejemplo, la emoción percibida de las películas de terror siempre es "aterradora". Sin embargo, la emoción inducida en el espectador puede variar dependiendo de su susceptibilidad al miedo y otros factores individuales. En nuestro análisis, nos vamos a centrar en la emoción percibida del paisaje sonoro.

Categorías	Ejemplos
Sonidos naturales	Pájaros, truenos, lluvia, viento
Sonidos humanos	Reír, susurrar, gritar
Sonidos y sociedad	Fiesta, concierto, tienda
Sonidos mecánicos	Motor, fábrica
Tranquilidad y silencio	Parque tranquilo, bosque silencioso
Sonidos como indicadores	Reloj, campana de iglesia

Fuente: Fan et al. (2017)

Tabla 1: Taxonomía de Schafer

En relación con la taxonomía del paisaje sonoro, la más utilizada es la propuesta por Schafer (1997) que agrupó los paisajes sonoros en función de la identificación de la fuente del sonido y el contexto auditivo en lugar de características sonoras. Por ejemplo, en la taxonomía de Schafer, un bosque tranquilo se clasifica bajo quietud y silencio, pero sin el contexto auditivo, puede ser clasificado como sonido natural. Estas categorías no son mutuamente excluyentes. La taxonomía de Schafer se muestra en la Tabla 1.

3. Objetivos

Los objetivos del presente trabajo son los siguientes:

- **Investigar la predicción de las emociones percibidas a partir de paisajes sonoros** utilizando técnicas de aprendizaje automático que permitan caracterizar cualquier sonido dentro del paisaje sonoro mediante las dimensiones de activación (*arousal*) y valencia (espacio AV).
- **Identificar las características más importantes de un sonido del paisaje sonoro** para modelizar el reconocimiento de emociones en función de la activación (*arousal*) y valencia.
- **Estudiar dentro del modelo de aprendizaje automático elegido (*k* vecinos más próximos)**:
    - Cómo realizar un diseño adecuado del esquema predictor, si utilizando dos salidas simultáneas de un mismo modelo, o si diseñando un modelo para cada variable dependiente.
    - Como elegir el mejor parámetro libre del modelo.
    - Cómo caracterizar la variabilidad del modelo utilizando estadísticos sobre la figura de mérito.
- **Evaluar la calidad del modelo** comparando las predicciones con los datos reales, a fin de asegurar una adecuada capacidad de generalización del modelo en la predicción de emociones.
- **Investigar la posibilidad de extracción automática de características acústicas** como un paso adicional, evaluando la viabilidad de automatizar este proceso para mejorar la eficiencia del modelo en su capacidad de identificar las emociones percibidas en los paisajes sonoros.

4. Metodología

Las metodologías más populares en el campo de la ciencia de datos provienen del ámbito del Aprendizaje Automático (*Machine Learning*). Algunos ejemplos destacados de éxito en el *machine learning* incluyen la tecnología de reconocimiento de voz como Siri de Apple, sistemas de recomendación de películas, detectores de *spam* y *malware*, automóviles autónomos y predictores de precio, entre otros. Aunque en la actualidad los términos Inteligencia Artificial y Aprendizaje Automático se utilizan de manera intercambiable, es importante hacer la siguiente distinción: mientras que los primeros algoritmos de inteligencia artificial, como aquellos utilizados por las máquinas de ajedrez, se basaban en la toma de

decisiones según reglas programables derivadas de la teoría, en *machine learning* el modelo que ayuda en la toma de decisiones se construye a partir de la experiencia, basándose en los datos disponibles.

Centrándonos el aprendizaje automático, los datos relevantes serían:

1. El resultado (*outcome* en inglés) que queremos predecir.
2. Las características (*features* en inglés) que utilizaremos para predecir el resultado.

El objetivo es construir un modelo dirigido por datos que tome los valores de las características del registro sonoro como entrada y devuelva una predicción. El enfoque de aprendizaje automático implica entrenar un modelo utilizando un conjunto de datos para el cual conocemos el resultado, y luego emplear este modelo en el futuro para hacer predicciones cuando el resultado es desconocido.

Si expresamos esto de forma matemática, llamaremos Y al resultado y $X_1,..,X_p$ a las características asociadas a un determinado registro sonoro. Es decir, tenemos una serie de características y un resultado desconocido que es lo que queremos predecir (véase la

Figura 3)

Característica 1	Característica 2	Característica3	…	Característica p	Resultado
X_1	X_2	X_3	...	X_p	¿?

Figura 3: Esquema del problema de Aprendizaje Automático

Para construir un modelo que proporcione una predicción para cualquier conjunto de valores observados $X_1 = x_1$, $X_2 = x_2$, …, $X_p = x_p$ recopilamos datos para los cuales conocemos el resultado (Figura 4).

Característica 1	Característica 2	Característica3	…	Característica p	Resultado
$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	…	$x_{1,p}$	y_1
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	…	$x_{2,p}$	y_2
⋮	⋮	⋮		⋮	⋮
$x_{n,1}$	$x_{n,2}$	$x_{n,3}$	…	$x_{n,p}$	y_n

Figura 4:Esquema del funcionamiento de Aprendizaje Automático mediante la utilización de características y resultados

Cuando el resultado es continuo, nos referimos a la tarea a abordar utilizando herramientas de aprendizaje automático como una tarea de regresión. El resultado principal del modelo es una función f que produce automáticamente una predicción, denotada como $\hat{y}$ para cualquier conjunto de predictores:

$$\hat{y} = f(x_1, x_2, \dots, x_p)$$

Utilizamos el término resultado real (*actual outcome* en inglés) para denotar lo que acabamos de observar. En este punto, lo mejor sería que la predicción $\hat{y}$ coincida lo más posible con el resultado real y. Debido a que en una tarea de regresión el resultado es una magnitud continua, las predicciones $\hat{y}$ no serán exactamente correctas o incorrectas, sino que determinaremos un error definido como la diferencia entre el valor real y el predicho ($y - \hat{y}$).

Adicionalmente, la evaluación de un modelo diseñado con herramientas de *machine learning* se basa en su utilidad en el mundo real con conjuntos de datos completamente nuevos. Sin embargo, como ya se ha explicado, durante el desarrollo de un modelo generalmente contamos con un conjunto de datos limitado para el cual se conoce el resultado deseado. Para simular el proceso de evaluación final, en general se dividen los datos en dos partes y se actúa como si no conociéramos el resultado de una de ellas. El conjunto de datos para el cual conocemos el resultado y que utilizamos para diseñar el modelo se denomina el conjunto de entrenamiento (*training set* en inglés), mientras que al conjunto de datos para el cual fingimos no conocer el resultado y que se utilizará para evaluar la capacidad de generalización del modelo se le denomina conjunto de evaluación (*test set* en inglés). El concepto de generalización es muy importante en aprendizaje automático, ya que determina

la capacidad de ofrecer un resultado razonable ante ejemplos (registros sonoros, en este caso), que no se han utilizado para diseñar el modelo.

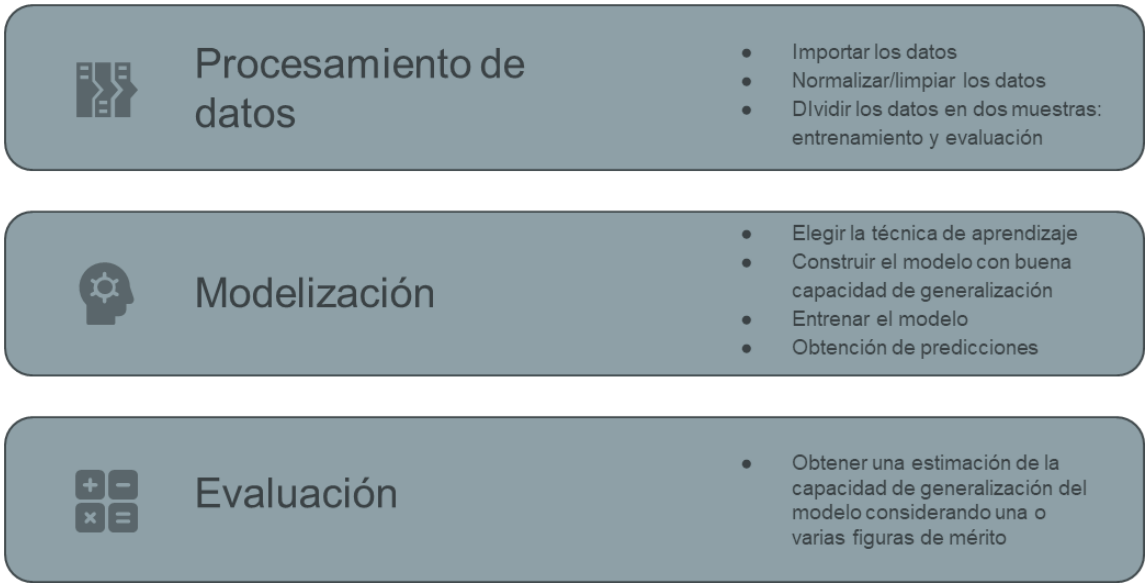


Figura 5: Esquema resumen de los pasos a seguir en el proceso de aprendizaje automático

A partir de este punto se diseña un modelo usando solo el conjunto de entrenamiento. Una vez se dispone del modelo, lo pondremos a prueba utilizando el conjunto de evaluación o validación. Una de las formas más utilizadas para calcular las prestaciones del modelo es utilizar una función de pérdida (*loss function* en inglés) que se obtendrá en función de la diferencia $y - \hat{y}$. El objetivo es obtener el diseño del modelo que minimiza la función de pérdida, razón por la cual normalmente se utiliza un valor positivo como el que se obtiene al elevarlo al cuadrado $(y - \hat{y})^2$. Debido a que frecuentemente tenemos un conjunto de evaluaciones con muchas observaciones, digamos N, usamos el error cuadrático medio (*mean squared error o MSE* por sus siglas en inglés), definido como $MSE = \frac{1}{N}(\sum(y - \hat{y})^2)$.

También se puede utilizar el Error Absoluto Medio (MAE, por sus siglas en inglés) que se calcula de la siguiente forma: $MAE = \frac{1}{N}(\sum|y - y\hat{}|)$. Así el MSE te dice cuánto se desvían en promedio los valores predichos de los valores reales, por lo que conviene valores bajos de MSE. El MAE (Mean Absolute Error) indica cómo de grande es el error promedio en las

predicciones, sin considerar su signo, por lo que un MAE más bajo también indica mejores prestaciones. Además, la Raíz del Error Cuadrático Medio (RMSE), que se calcula como la raíz cuadrada del MSE, proporciona una medida de error en las mismas unidades que la variable de interés. El RMSE también penaliza más los errores grandes, lo que lo convierte en una métrica útil para evaluar modelos en los que los errores significativos son especialmente problemáticos. Al igual que el MSE y el MAE, un RMSE más bajo sugiere un mejor rendimiento del modelo.

Uno de los métodos más simples y efectivos en el ámbito del *machine learning* es el método de los *k*-vecinos más cercanos o k-NN (*k-nearest neighbors en inglés*). El método k-NN se puede utilizar para abordar una tarea de clasificación o de regresión, siendo en ambos casos esquemas supervisados para cuyo diseño se requiere disponer de la salida deseada para los ejemplos del conjunto de entrenamiento. La idea fundamental detrás de k-NN es que ejemplos similares (próximos, en términos del vector que define las características de los ejemplos) tienden a tener respuestas (salida del modelo) similares.

En nuestro caso, dado que los datos de salida o a modelizar son continuos deberemos utilizar un algoritmo de regresión. No obstante, para una mejor comprensión de la técnica, procedemos a explicar cómo se utiliza el k-NN para el caso de clasificación.

Vamos a explicar cómo funciona el algoritmo k vecinos más próximos (k-NN) a través de un ejemplo relacionado con un problema de clasificación. Este esquema se utiliza para predecir la clase a la que pertenece un nuevo ejemplo basándose en las clases de sus vecinos más cercanos en el espacio de características.

Imaginemos que tenemos un conjunto de datos que incluye dos características, X_1 y X_2. Estas características son propiedades que describen cada observación o ejemplo y están relacionadas con una variable que ya hemos clasificado en dos categorías: A o B. Por ejemplo, supongamos que queremos clasificar individuos en dos grupos: mujeres (A) y hombres (B). En este caso, X_1 podría ser la altura y X_2 el peso. En otro escenario diferente podríamos desear clasificar manzanas (A) y naranjas (B). Aquí, X_1 podría representar el diámetro de la fruta y X_2 su rugosidad. Incluso, imaginemos que estamos clasificando imágenes de lesiones cutáneas como malignas (A) o benignas (B), en cuyo caso X_1 podría ser el tamaño de la lesión y X_2 podría representar su forma.

El esquema k-NN clasifica una nueva observación (por ejemplo, una nueva persona, fruta o imagen) basándose en la cercanía de este ejemplo a otros en el conjunto de entrenamiento ya clasificado.

Las observaciones que vamos a utilizar en el ejemplo aparecen representadas en la Figura 6 en la que se puede observar que las observaciones de la clase A suelen situarse en la parte superior derecha del gráfico, mientras que las de clase B se sitúan en la esquina inferior.

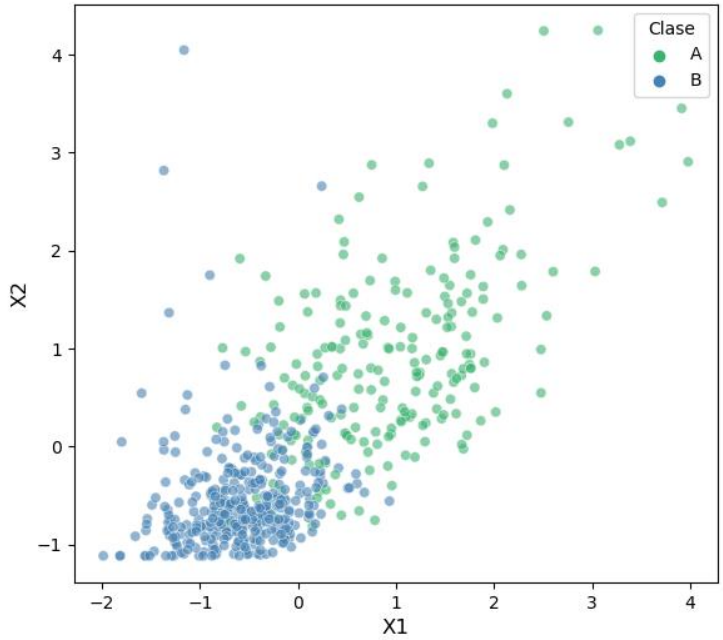


Figura 6: Diagrama de dispersión de X_1 frente X_2 coloreado por la etiqueta de clasificación

Supongamos ahora que obtenemos una nueva observación que no está en el conjunto de ejemplos actual y de la cual conocemos las dos características, pero no está clasificada. ¿Podríamos clasificar esta nueva observación? Imaginemos que el nuevo valor es como el que aparece marcado en rojo en la Figura 7. En este caso, como está en la zona de puntos verdes, donde la mayoría de ejemplos son de clase A, mediante la visualización del gráfico parece que sería apropiado afirmar que dicho punto sería de la clase A.

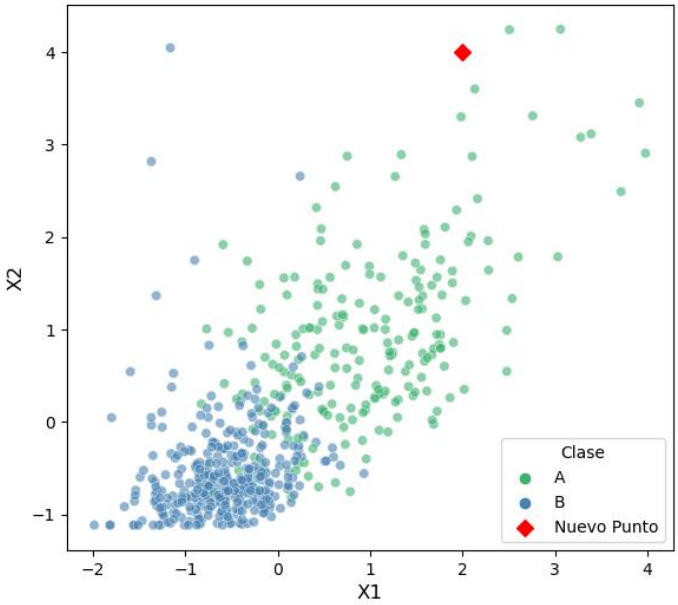


Figura 7: Diagrama de dispersión de X_1 frente X_2 coloreado por la etiqueta de clasificación con una nueva observación representada por el punto rojo

Supongamos ahora que la nueva observación es tal y como aparece en la Figura 8 ¿Cómo clasificaríamos esta nueva observación? Para clasificar esta nueva observación, comenzamos analizando su vecino más cercano. A primera vista, parece razonable clasificarla como B, ya que este vecino pertenece a la clase B. Sin embargo, al observar el conjunto de puntos cercanos, nos damos cuenta de que esta no es necesariamente la mejor clasificación.

Si ampliamos nuestro análisis y consideramos más puntos en la vecindad, notamos que la nueva observación tiene vecinos más cercanos que pertenecen a la clase A. Por ejemplo, si elegimos los cuatro vecinos más cercanos (Figura 9), observamos que la mayoría de ellos están clasificados como A. Esto sugiere que una clasificación como A podría ser más adecuada, ya que refleja mejor la agrupación de observaciones en el espacio de características.

Este ejemplo pone de manifiesto la importancia de la elección del número de vecinos cercanos en el esquema k-NN. Si seleccionamos un valor de k muy bajo, como 1, corremos el riesgo de que la clasificación esté influida únicamente por un solo vecino, lo que puede llevar a decisiones erróneas. Por otro lado, elegir un k demasiado alto podría suavizar las diferencias entre las clases y llevar a clasificaciones menos plausibles.

Por lo tanto, la selección de k es crucial para obtener una buena clasificación. Un enfoque común es explorar diferentes valores de k y evaluar las prestaciones de cada uno de los modelos sobre un conjunto de validación (no utilizado en el diseño), para determinar cuál proporciona la mejor capacidad de generalización.

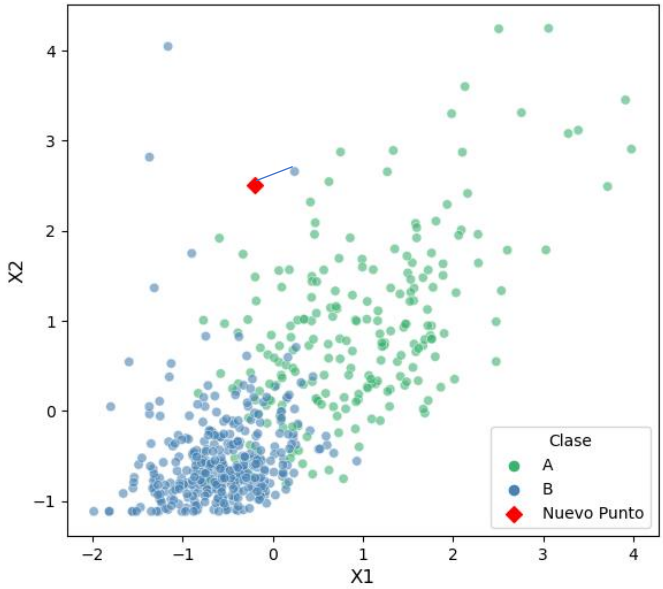


Figura 8: Diagrama de dispersión de X_1 frente X_2 coloreado por la etiqueta de clasificación con una nueva observación representada por el punto rojo y su cercano más próximo.

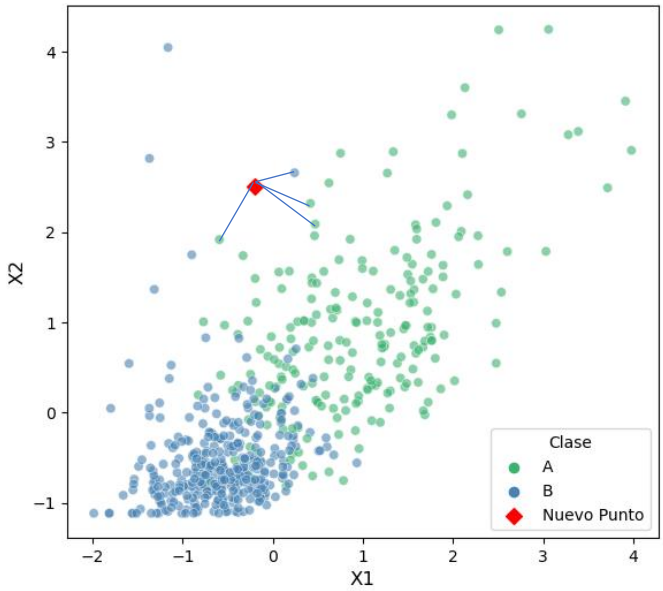


Figura 9: Diagrama de dispersión de X_1 frente X_2 coloreado por la etiqueta de clasificación con una nueva observación representada por el punto rojo y sus cuatro cercanos más próximos.

En el ejemplo hemos elegido los puntos más cercanos trazando una línea recta entre los puntos, es decir calculando su distancia euclídea según la siguiente fórmula:

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

por lo que, para encontrar los vecinos más próximos a una nueva observación, calculamos la distancia de esa nueva observación al resto de ejemplos de conjunto de entrenamiento, seleccionando las k observaciones correspondientes a las menores K distancias.

Supongamos en nuestro ejemplo que elegimos $k = 5$. El resultado del cálculo de las distancias sería el que aparece en la Figura 10. A la vista del gráfico, como la mayoría de los puntos cercanos al nuevo son de la categoría A, parece que lo más apropiado sería clasificar el nuevo punto como A

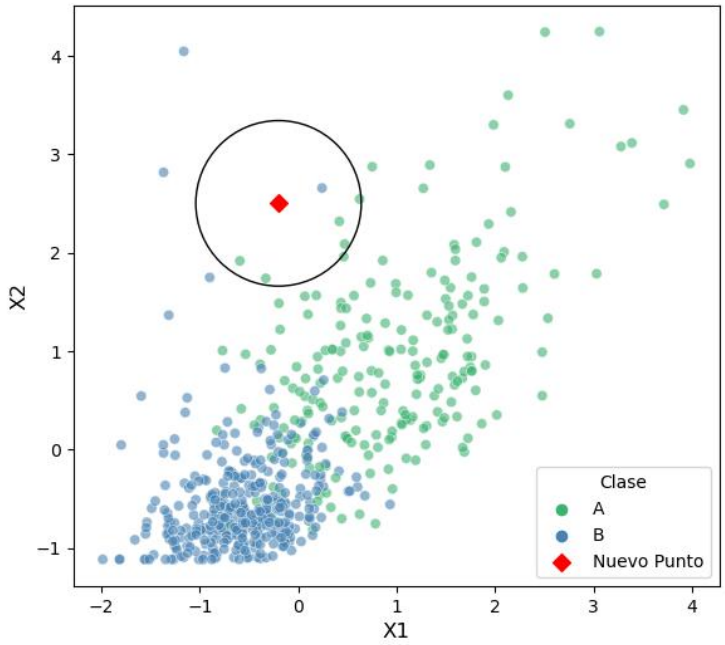


Figura 10: Diagrama de dispersión de X_1 frente X_2 con los 5 vecinos más próximos en un círculo.

Pasemos ahora a ver como funcionaría el algoritmo si hubiera más de dos características para clasificar el nuevo ejemplo. Aunque lo expuesto anteriormente se refería únicamente a dos variables predictoras, el mismo algoritmo de k vecinos próximos se utiliza cuando se tiene un mayor número de variables predictoras. Cada variable predictora puede darnos

nueva información para mejorar la clasificación. La única diferencia es el cálculo de la distancia entre puntos que pasaría a ser

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \cdots + (z_2 - z_1)^2}$$

que es una extrapolación de la fórmula anterior para un espacio con más dimensiones.

En nuestro ejemplo, si añadimos una nueva característica X_3 y elegimos los $K = 5$ vecinos más próximos tendríamos el gráfico de la Figura 11; como la mayoría de los vecinos próximos son de la categoría A, clasificaríamos el nuevo punto como A.

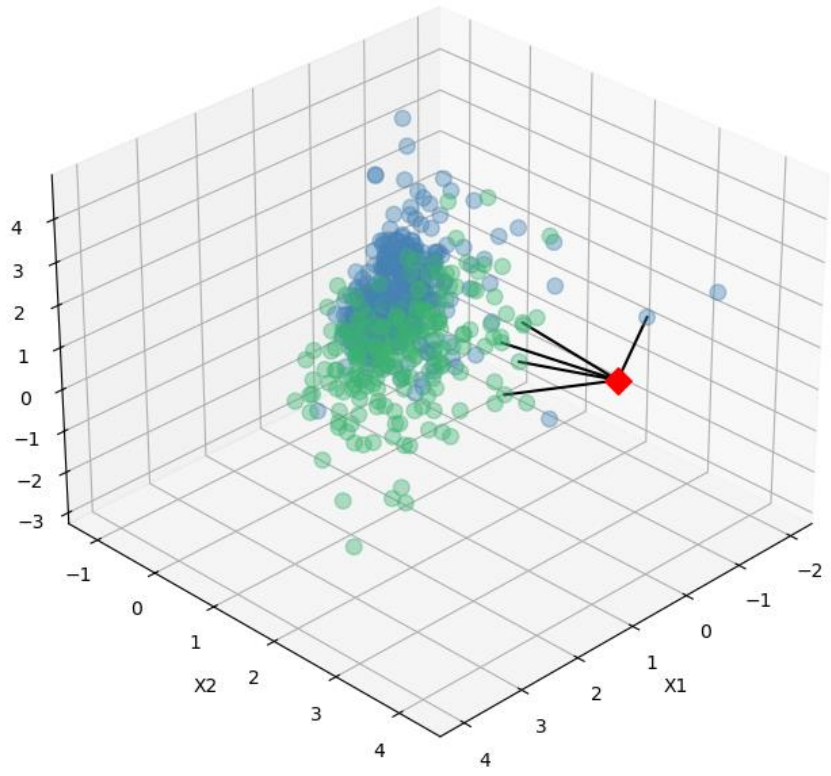
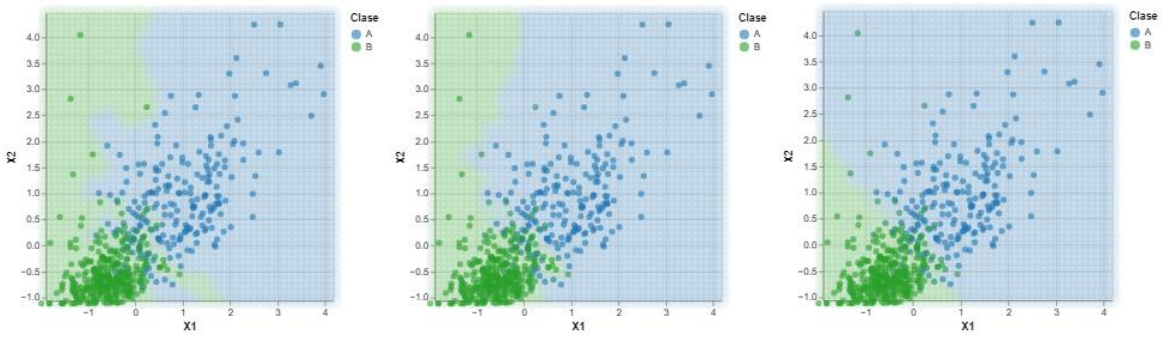


Figura 11:Diagrama de dispersión 3D de las variables de X_1, X_2 X_3 y los $k = 5$ vecinos más cercanos

Resumiendo, el proceso de clasificación con k-NN implica los siguientes pasos:

1. Cálculo de la distancia: Para un nuevo ejemplo, se calcula su distancia con respecto a todos los ejemplos en el conjunto de entrenamiento. La distancia más comúnmente utilizada es la distancia euclídea, aunque también se pueden utilizar otras medidas de distancia según la naturaleza de las características y la tarea en cuestión.
2. Selección de los vecinos: Se eligen los *k* puntos más cercanos al nuevo ejemplo según la distancia calculada en el paso anterior.
3. Realización de la clasificación: Se asigna al nuevo ejemplo la clase más frecuentemente representada entre sus *k* vecinos más cercanos en el caso de clasificación, y el promedio de las salidas de los *k* vecinos más cercanos para una tarea de regresión.

Como ya hemos mencionado, la elección adecuada del parámetro *k* en el algoritmo de *k*-vecinos más cercanos (k-NN) es crucial para obtener modelos de aprendizaje automático con buena capacidad de generalización. El valor de *k* determina la cantidad de vecinos más cercanos que se consideran al realizar una predicción, y afecta directamente a la capacidad del modelo para capturar patrones en los datos y evitar tanto sobreajuste como subajuste. Este caso se ilustra en la Figura 12, donde el color de fondo indica la decisión de clasificación para distintos valores de *k*.



(a) *(b)* *(c)*

Figura 12: Gráfico de dispersión de las variables X_1, X_2 donde el color de fondo que indica la decisión en la clasificación para distintos valores de k. (a) k = 1, (b) k = 5 y (c) k = 20.

Otro aspecto relevante a tener en cuenta al utilizar esta u otras técnicas de aprendizaje automático es el de la normalización de las características. Los datos a los que nos enfrentamos pueden estar medidos en diferentes escalas, unos pueden ser muy grandes y

otros muy pequeños, y normalizar o escalar los datos sirve para que todas las características contribuyan de modo equitativo al modelo, ya que, a la hora de calcular distancias, los valores grandes pueden dar resultados diferentes a los pequeños. Además, la normalización facilita la comparación de las características al ponerlas todas en la misma escala.

Los métodos más utilizados para normalizar son dos:

- Escalado Min-Max:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

  Que transforma cada característica para que todos estén entre 0 y 1.

- Estandarización:

$$x' = \frac{x - media}{desv.tipica}$$

  Transforma cada característica para que tengan una media de 0 y una desviación estándar de 1.

Una de las principales ventajas del método k-NN es que es fácil de entender e implementar, y no requiere la estimación de parámetros adicionales. Por otro lado, su principal limitación es su coste computacional, especialmente cuando se trabaja con grandes conjuntos de datos o con un alto número de características. Además, k-NN tiende a no funcionar bien en conjuntos de datos con características irrelevantes o con desequilibrios en la distribución de las clases.

A pesar de sus limitaciones, el método k-NN sigue siendo muy utilizado en la práctica debido a su simplicidad. A menudo se utiliza como un primer punto de referencia para compararlo con otros algoritmos más avanzados en tareas de clasificación y regresión en el aprendizaje automático.

5. Análisis y resultados

Vamos ahora a comenzar ahora a elaborar el modelo que permitirá predecir las emociones percibidas resultantes de un sonido mediante la activación y la valencia. En este caso la

respuesta que esperamos obtener del modelo es numérica, en lugar de categórica, como hemos usado en el ejemplo, por lo que nos enfrentamos a un caso de regresión. Sin embargo, muchos de los conceptos explicados para el caso de clasificación son similares en regresión. Por ejemplo, la variable de respuesta de una nueva observación se obtendrá a partir de las variables de respuesta de observaciones de entrenamiento similares en el conjunto de características. Al construir un modelo de regresión, al igual que en el caso de la clasificación, dividiremos nuestros datos en conjuntos de entrenamiento y test. Usaremos las librerías de python de *scikit-learn*, para aplicar el enfoque de *k* vecinos más cercanos (k-NN), para hacer predicciones además de explicar cómo elegimos el valor de *k* más apropiado.

5.1 Preprocesamiento de datos

Tal y como se ha comentado en el punto 2, para la realización del trabajo se utilizará la base de datos de paisajes sonoros Emo-soundscapes database (EMO) (Fan et al., 2017) que podría considerarse como la más grande disponible públicamente con anotaciones de etiquetas emocionales. EMO contiene 1213 clips de audio que se publicaron bajo licencia Creative Commons en la plataforma colaborativa de audio Freesound. Como hemos expuesto anteriormente, la taxonomía de Schafer clasifica los clips seleccionados en seis categorías debido a su generalidad y simplicidad. Las categorías de Schafer consideran tanto la identificación de la fuente como el contexto auditivo: sonidos naturales, sonidos humanos, sonidos y sociedad, sonidos mecánicos, tranquilidad y silencio, y sonidos como indicadores. EMO consta de 100 clips de audio por categoría dentro de un primer subconjunto (es decir, 600 clips de audio) y 613 sonidos mezclados manualmente de dos o tres categorías del primer subconjunto. Un procedimiento de *crowdsourcing* proporcionó anotaciones de activación y valencia percibidas, mediante un cuestionario basado en clasificaciones de comparación de dos clips.

Los clips de audio son monofónicos y las grabaciones monofónicas son suficientes para evaluar la activación (*arousal*) y la valencia (*valence*) de los entornos acústicos, entre muchos otros indicadores que pueden usarse para el análisis de paisajes sonoros. EMO utilizó tanto YAAFE (Mathieu, et al., 2010) como MIRToolbox (Lartillot, et al, 2008) para la extracción de 123 características de audio. Dichas características se toman normalizadas por lo que todas tienen un valor entre 0 y 1.

Hay tres grupos principales de características de las señales de audio (San Millán et al, 2022):

1. *Características psicoacústicas*: Estas características representan atributos perceptuales (es decir, subjetivos) de los sonidos, como el nivel (es decir, la sonoridad para el nivel general y los MFCC para los niveles limitados en banda), el espectro (es decir, la nitidez para las altas frecuencias) y la modulación temporal y espectral (es decir, la fluctuación).
2. *Características en el dominio del tiempo*: Estas características representan la dinámica de la señal, como los estimadores clásicos basados en muestras de la señal de audio (es decir, energía, entropía de energía o raíz cuadrada del valor cuadrático medio (RMS)), la relación entre la diferencia de magnitud al principio y al final de un período de decaimiento (es decir, pendiente de disminución) y el porcentaje de tramas que muestran menos energía que la energía promedio (es decir, baja energía).
3. *Características en el dominio de la frecuencia*: Estas características representan la forma del espectro y la estructura armónica de los sonidos, como la frecuencia fundamental de la señal de audio (es decir, tono), la proporción de frecuencias que no son múltiplos de la frecuencia fundamental (es decir, inarmonía), la representación espectral basada en las 12 notas temperadas igualmente del sistema musical occidental (es decir, cromagramas), los momentos estadísticos espectrales (es decir, centroide, el primero; dispersión, el segundo; asimetría, el tercero), la relación entre la media geométrica y la media aritmética (es decir, planitud), los cambios espectrales entre dos tramas sucesivas (es decir, flujo) y la estimación de la cantidad de alta frecuencia (es decir, atenuación).

Ya hemos indicado anteriormente la importancia de utilizar características normalizadas. En este caso, las características se han normalizado con el método de escalado Min-Max.

Pasamos ahora a analizar los valores de activación y valencia. Para ello utilizamos un gráfico de dispersión entre ambas series de datos. Como puede observarse en la Figura 13, la mayoría de las observaciones se encuentran localizadas en una diagonal, indicando que no hay registros sonoros con un *arousal* y *valence* cercanos ambos al -1, ni tampoco registros con *arousal* y *valence* con valores igual a 1, conjuntamente.

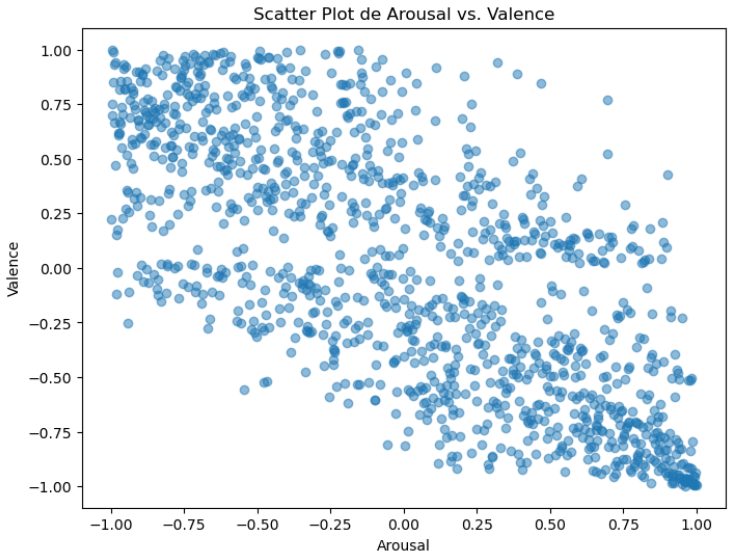


Figura 13: Diagrama de dispersión de los datos arousal y valence.

La Figura 14 muestra el histograma de las variables a explicar, que como puede observarse tiene forma "plana", lo que implica que los valores de estas variables dependientes, en el conjunto de datos disponible están distribuidas de manera relativamente uniforme a lo largo del rango de valores de cada variable. Esto implica que no hay valores que ocurran con mucha más frecuencia que otros.

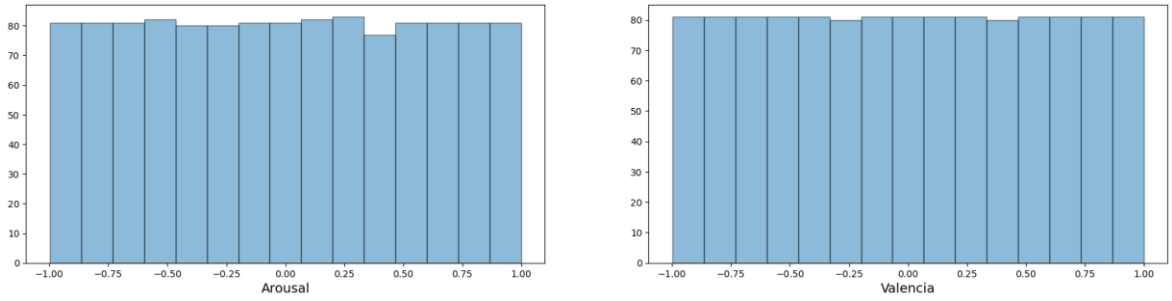


Figura 14: Histogramas de arousal y valence

A modo de ejemplo se han representado en la Figura 15 diagramas de dispersión tri-dimensionales con los valores de arousal, valence y una de las características de alguno de los tres grupos principales de características (loudness_mean, brightness_mean, decreaseslope_mean, eventsensity_mean, perceptual_sharp_mean, mfcc_mean_13, chromagram_mean_3 y spectral_var_mean). En esta figural una mayor intensidad del color indica un mayor número de observaciones en esa parte del espacio.

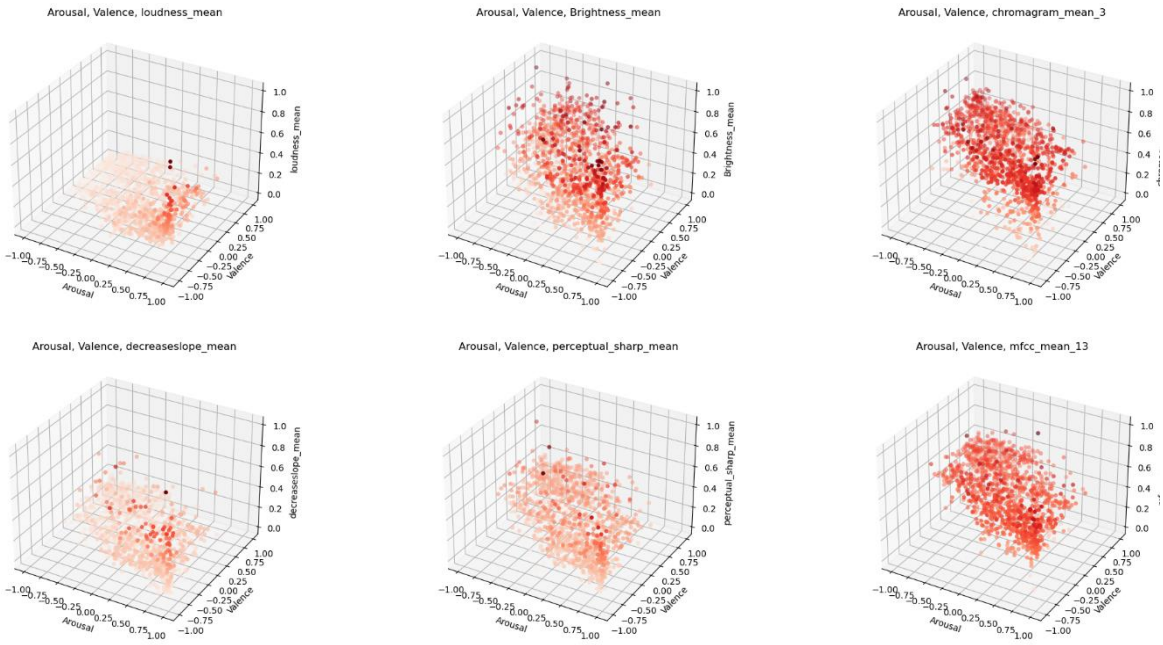


Figura 15: Diagramas de dispersión 3D de los valores de arousal, valencia y distintas características

Si observamos ahora el coeficiente de correlación lineal entre pares de variables del modelo, el *arousal*, la *valence* y las distintas características tal y como aparecen en la Figura 16, podemos observar que algunas de las características parecen presentar una correlación muy elevada (zonas azul oscuro), lo que puede ser un indicio de que ambas variables podrían contener información similar (esto es, en terminología de aprendizaje automático, podrías ser variables redundantes). Sin embargo, una vez analizadas las correlaciones numéricas y seleccionando aquellas que tienen un valor mayor que 0.8, vemos que individualmente ninguna presenta una correlación tan elevada con las variables *arousal* y valencia, por lo que decimos utilizarlas todas para construir el modelo final.

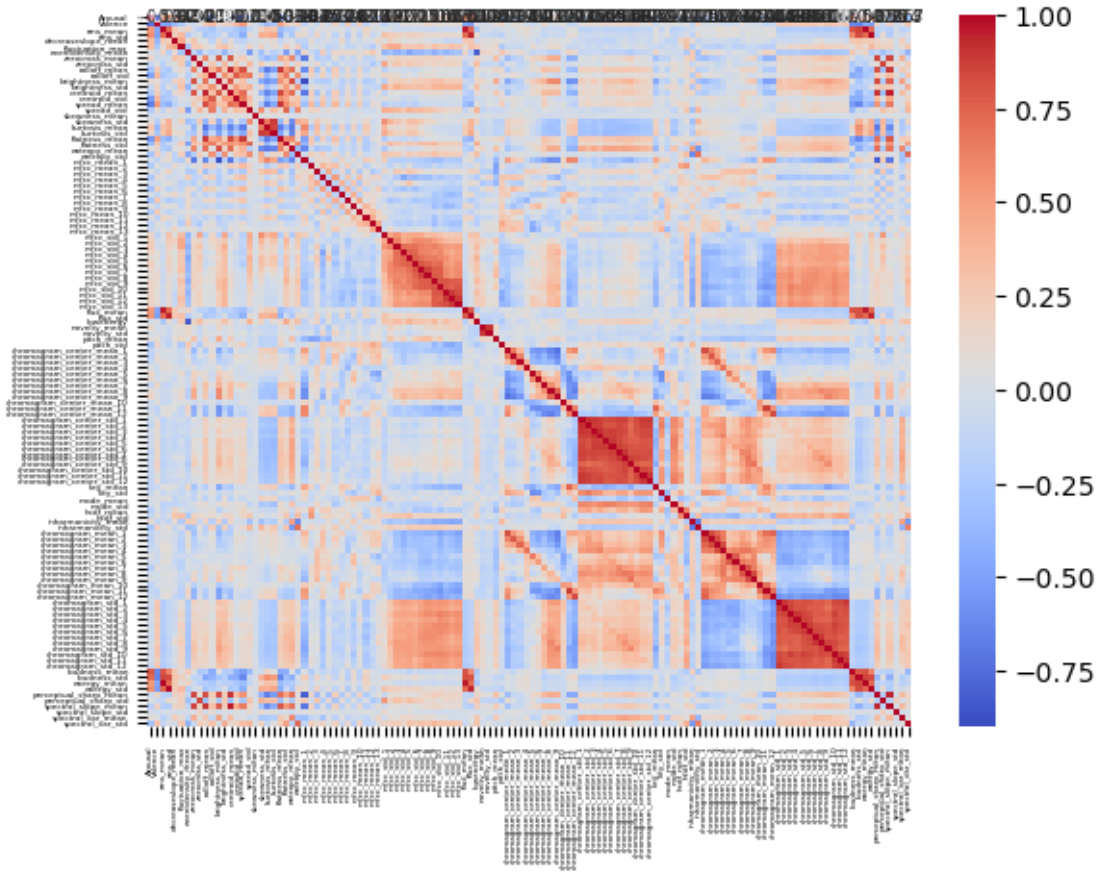


Figura 16: Mapa de correlaciones entre las características

5.2 Modelización

Vamos a comenzar la modelización aplicando el método k-NN al caso unidimensional. Primero, explicaremos una de las variables utilizando solo una característica. Esto nos permitirá entender el procedimiento básico. Luego, pasaremos al caso multivariante, donde explicaremos una única variable con todas las características disponibles. Finalmente, desarrollaremos un único modelo que explique ambas variables (*arousal* y *valence*) utilizando todas las características.

Además, para el análisis vamos a proceder siguiendo el procedimiento que figura en el esquema de la Figura 5.

5.2.1 Modelización univariante

Comenzaremos la modelización aplicando el método k-NN al caso univariante, en el que intentaremos explicar la valencia o el nivel de placer utilizando únicamente la variable "*loudness_mean*". Hemos seleccionado "*loudness_mean*" porque creemos que el volumen puede ser un buen predictor del sentimiento de agrado o desagrado. Se espera una correlación negativa entre estas variables, ya que, a mayor volumen, menor será la sensación de agrado. De hecho, la correlación estimada entre ambas variables es de -0.62.

Este primer paso permitirá entender el procedimiento básico de la modelización k-NN en un contexto univariante. Como ya hemos dicho, posteriormente ampliaremos el análisis al caso multivariante, donde utilizaremos todas las características disponibles para explicar la valencia. Finalmente, desarrollaremos un modelo que integre todas las características para explicar ambas variables objetivo.

La representación del diagrama de dispersión de las dos variables, así como el histograma de la variable *loudness_mean* aparecen en la Figura 17.

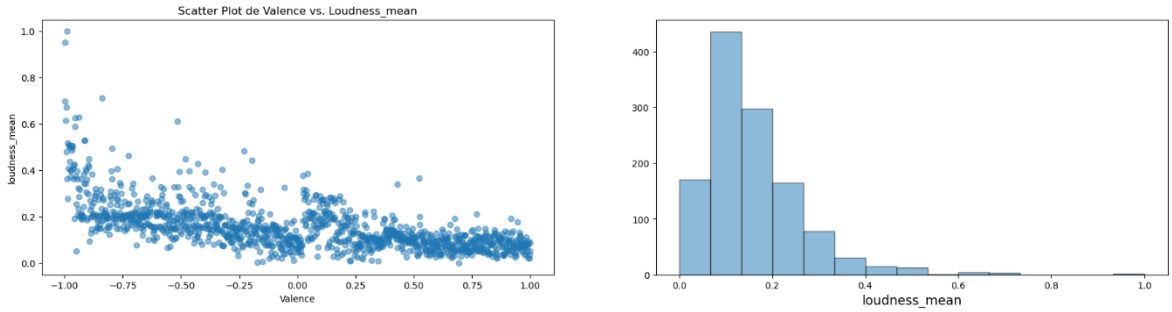


Figura 17: Gráfico de dispersión entre valence y loudness_mean,histograma de loudness_mean

Dividimos aleatoriamente la muestra (conjunto total de observaciones) en dos partes independientes: entrenamiento, correspondiente al 80% de tamaño de la muestra original (970); y la segunda, test, que corresponde al 20% (243) restante.

El siguiente paso es elegir el mejor valor del parámetro *k*. Para ello, utilizamos como criterio de elección el mínimo MSE; para ello se van explorando distintos valores de *k* y se elige el

que proporcione el menor MSE sobre un conjunto independiente. Encontramos (Figura 18) que el menor MSE en validación es 0.173, que corresponde al valor $k = 39$.

Para seleccionar el valor del hiperparámetro se utiliza la técnica de validación cruzada con n subconjuntos (n-fold cross validation):

- El conjunto de diseño se divide aleatoriamente en n subconjuntos del mismo tamaño.
- Las observaciones de (n-1) subconjuntos se juntan y se utilizan como conjunto de aprendizaje para diseñar el modelo. El subconjunto no utilizado en el aprendizaje se reserva como conjunto de validación para estimar las prestaciones del modelo.
- El proceso anterior se repite n veces, dejando cada vez un subconjunto diferente como conjunto de validación. Esto significa que se obtienen n medidas de prestaciones diferentes, una por cada subconjunto de validación. Las gráficas que se muestran a continuación representan el valor medio de los n resultados, tanto para las particiones de entrenamiento como de validación.
- Una vez seleccionado el valor del hiper-parámetro, el modelo final se diseña utilizando todo el conjunto de observaciones disponible para realizar el diseño. Es decir, se juntan las observaciones de los n subconjuntos, se diseña el modelo y sobre ese modelo se estima su capacidad de generalización considerando el conjunto de test (independiente del de validación).

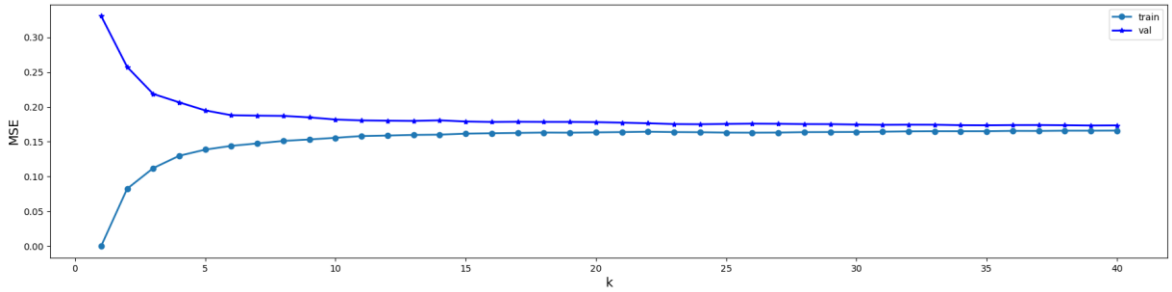


Figura 18: Figura de mérito de MSE versus k para los conjuntos de entrenamiento y validación, cuya unión corresponde al conjunto de diseño.

Observando el gráfico, vemos que el MSE sobre el conjunto de validación se estabiliza en su valor más pequeño a partir de $k = 8$. Puesto que este esquema de aprendizaje construye modelos locales, decidimos usar ese valor para construir el modelo con el conjunto de diseño

y evaluamos las prestaciones en test calculando MSE, RMSE y MAE para el conjunto de test. Los resultados obtenidos nos proporcionan los siguientes valores:

MSE en el conjunto de test cuando se utiliza sólo la característica *loudness_mean* en k-NN para predecir valencia = 0.175.

RMSE en el conjunto de test se utiliza sólo la característica *loudness_mean* en k-NN para predecir valencia = 0.418.

MAE en el conjunto de test cuando se utiliza sólo la característica *loudness_mean* en k-NN para predecir valencia = 0.321.

Si representamos las estimaciones sobre el conjunto de diseño se obtiene la Figura 19.

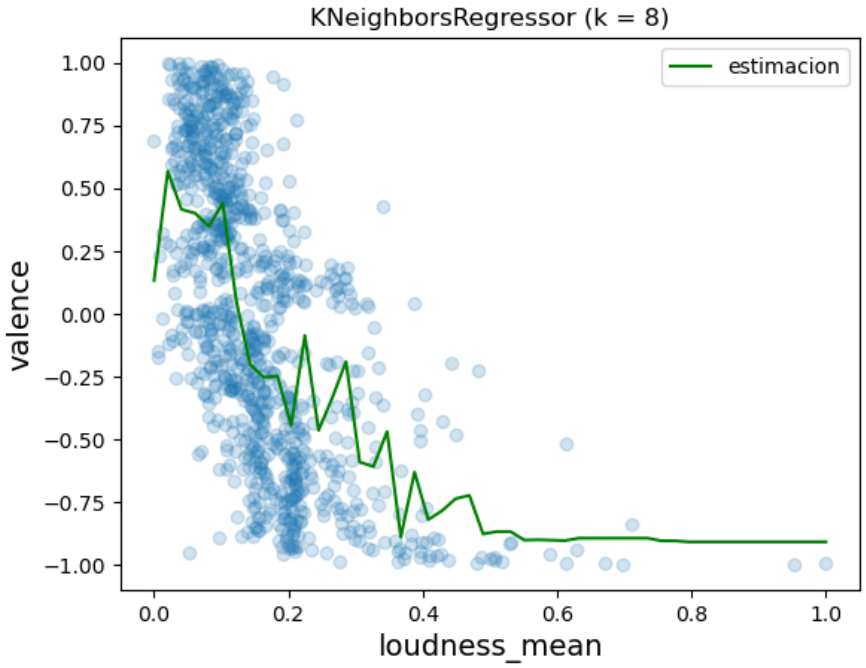


Figura 19: Estimación del modelo cuando se utiliza sólo la característica loudness_mean en K-nn para predecir valencia y el número de vecinos elegido es 30.

A modo de ejemplo, vamos a representar el modelo estimado con dos valores diferentes de *k*. Como vemos en la *k* = 1 (Figura 21) da lugar a un sobreajuste del modelo lo que implicará

que no ajustará correctamente cuando aparezca un dato nuevo, mientras que $k = 80$ (Figura 21) no permite explicar el modelo con precisión.

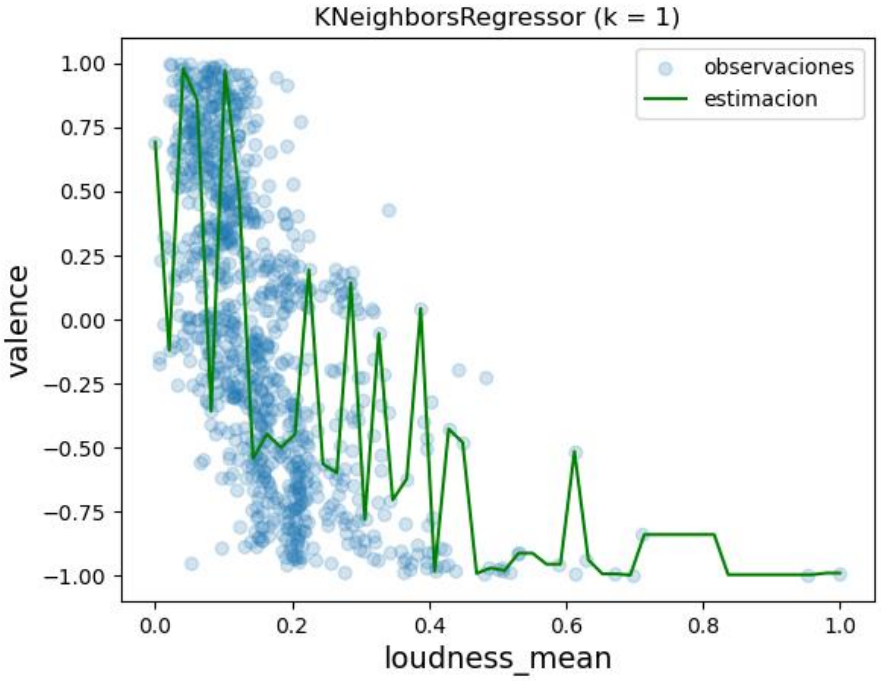


Figura 20: Estimación del modelo cuando se utiliza sólo la característica "loudness_mean" en k-NN para predecir valencia y el número de vecinos elegido es 1.

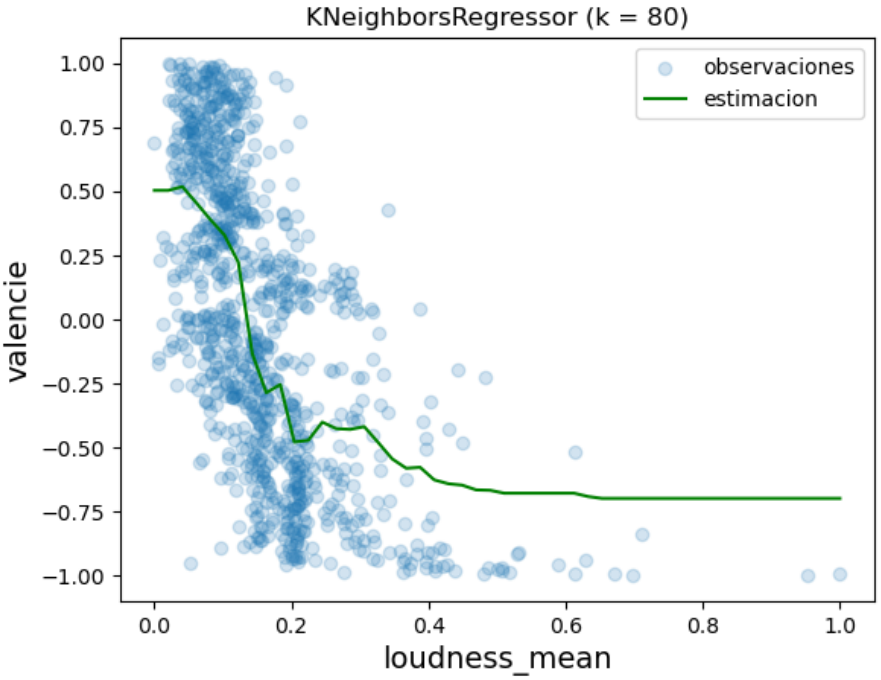


Figura 21: Estimación del modelo cuando se utiliza sólo la característica "loudness_mean" en k-NN para predecir valencia y el número de vecinos elegido es 80.

En todo caso, se observa, que con una única variable la precisión del modelo no es muy elevada. Por lo que repetimos el procedimiento, pero en este caso explicando la valencia con todas las variables.

5.2.1 Modelización multivariante

Iniciaremos el proceso de modelización multivariante centrándonos inicialmente en el *arousal*, una de las variables de interés. Para ello, utilizaremos todas las características disponibles en nuestro conjunto de datos.

Una vez que hayamos construido y validado el modelo para *arousal*, procederemos a modelizar la variable *valence* de manera similar, aprovechando también todas las características disponibles.

Finalmente, buscaremos generalizar nuestro enfoque desarrollando un modelo que prediga ambas variables de manera conjunta, utilizando todas las características. Este modelo multivariante no solo proporcionará predicciones más robustas, sino que también facilitará la exploración de las relaciones interdependientes entre *arousal* y *valencia*.

5.2.1.1. Modelización de la valencia utilizando todas las características.

La figura de mérito para el conjunto de diseño de la variable valencia se representa en la Figura 22. Tras un análisis detallado del gráfico, decidimos utilizar $k = 5$ en el proceso de test. Con este valor de k, se obtienen los siguientes parámetros en el conjunto de test, demostrando una mejora significativa en la precisión del modelo,

MSE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-NN para predecir valencia = 0.127
RMSE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-NN para predecir valencia = 0.356

MAE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-NN para predecir valencia = 0.272.

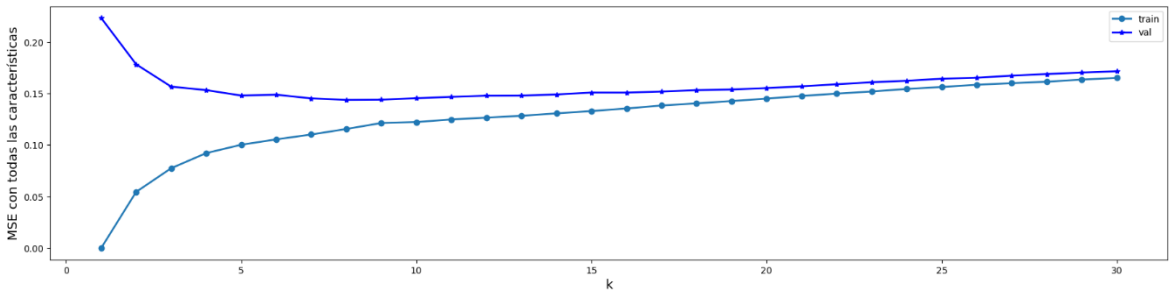


Figura 22: Figura de mérito de MSE versus k para los conjuntos de entrenamiento y validación, cuya unión corresponde al conjunto de diseño para explicar la valencia con todas las características.

El gráfico de dispersión entre los valores reales y los valores estimados es una herramienta fundamental para evaluar el rendimiento de un modelo de predicción, en nuestro caso, la Figura 23 muestra la relación entre la valencia real y la valencia estimada utilizando el algoritmo k-NN (k-vecinos más cercanos). Cada punto en el gráfico representa una observación, donde el eje horizontal indica el valor real de la valencia y el eje vertical muestra el valor estimado por el modelo. La línea discontinua roja representaría el ajuste perfecto, el valor real y el estimado coincidirían. En primer lugar, si los puntos siguen una tendencia con pendiente positiva, en general, el modelo está capturando la relación esperada: a medida que la valencia real aumenta, la valencia estimada también lo hace. Sin embargo, si la línea se ajusta mal o tiene una pendiente negativa, eso indicaría un problema con el modelo. Además, si la mayoría de los puntos están cerca de la línea discontinua roja, esto sugiere que el modelo k-NN está haciendo buenas predicciones. En general, un ajuste cercano a esta línea indica que los valores estimados se aproximan a los valores reales. Si, por el contrario, hay puntos con una gran discrepancia entre el valor real y el valor estimado (es decir, puntos lejanos a la línea), esto significa que hay errores de predicción. Un buen modelo debería minimizar la cantidad y el tamaño de estos errores. En nuestro caso, se puede notar que algunos valores se encuentran un poco alejados de la línea roja, lo que sugiere que las predicciones para esos puntos podrían no ser precisas.

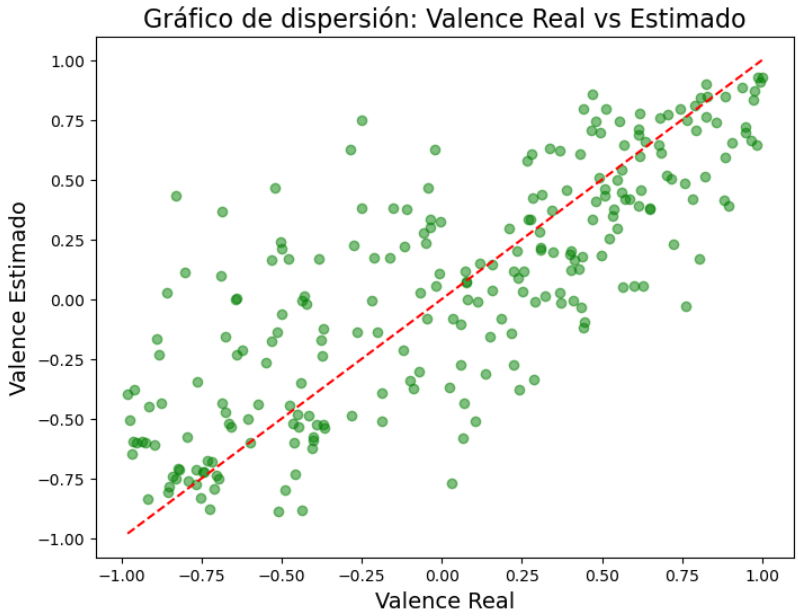


Figura 23: Gráfico de Dispersión: Valence Real vs Estimado

5.2.1.2. Modelización de arousal utilizando todas las características.

La Figura 24 muestra la figura de mérito para el conjunto de diseño con relación a los posibles valores de *k* y permite observar que el número de vecinos *k* que minimiza este valor es 4.

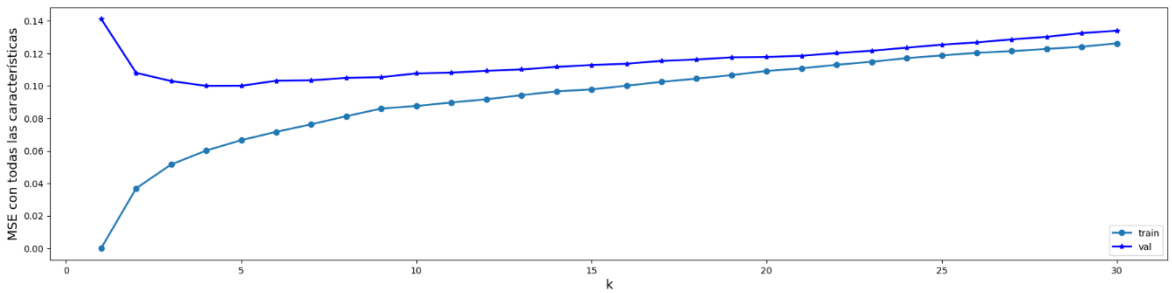


Figura 24: Figura de mérito de MSE versus k haciendo uso del conjunto de diseño para explicar el arousal con todas las características.

Observando el gráfico y haciendo algunas pruebas adicionales, decidimos utilizar el parámetro $k = 4$, ya que se observa que desde ese punto empieza a crecer el MSE. Con este valor de k, se obtienen los siguientes parámetros para el conjunto de test:

MSE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-NN para predecir arousal = 0.089

RMSE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-NN para predecir arousal = 0.298

MAE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-NN para predecir arousal = 0.227

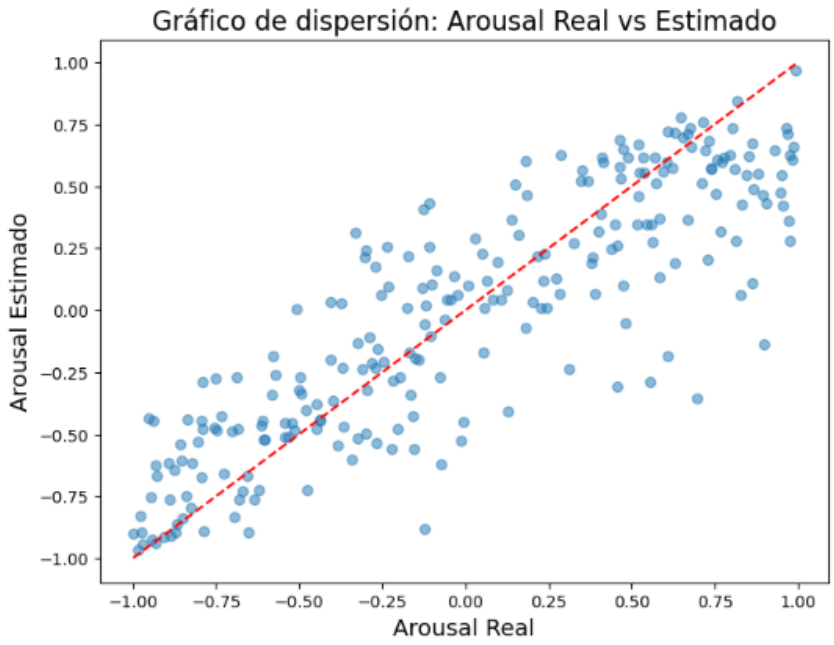


Figura 25: Gráfico de Dispersión: Arousal Real vs Estimado

Observando el gráfico de dispersión entre los valores reales y los valores estimados de la Figura 25, se puede observar que los puntos siguen una tendencia de pendiente positiva, lo que indica que el modelo está logrando capturar la relación esperada: a medida que el *arousal* real aumenta, su valor estimado también tiende a incrementarse. Además, hay un mejor ajuste general alrededor de la línea central que en el caso anterior, lo que sugiere que, en muchos de los casos, los valores estimados son bastante cercanos a los valores

reales. Sin embargo, algunos puntos se desvían un poco de la línea roja, lo que indica que las predicciones para esos casos podrían no ser precisas.

5.2.1.3. Modelización de arousal y la valence utilizando todas las características.

Continuamos como en los casos anteriores al analizar la figura de mérito (Figura 26) y llevar a cabo algunas pruebas adicionales. La elección del hiperparámetro se establece en k = 5, dado que se observa un aumento en el MSE a partir de ese punto.

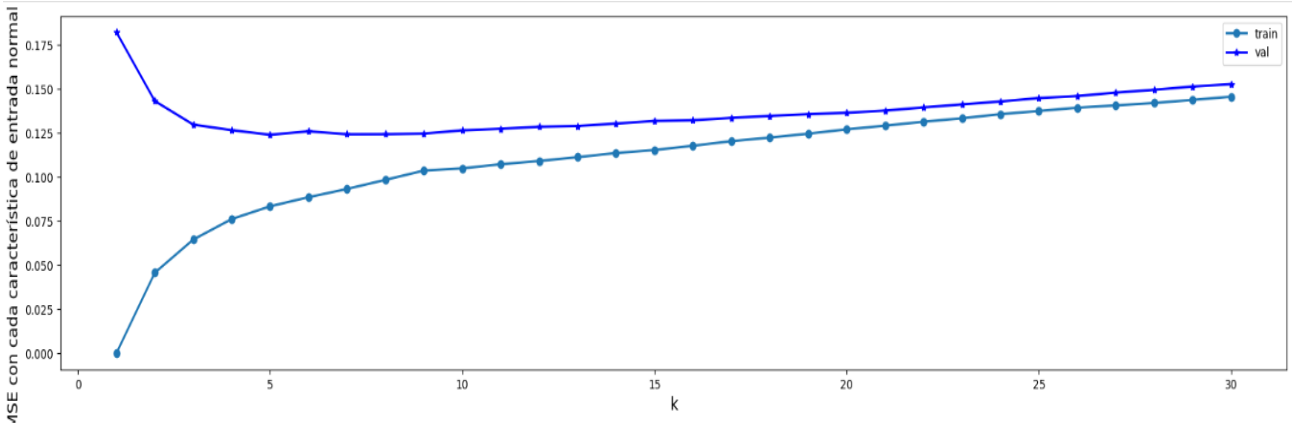


Figura 26: Figura de mérito de MSE versus k haciendo uso del conjunto de diseño para explicar el arousal y la valencia con todas las características.

Utilizando el parámetro k = 5, los resultados obtenidos en el conjunto de test cuando se modelizan *arousal* y *valence* y todas las características es:

MSE en el conjunto de test cuando se utilizan todas las características (normalizadas) para 'valence' = 0.118

RMSE en el conjunto de test cuando se utilizan todas las características (normalizadas) para 'valence' = 0.291

MAE en el conjunto de test en el conjunto de test cuando se utilizan todas las características (normalizadas) para 'valence' = 0.257

MSE en el conjunto de test cuando se utilizan todas las características (normalizadas) ‘arousal' = 0.084

RMSE en el conjunto de test cuando se utilizan todas las características (normalizadas) para 'arousal' = 0.344

MAE en el conjunto de test cuando se utilizan todas las características (normalizadas) 'arousal' = 0.219

Estos resultados reflejan una mejora significativa en la precisión del modelo, tanto para la predicción de '*valence*' como para '*arousal*'. A la vista de los resultados podemos decir que el enfoque multivariante permite modelizar mejor las variables. Las mejoras son evidentes, demostrando que el uso de múltiples características en la modelización proporciona mejores resultados en la predicción de la activación (*arousal*) y la valencia (*valence*), así como en la predicción conjunta de ambas variables.

La Tabla 2 compara los resultados de aplicar del modelo de k-NN bajo diferentes configuraciones de características y variables objetivo. En la primera columna aparecen los resultados solo se utiliza la variable *loudness_mean* para predecir valencia, el error cuadrático medio (MSE) es de 0.155, la raíz del error cuadrático medio (RMSE) es 0.418 y el error absoluto medio (MAE) es de 0.321. Estos sugieren que una sola característica no es suficiente para capturar la variabilidad en valencia. Al incorporar todas las características para predecir valencia (columna 2), se observa una reducción tanto en el MSE (0.127), en el RMSE (0.356) como en el MAE (0.272). Esto demuestra que un enfoque multivariante mejora la precisión del modelo, al proporcionar más información para realizar predicciones más precisas. La tercera columna muestra los errores al predecir el *arousal* utilizando todas las características, el modelo logra un MSE de 0.084, un RMSE de 0.356 y un MAE también de 0.272. Finalmente, el caso más completo, donde se modelizan tanto *valence* como *arousal* utilizando todas las características disponibles se presenta la columna 4, con los menores valores de MSE y MAE para ambas variables. Para *valence*, el MSE es 0.1187, el RMSE es 0.344 y el MAE es 0.257, mientras que para *arousal*, el MSE es 0.084, el RMSE es 0.291 y el MAE es 0.219. Esto confirma que el modelo multivariante, que considera todas las características para predecir ambas variables simultáneamente, ofrece la mejor capacidad predictiva.

Tabla 2: Resumen de resultados de los distintos modelos al evaluar el conjunto de test

Modelo	Valence con Loudness_mean	Valence con todas las características	Arousal con todas las características	Valence y arousal con todas las características
MSE	0.155	0.127	0.084	Arousal =0.084 Valence = 0.118
RMSE	0.418	0.356	0.298	Arousal =0.291 Valence = 0.344
MAE	0.321	0.272	0.227	Arousal = 0.219 Valence = 0.257

Al examinar el gráfico de dispersión (Figura 27) que ilustra la relación entre los valores reales y estimados del *arousal*, se observa una clara tendencia de pendiente positiva. Esto sugiere que el modelo ha logrado captar adecuadamente la relación esperada: a medida que aumenta el *arousal* real, también lo hacen los valores estimados. Además, se nota un mejor ajuste en torno a la línea central en comparación con casos anteriores, lo que indica que, en muchas ocasiones, las estimaciones son bastante cercanas a los valores reales. Sin embargo, algunos puntos se desvían ligeramente de la línea roja, lo que indica que en esos casos específicos las predicciones pueden no ser precisas.

En cuanto al gráfico que representa la valencia, se observa una tendencia similar, con los puntos alineándose a lo largo de una línea de pendiente positiva. No obstante, en este caso, los resultados muestran una mayor dispersión en relación con la línea roja, lo que sugiere que el modelo predice la valencia con menor precisión que el arousal.

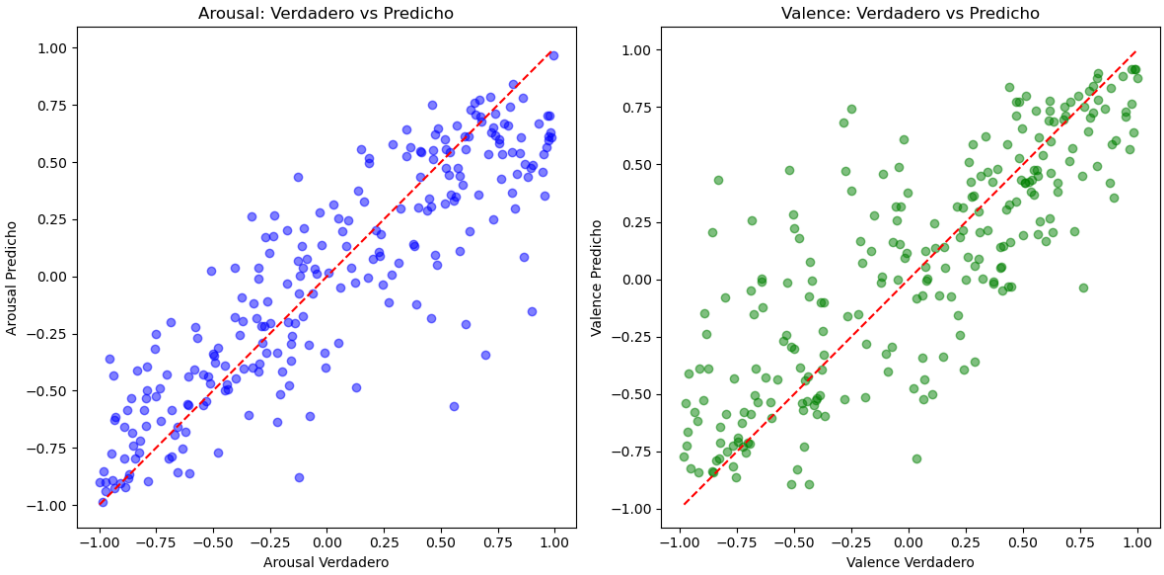
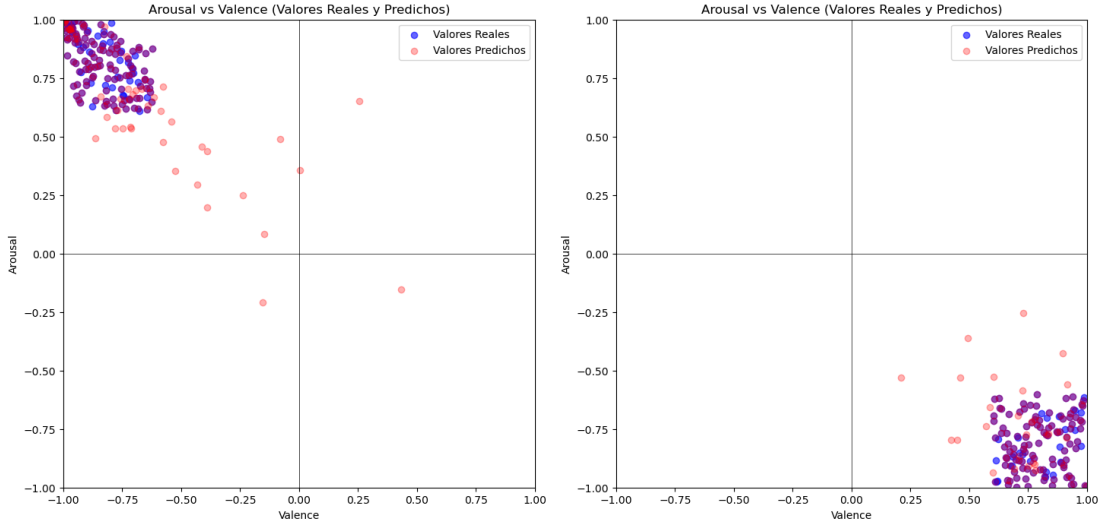


Figura 27: Gráfico de Dispersión: Arousal Real vs Estimado y Valence Real vs. estimado

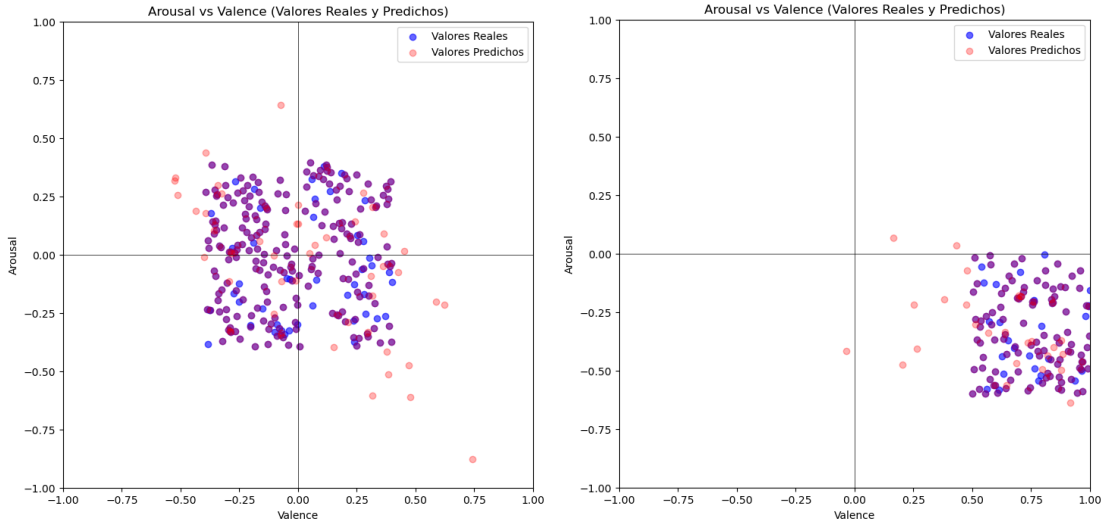
6. Análisis de Resultados para Casos Extremos en el Esquema Circumplexo

A continuación, se presentan gráficos que ilustran cómo se ajustan los resultados del modelo para los casos extremos de arousal y valence, así como para el estado neutral. Estos gráficos permiten observar cómo se comportan las estimaciones en situaciones donde las emociones son más intensas o donde no se presentan emociones significativas, proporcionando una perspectiva clara sobre la capacidad del modelo para capturar variaciones en el espectro emocional. Al enfocarnos en estos puntos extremos dentro del esquema circumplexo, se busca evaluar la efectividad del modelo en la identificación y predicción de emociones en situaciones donde los valores de arousal y valencia alcanzan sus máximos o mínimos, así como en la zona de neutralidad.



a) b)

Figura 28: Relación entre Arousal y Valencia: Comparación de Valores Reales y Predichos en el esquema circumplexo para: a) arousal entre valores [0.6,1] y valence entre[-0.6,1] b) arousal entre valores [-1,0.6] y valence entre [0.6,1]



a) b)

Figura 29: Relación entre Arousal y Valencia: Comparación de Valores Reales y Predichos en el esquema circumplexo para: a) arousal entre valores [-0.4, 0.4] y valence entre [-0.4 ,0.4] b) arousal entre valores [-0.6,0] y valence entre [0.5,1]

En general, la Figura 28 y la Figura 29 muestran que el modelo logra predecir adecuadamente la relación entre el arousal y la valencia tanto para los valores extremos como para los intermedios, con la mayoría de los puntos de valores predichos alineándose de manera razonable con los valores reales. Sin embargo, se puede observar que algunos datos de los valores predichos se encuentran significativamente alejados de los valores reales, lo que indica que existen casos específicos en los que el modelo no logra captar con precisión la dinámica entre estas dos dimensiones emocionales.

7. Análisis de la posibilidad de extracción de características

El proceso de extracción de características se ha realizado utilizando la biblioteca *Librosa* de Python (McFee et al., 2015), utilizada en el análisis de audio. Esta biblioteca proporciona diversas herramientas para extraer información significativa de señales de audio, que luego pueden ser utilizadas en aplicaciones de machine learning.

Sin conocer exactamente que mide cada una de las características, pero guiándome de las recomendaciones sobre las características son comúnmente utilizadas en estudios de análisis de audio, música, y reconocimiento de voz más relevantes he seleccionado las siguientes:

1. *Espectrograma de frecuencia*: Representa cómo la energía de diferentes frecuencias varía con el tiempo. Es útil para analizar la evolución temporal de las características frecuenciales.
2. *Espectro de potencia*: Muestra la distribución de la potencia de la señal en distintas frecuencias, lo que permite identificar la cantidad de energía presente en diferentes bandas de frecuencia.
3. *MFCC (Coeficientes Cepstrales en Frecuencia Mel):* Son coeficientes que capturan la envolvente espectral de la señal y son esenciales en tareas de reconocimiento de audio y voz.
4. *Rolloff espectral:* Representa la frecuencia por debajo de la cual se encuentra un porcentaje predeterminado de la energía total del espectro, lo que ayuda a caracterizar la distribución espectral.
5. *Centroide espectral:* Indica el "centro de masa" del espectro de la señal y permite identificar dónde se concentra la mayor parte de la energía en términos de frecuencia.
6. *Zero-Crossing Rate (Tasa de cruces por cero):* Calcula cuántas veces la señal cruza el eje cero, útil para distinguir entre sonidos suaves y agudos.

7. *Chroma Feature (Croma):* Representa la energía distribuida a lo largo de las 12 clases de tonos (notas musicales) y es útil en el análisis de la tonalidad de la música.
8. *Spectral Bandwidth (Ancho de banda espectral):* Mide la dispersión o el rango de las frecuencias presentes en el espectro de la señal.
9. *Spectral Contrast (Contraste espectral):* Evalúa la diferencia de amplitud entre los picos y los valles en el espectro de la señal, lo que ayuda a identificar distintas texturas en la señal de audio.
10. *RMS (Root Mean Square) Energy (Energía RMS):* Calcula la energía promedio de la señal a lo largo del tiempo, útil para evaluar la intensidad del audio.

Las características extraídas se organizan de la siguiente manera y una representación gráfica de las mismas puede verse en el ANEXO 3 :

- *MFCC_1 a MFCC_13:* Los 13 coeficientes cepstrales en la escala de frecuencia Mel.
- *Zero_Crossing_Rate*: Tasa de cruces por cero de la señal.
- *Spectral_Rolloff*: Punto donde se acumula el 85% de la energía espectral.
- *Spectral_Centroid*: Centroide del espectro de frecuencias.
- *Spectral_Bandwidth*: Ancho de banda espectral.
- *Spectral_Contrast_1 a Spectral_Contrast_7*: Contraste espectral en 7 bandas de frecuencia.
- *Chroma_1 a Chroma_12:* Energía distribuida en las 12 clases de tonos (notas musicales).
- *RMS_Energy*: Energía RMS de la señal.

A continuación, para cada característica (como MFCC, Zero-Crossing Rate, etc.), se calcula tanto la media (mean) como la desviación estándar (std), proporcionando una base de datos de características semejante a la utilizada para la fase anterior del análisis. A partir de este punto se procede como en el caso anterior, a entrenar el modelo y validarlo. Los resultados obtenidos se resumen a continuación. Por un lado, haciendo uso de la figura de mérito MSE versus k haciendo uso del conjunto de diseño (Figura 30) seleccionamos un $k = 6$.

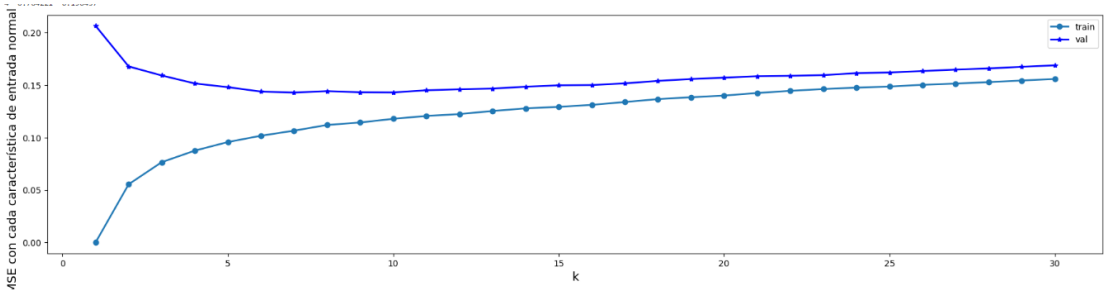


Figura 30:Figura de mérito de MSE versus k haciendo uso del conjunto de diseño para explicar el arousal y la valencia con todas las características.

Los resultados al evaluar el conjunto de test y su comparativa con los obtenidos utilizando las características escaladas de la base de datos se resumen en la Tabla 3. En los resultados obtenidos, los modelos entrenados con las características extraídas utilizando la librería Librosa han mostrado un rendimiento ligeramente inferior en comparación con los modelos basados en la base de datos original.

Tabla 3: Resumen de resultados de los distintos modelos al evaluar el conjunto de test con distintos conjuntos de características

Modelo	Valence y arousal con todas las características (base de datos)	Valence y arousal con todas las características (librosa)
MSE	Arousal = 0.084 Valence = 0.118	Arousal = 0.105 Valence = 0.127
RMSE	Arousal = 0.291 Valence = 0.344	Arousal = 0.324 Valence = 0.356
MAE	Arousal = 0.219 Valence = 0.257	Arousal = 0.240 Valence = 0.281

Estos resultados indican que, si bien las características extraídas con *Librosa* son eficaces y permiten el entrenamiento de modelos con un buen desempeño, los valores de MSE, RMSE y MAE son algo más altos en comparación con los obtenidos utilizando las características originales de la base de datos. Esto sugiere que, en este caso particular, las características originales proporcionan un ajuste más preciso al modelo, aunque la diferencia no es drástica.

Al examinar el gráfico de dispersión (Figura 27) que ilustra la relación entre los valores reales y estimados del *arousal*

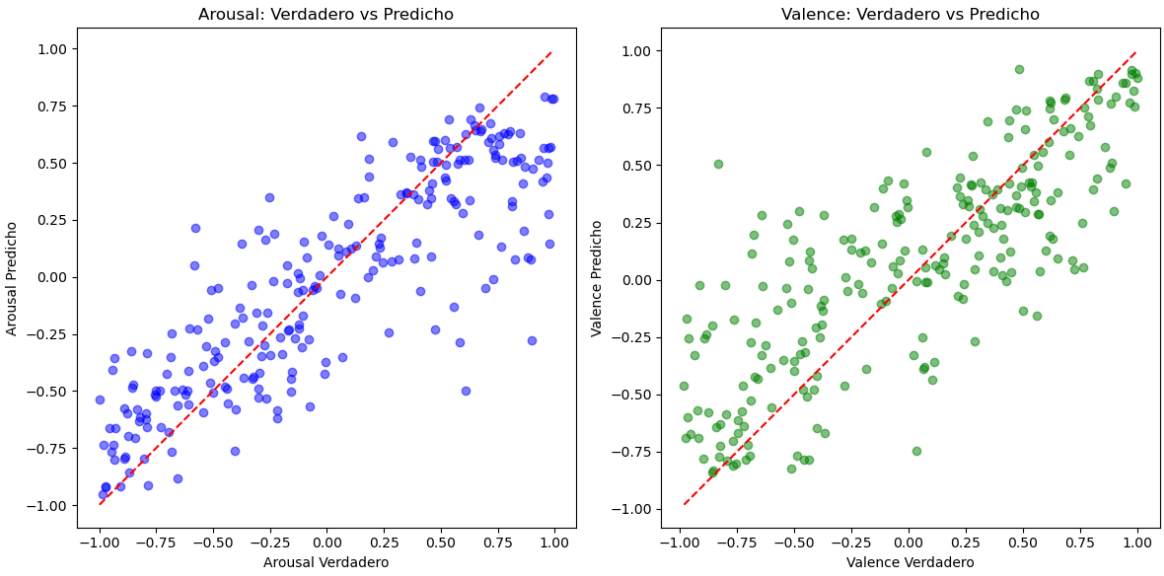


Figura 31: Gráfico de Dispersión: Arousal Real vs Estimado y Valence Real vs. Estimado

Además de los resultados numéricos, los gráficos de dispersión de las predicciones también muestran un ajuste ligeramente peor cuando se utilizan las características extraídas con *Librosa*, en comparación con los gráficos basados en las características originales de la base de datos. En los gráficos de dispersión, la distribución de los puntos alrededor de la línea de predicción ideal (donde los valores reales son iguales a los predichos) es algo menos precisa con las características extraídas de *Librosa*, lo que refuerza la observación de que los resultados obtenidos con la base de datos original ofrecen un mejor ajuste.

Es posible que este comportamiento se deba que ciertas características de la base de datos original contienen información que no ha sido capturada de la misma manera en la extracción con *Librosa*.

8. Conclusiones

A lo largo de este trabajo, se ha desarrollado un modelo predictivo eficaz que relaciona los paisajes sonoros con las emociones percibidas en términos de activación (arousal) y valencia. Utilizando técnicas de aprendizaje automático, en particular el método de k-vecinos más cercanos (k-NN), se ha demostrado la viabilidad de caracterizar sonidos dentro de un paisaje sonoro y predecir las emociones que suscitan. El modelo ha permitido establecer una relación clara entre las características acústicas de los sonidos y las dimensiones emocionales, cumpliendo el objetivo de poder predecir emociones a partir de paisajes sonoros.

Además, se ha evaluado el esquema predictor tanto para una salida conjunta de arousal y valencia como para modelos separados. Los resultados indican que el enfoque con salidas conjuntas es más eficaz. A su vez, se seleccionado el hiperparámetro *k* del modelo mediante un proceso de búsqueda en malla, logrando un ajuste que maximiza las prestaciones del modelo.

La calidad del modelo ha sido evaluada comparando las predicciones con los datos reales, utilizando validación cruzada. Esta evaluación ha confirmado que el modelo tiene una buena capacidad de generalización y puede predecir emociones con bastante precisión en la mayoría de los casos.

Por otro lado, se ha explorado la extracción automática de características utilizando la librería *Librosa*. Aunque el rendimiento de las características extraídas automáticamente fue ligeramente inferior al de las características de la base de datos, este enfoque representa un paso prometedor hacia la automatización del proceso.

Este análisis puede ser aplicado en diversas áreas: la capacidad de predecir las emociones asociadas a diferentes paisajes sonoros permite diseñar entornos sonoros más saludables y enriquecedores. Esto tiene implicaciones en la arquitectura acústica de espacios públicos, urbanos y naturales. Además, contribuye a una mayor comprensión de cómo los sonidos en nuestro entorno influyen en nuestra experiencia cotidiana y puede ayudar a identificar y mitigar sonidos que generan emociones negativas, promoviendo bienestar emocional. Los resultados pueden ser utilizados en terapias de sonido y programas de bienestar que buscan mejorar la salud mental a través de la manipulación de paisajes sonoros.

Finalmente, durante el desarrollo de este proyecto, se han fortalecido diversas habilidades. La capacidad para resolver problemas ha sido clave, permitiendo abordar de manera eficaz los desafíos surgidos durante el proceso, como la selección de características óptimas y el ajuste del modelo. Además, se ha potenciado la adaptabilidad frente a cambios y nuevas técnicas. Todas estas habilidades son un recurso inestimable que llevaré conmigo para mi desarrollo futuro.

9. Bibliografía y webgrafía

Bakker, I., Van Der Voordt, T., Vink, P., y De Boon, J. (2014). Pleasure, arousal, dominance: Mehrabian and Russell revisited. *Current Psychology*, *33*, 405-421.

Barkana, B. D., y Saricicek, I. (2010, April). Environmental noise source classification using neural networks. In *2010 seventh international conference on information technology: new generations* (pp. 259-263). IEEE.

Berglund, B., Nilsson, M. y Axelsson, O. (2007) "Soundscape Psychophysics in Place," in International Congress and Exhibition on Noise Control Engineering, pp. 3704–3712.

Bruce, N. S., y Davies, W. J. (2014). The effects of expectation on the perception of soundscapes. *Applied acoustics*, *85*, 1-11.

Davies, W. J., Bruce, N. S., y Murphy, J. E. (2014). Soundscape reproduction and synthesis. *Acta Acustica United with Acustica*, *100*(2), 285-292.

Ekman, I. (2008). Psychologically motivated techniques for emotional sound in computer games. *Proc. AudioMostly*, 20-26.

Fan, J. (2020). Advances in soundscape and music emotion recognition. https://summit.sfu.ca/_flysystem/fedora/2022-08/input_data/21346/etd20932.pdf

Fan, J., Thorogood, M., y Pasquier, P. (2017, October). Emo-soundscapes: A dataset for soundscape emotion recognition. In *2017 Seventh international conference on affective computing and intelligent interaction (ACII)* (pp. 196-201). IEEE

ISO, I. (2014). 12913-1: 2014: Acoustics—Soundscape Part 1: Definition and Conceptual Framework. ISO: Geneva, Switzerland.

Kallinen, K., y Ravaja, N. (2006). Emotion perceived and emotion felt: Same and different. *Musicae Scientiae*, *10*(2), 191-213.

Lartillot, O., Toiviainen, P., y Eerola, T. (2008). A matlab toolbox for music information retrieval. In *Data Analysis, Machine Learning and Applications: Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation eV, Albert-Ludwigs-Universität Freiburg, March 7–9, 2007* (pp. 261-268). Springer Berlin Heidelberg.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning. New York: Springer. https://link.springer.com/book/10.1007/978-0-387-84858-7.

Lundén, P., y Hurtig, M. (2016, August). On urban soundscape mapping: A computer can predict the outcome of soundscape assessments. In INTER-NOISE and NOISE-CON Congress and Conference Proceedings (Vol. 253, No. 6, pp. 2017-2024). Institute of Noise Control Engineering.

Mathieu, B., Essid, S., Fillon, T., Prado, J., y Richard, G. (2010, August). YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In *ISMIR* (Vol. 2010, pp. 441-446).

Payne, S. R., Davies, W. J., y Adams, M. D. (2009). Research into the practical and policy applications of soundscape concepts and techniques in urban areas. (NANR 200). Department for Environment Food and Rural Affairs: London, UK.

Russell, J. A. (1980). A circumplex model of affect. *Journal of personality and social psychology*, *39*(6), 1161.

Schafer, R. M. (1993). *The soundscape: Our sonic environment and the tuning of the world*. Simon and Schuster.

Timbers, T., Campbell, T., Lee, M., Ostblom, J., & Heagy, L. Data Science: A First Introduction with Python. https://python.datasciencebook.ca/

Wold, E., Blum, T., Keislar, D., y Wheaten, J. (1996). Content-based classification, search, and retrieval of audio. *IEEE multimedia*, *3*(3), 27-36.

ANEXO 1:*Códigos en Python para la generación de los gráficos del apartado Metodología*

```
#importación de las librerías necesarias
import pandas as pd
import altair as alt
# Lectura de fichero de datos de ejemplo
datos = pd.read_csv("data_prueba.csv",sep=';')
datos
#Tratamiento de los datos
datos = pd.DataFrame(datos)

# Separar las columnas numéricas de las categóricas
numeric_columns = datos.select_dtypes(include=['int64', 'float64']).columns
categorical_columns = datos.select_dtypes(include=['object', 'category']).columns

# Estandarizar las columnas numéricas
scaler = StandardScaler()
datos[numeric_columns] = scaler.fit_transform(datos[numeric_columns])

# Mostrar el dataframe estandarizado
print(datos)
#representación del grafico de dispersión
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(
    data=datos,
    x='X1',
    y='X2',
    hue='Class',
    palette={'A': 'mediumseagreen', 'B': 'steelblue'},
    alpha=0.6
)

# Ajustar la relación de aspecto de la figura para que sea cuadrada
plt.gca().set_aspect('equal', adjustable='box')

# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")

# Mostrar el gráfico
plt.show()
#representación del grafico de dispersión y un punto sin clasificar
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(
    data=datos,
    x='X1',
    y='X2',
    hue='Class',
    palette={'A': 'mediumseagreen', 'B': 'steelblue'},
    alpha=0.6
)

# Ajustar la relación de aspecto de la figura para que sea cuadrada
plt.gca().set_aspect('equal', adjustable='box')
```

```
# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")
plt.scatter(2, 4, color='red', marker='D', s=50, label='Nuevo Punto')

# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")

# Mostrar el gráfico
plt.show()
#representación del grafico de dispersión y un punto sin clasificar

plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(
   data=datos,
   x='X1',
   y='X2',
   hue='Class',
   palette={'A': 'mediumseagreen', 'B': 'steelblue'},
   alpha=0.6
)

# Ajustar la relación de aspecto de la figura para que sea cuadrada
plt.gca().set_aspect('equal', adjustable='box')

# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")
plt.scatter(-0.2, 2.5, color='red', marker='D', s=50, label='Nuevo Punto')

# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")

# Mostrar el gráfico
plt.show()
#representación del grafico de dispersión y un punto sin clasificar y un círculo con la distancia

from matplotlib.patches import Circle
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(
   data=datos,
   x='X1',
   y='X2',
   hue='Class',
   palette={'A': 'mediumseagreen', 'B': 'steelblue'},
   alpha=0.6
)

# Ajustar la relación de aspecto de la figura para que sea cuadrada
plt.gca().set_aspect('equal', adjustable='box')

# Configurar etiquetas y título
```

```
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")
plt.scatter(-0.2, 2.5, color='red', marker='D', s=50, label='Nuevo Punto')

# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")

# Agregar un círculo alrededor del punto
circle = Circle((-0.2, 2.5), radius=0.84, edgecolor='black', facecolor='none', linewidth=1)
plt.gca().add_patch(circle)

# Configurar etiquetas y título
scatter_plot.set_xlabel("X1", fontsize=12)
scatter_plot.set_ylabel("X2", fontsize=12)
scatter_plot.legend(title="Clase")
# Mostrar el gráfico
plt.show()

#representación del grafico de dispersión en 3D, un punto sin clasificar
#el gráfico se pude ver desde distintos ángulos


import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Importar para gráficos 3D
from ipywidgets import interact, IntSlider

def plot_3d(elev, azim):
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Asignar colores a las categorías
    colors = {'A': 'mediumseagreen', 'B': 'steelblue'}
    datos['Color'] = datos['Class'].map(colors)

    # Graficar puntos en 3D
    ax.scatter(datos['X1'], datos['X2'], datos['X3'], c=datos['Color'], s=70, alpha=0.6)

    # Añadir un punto adicional en coordenadas (0, 3.5, 1) con forma de diamante
    ax.scatter(0, 3.5, 1, color='red', marker='D', s=100, label='Nuevo Punto')

    # Etiquetas de ejes
    ax.set_xlabel('X1')
    ax.set_ylabel('X2')
    ax.set_zlabel('X3')

    # Título del gráfico
    plt.title('Gráfico en 3D con X1, X2, X3 y Categorías')

    ax.legend(loc='upper right')

    # Ajustar la vista del gráfico en 3D
    ax.view_init(elev=elev, azim=azim)

    # Mostrar el gráfico
```

```
    plt.show()

# Crear controles deslizantes para ángulo de elevación y azimut
interact(plot_3d, elev=IntSlider(min=0, max=90, step=5, value=30), azim=IntSlider(min=0, max=360, step=5, value=45))

#representación del grafico de dispersión en 3D, un punto sin clasificar y lineas a los cuatro puntos mas cercanos
#el gráfico se pude ver desde distintos angulos


import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Importar para gráficos 3D
from ipywidgets import interact, IntSlider
from scipy.spatial import KDTree

# Coordenadas del nuevo punto
new_point = (0, 3.5, 1)

# Función para graficar en 3D con interactividad
def plot_3d(elev, azim):
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Asignar colores a las categorías
    colors = {'A': 'mediumseagreen', 'B': 'steelblue'}
    datos['Color'] = datos['Class'].map(colors)

    # Graficar puntos en 3D
    ax.scatter(datos['X1'], datos['X2'], datos['X3'], c=datos['Color'], s=70, alpha=0.4)

    # Añadir un punto adicional en coordenadas (0, 3.5, 1) con forma de diamante
    ax.scatter(new_point[0], new_point[1], new_point[2], color='red', marker='D', s=100, label='Nuevo Punto')

    # Etiquetas de ejes
    ax.set_xlabel('X1')
    ax.set_ylabel('X2')
    ax.set_zlabel('X3')

    # Título del gráfico
    #plt.title('Gráfico en 3D con X1, X2, X3 y Categorías')

    # Ajustar la vista del gráfico en 3D
    ax.view_init(elev=elev, azim=azim)

    # Encontrar los 5 puntos más cercanos al nuevo punto
    tree = KDTree(datos[['X1', 'X2', 'X3']])
    distances, indices = tree.query([new_point], k=5)

    # Graficar líneas desde el nuevo punto a los 5 más cercanos
    for idx in indices[0]:
        point = datos.loc[idx, ['X1', 'X2', 'X3']]
        ax.plot([new_point[0], point['X1']], [new_point[1], point['X2']], [new_point[2], point['X3']], color='black', linestyle='-')

    # Mostrar el gráfico
```

```
    plt.show()

# Crear controles deslizantes para ángulo de elevación y azimut
interact(plot_3d, elev=IntSlider(min=0, max=90, step=5, value=30), azim=IntSlider(min=0, max=360,
step=5, value=45))

#Código para visualizar las areas que de clasificación según el valor de k


import pandas as pd
import numpy as np
import altair as alt
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer

# Crear un pipeline de k-NN
k-NN_pipeline = make_pipeline(
    StandardScaler(),
    KNeighborsClassifier(n_neighbors=20)
)

# Ajustar el modelo a los datos
X = datos[['X1', 'X2']]
y = datos['Class']
knn_pipeline.fit(X, y)

# Crear la malla de valores para X1 y X2 y organizarlos en un DataFrame
are_grid = np.linspace(
    datos["X1"].min() * 0.95, datos["X1"].max() * 1.05, 50
)
smo_grid = np.linspace(
    datos["X2"].min() * 0.95, datos["X2"].max() * 1.05, 50
)
asgrid = np.array(np.meshgrid(are_grid, smo_grid)).reshape(2, -1).T
asgrid = pd.DataFrame(asgrid, columns=["X1", "X2"])

# Usar el pipeline para hacer predicciones en los puntos de la malla
knnPredGrid = knn_pipeline.predict(asgrid)

# Añadir las predicciones como una nueva columna en los puntos de la malla
prediction_table = asgrid.copy()
prediction_table["Class"] = knnPredGrid

# Crear el gráfico original de datos
unscaled_plot = alt.Chart(datos).mark_point(
    opacity=0.6,
    filled=True,
    size=40
).encode(
    x=alt.X("X1").scale(
        nice=False,
        domain=(
            datos["X1"].min() * 0.95,
            datos["X1"].max() * 1.05
        )
    ),
```

```
    y=alt.Y("X2").scale(
        nice=False,
        domain=(
            datos["X2"].min() * 0.95,
            datos["X2"].max() * 1.05
        )
    ),
    color=alt.Color("Class:N", scale=alt.Scale(range=['#1f77b4', '#2ca02c'])).title("Clase")
)

# Crear el gráfico de predicciones en los puntos de la malla
prediction_plot = alt.Chart(prediction_table).mark_point(
    opacity=0.05,
    filled=True,
    size=300
).encode(
    x="X1",
    y="X2",
    color=alt.Color("Class:N", scale=alt.Scale(range=['#1f77b4', '#2ca02c'])).title("Clase")
)

# Mostrar el gráfico combinado
final_plot = unscaled_plot + prediction_plot
final_plot
```

ANEXO 2: Codigos en Python de los algoritmos de modelización K-NN

Importamos toda las librerias necesiaras

```
import numpy as np
import pandas as pd
# importar todas las funciones de pylab
from pylab import *
import matplotlib.pyplot as plt

from six import StringIO
from IPython.display import Image

from mpl_toolkits import mplot3d
import numpy as np

from sklearn import model_selection
from sklearn import preprocessing

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

import joblib
```

Generamos el código para unir en un ficherlo los dos ficheros de "Arousal" y "Valence"

```
Arousal = pd.read_csv('Arousal.csv',header=None)
Arousal.columns=['Sonido', 'Arousal']
#print(Arousal)
Valence = pd.read_csv('Valence.csv',header=None)
Valence.columns=['Sonido', 'Valence']

Arousal_Valence = pd.merge(Arousal, Valence,on='Sonido', how='inner')
#print(Arousal_Valence.head())

# Guardar el DataFrame combinado en un nuevo archivo CSV
Arousal_Valence.to_csv('Arousal_Valence.csv', index=False)
print("Se ha guardado el archivo combinado como 'Arousal_Valence.csv.csv'")
Se ha guardado el archivo combinado como 'Arousal_Valence.csv.csv'
```

Leemos el fichero de características

```
Features = pd.read_csv('Features.csv')
#print(Features.head())
#Combinamos todo en un fichero unico

#Renombramos la primera columna
Features.rename(columns={Features.columns[0]: 'Sonido'}, inplace=True)
```

```
#Eliminamos la primera columna con los nombres de los ficheros de sonido

# Realizar el merge, manteniendo solo las filas comunes entre los tres archivos
#comunes_AB = pd.merge(A, B, on='sonido',
how='inner')  # Mantener solo los nombres comunes entre A y B
Arousal_Valence_Features = pd.merge(Arousal_Valence, Features, on='Sonido', how='inner')
Arousal_Valence_Features.to_csv('Arousal_Valence_Features.csv', index=False)

[232]:
arousal = Arousal_Valence_Features.iloc[:, 1]
valence = Arousal_Valence_Features.iloc[:, 2]
#print (valence.head())

Features_solo = Arousal_Valence_Features.iloc[:, 3:]
#print (Features_solo.head())
```

Normalizamos las características según Min-Max Scaling

```
# Inicializar el scaler
scaler = MinMaxScaler()

# Aplicar el Min-Max Scaling a todas las columnas del DataFrame
Features_scaled = pd.DataFrame(scaler.fit_transform(Features_solo),
columns=Features_solo.columns)

joblib.dump(scaler, 'minmax_scaler.pkl')

# Mostrar el DataFrame normalizado
#print(Features_scaled.head())

['minmax_scaler.pkl']

# Inicializar el StandardScaler para Z-Score Standardization
scaler = StandardScaler()

# Aplicar Z-Score Standardization a todas las columnas del DataFrame
Features_Standarized = pd.DataFrame(scaler.fit_transform(Features_solo),
columns=Features_solo.columns)
# Mostrar el DataFrame normalizado
#print(Features_Standarized)
#Comprobación de que los datos están estandarizados
print(Features_Standarized['brightness_mean'].mean())
print(Features_Standarized['brightness_mean'].std())
4.686184572201815e-17
1.000412456194069
```

Creación de gráficos de dispersión

```
#x = data['Arousal'] # Asignar el valor de Arousal a x
#y = data['Valence'] # Asignar el valor de Valence a y

# Crear el scatter plot de Arousal vs Valence
plt.figure(figsize=(8, 6))  # Tamaño opcional para el gráfico
plt.scatter(arousal, valence, alpha=0.5)  # alpha establece la transparencia de los puntos

# Etiquetas y título
plt.xlabel('Arousal')
```

```
plt.ylabel('Valence')
plt.title('Scatter Plot de Arousal vs. Valence')
Text(0.5, 1.0, 'Scatter Plot de Arousal vs. Valence')

#Seleccion de caracteristicas para hacer los graficos en 3D
loudness_mean = Features_scaled['loudness_mean']
brightness_mean = Features_scaled['brightness_mean']
decreaseslope_mean = Features_scaled['decreaseslope_mean']
chromagram_mean_3 = Features_scaled['chromagram_mean_3']
perceptual_sharp_mean = Features_scaled['perceptual_sharp_mean']
mfcc_mean_13 = Features_scaled['mfcc_mean_13']

#Scatter plot de arousal frente a loudness_mean
plt.figure(figsize=(8, 6))  # Tamaño opcional para el gráfico
plt.scatter(valence,loudness_mean,alpha=0.5)

# Etiquetas y título
plt.xlabel('Valence')
plt.ylabel('loudness_mean')
plt.title('Scatter Plot de Arousal vs. loudness_mean')
plt.show()
```

Construimos gráficos en 3D con Arousal, Valencia y algunas características el código de el scater plot en 3D

```
# Representaciones gráficas

plt.figure(figsize=(22,5))

plt.subplot(1,2,1)
plt.hist(arousal,bins=15,edgecolor='black',alpha=0.5)
plt.xlabel('Arousal',fontsize=14)

plt.subplot(1,2,2)
plt.hist(valence,bins=15,edgecolor='black',alpha=0.5)
plt.xlabel('Valence',fontsize=14)

plt.show()



# Representaciones gráficas scater plot e histograma

plt.figure(figsize=(22,5))

plt.subplot(1,2,1)
plt.scatter(valence,loudness_mean,alpha=0.5)

# Etiquetas y título
plt.xlabel('Valence')
plt.ylabel('loudness_mean')
plt.title('Scatter Plot de Valence vs. Loudness_mean')


plt.subplot(1,2,2)
plt.hist(loudness_mean,bins=15,edgecolor='black',alpha=0.5)
```

```
plt.xlabel('loudness_mean',fontsize=14)


plt.show()

# Representaciones gráficas 3D

# Crear una figura grande
fig = plt.figure(figsize=(22, 5))

# Crear el primer subplot con un gráfico 3D
ax = fig.add_subplot(1, 3, 1, projection='3d')
scatter = ax.scatter3D(arousal, valence, loudness_mean, c=loudness_mean, cmap='Reds',
linewidth=0.2)
# Etiquetas de ejes
ax.set_xlabel('Arousal')
ax.set_ylabel('Valence')
ax.set_zlabel('loudness_mean')
# Título del gráfico
ax.set_title('Arousal, Valence, loudness_mean')

# Crear el segundo subplot con un gráfico 3D
ax = fig.add_subplot(1, 3, 2, projection='3d')
scatter = ax.scatter3D(arousal, valence, brightness_mean,c=brightness_mean, cmap='Reds',
linewidth=0.2)
# Etiquetas de ejes
ax.set_xlabel('Arousal')
ax.set_ylabel('Valence')
ax.set_zlabel('Brightness_mean')
# Título del gráfico
ax.set_title('Arousal, Valence, Brightness_mean')


# Crear el tercero subplot con un gráfico 3D
ax = fig.add_subplot(1, 3, 3, projection='3d')
scatter = ax.scatter3D(arousal, valence, chromagram_mean_3, c=chromagram_mean_3, cmap='Reds',
linewidth=0.2)
# Etiquetas de ejes
ax.set_xlabel('Arousal')
ax.set_ylabel('Valence')
ax.set_zlabel('chromagram_mean_3')
# Título del gráfico
ax.set_title('Arousal, Valence, chromagram_mean_3')
# Ajustar el diseño para evitar solapamientos
plt.tight_layout()

# Representaciones gráficas 3D

# Crear una figura grande
fig = plt.figure(figsize=(22, 5))

# Crear el primer subplot con un gráfico 3D
ax = fig.add_subplot(1, 3, 1, projection='3d')
scatter = ax.scatter3D(arousal, valence, decreaseslope_mean, c=decreaseslope_mean, cmap='Reds',
linewidth=0.2)
# Etiquetas de ejes
ax.set_xlabel('Arousal')
ax.set_ylabel('Valence')
```

```
ax.set_zlabel('decreaseslope_mean')
# Título del gráfico
ax.set_title('Arousal, Valence, decreaseslope_mean')

# Crear el segundo subplot con un gráfico 3D
ax = fig.add_subplot(1, 3, 2, projection='3d')
scatter = ax.scatter3D(arousal, valence, perceptual_sharp_mean,
c=perceptual_sharp_mean, cmap='Reds', linewidth=0.2)
# Etiquetas de ejes
ax.set_xlabel('Arousal')
ax.set_ylabel('Valence')
ax.set_zlabel('perceptual_sharp_mean')
# Título del gráfico
ax.set_title('Arousal, Valence, perceptual_sharp_mean')


# Crear el tercero subplot con un gráfico 3D
ax = fig.add_subplot(1, 3, 3, projection='3d')
scatter = ax.scatter3D(arousal, valence, mfcc_mean_13, c=mfcc_mean_13, cmap='Reds', linewidth=0.2)
# Etiquetas de ejes
ax.set_xlabel('Arousal')
ax.set_ylabel('Valence')
ax.set_zlabel('mfcc_mean_13')
# Título del gráfico
ax.set_title('Arousal, Valence, mfcc_mean_13')
# Ajustar el diseño para evitar solapamientos
plt.tight_layout()
```

Estudio de las correlaciones

```
#Eliminamos la primera columna de fichero Arousal_Valence_Features
Arousal_Valence_Features_sin_nombre = Arousal_Valence_Features.iloc[:, 1:]


correlation_all = Arousal_Valence_Features_sin_nombre.corr()
#print(correlation_all.head())


Estudio de las correlaciones
#Eliminamos la primera columna de fichero Arousal_Valence_Features
Arousal_Valence_Features_sin_nombre = Arousal_Valence_Features.iloc[:, 1:]


correlation_all = Arousal_Valence_Features_sin_nombre.corr()
#print(correlation_all.head())
#Vamos a ver los valores con correlaciones altas
# Ajustar las opciones para mostrar toda la salida
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_seq_items', None)

# Crear una máscara para ocultar los valores menores o iguales a 0.8
mask = np.triu(np.ones_like(correlation_all, dtype=bool), k=1)
```

```
# Filtrar los valores de la matriz de correlación que son mayores que 0.8
high_correlation_pairs = correlation_all.where(mask & (np.abs(correlation_all) > 0.9)).stack()

# Mostrar los pares de columnas con correlación mayor que 0.8
print("\nPares de columnas con correlación mayor que 0.9:")
print(high_correlation_pairs)

Correlacion_feat = Arousal_Valence_Features_sin_nombre.corr()

sns.heatmap(Correlacion_feat, annot=True, cmap ='coolwarm',annot_kws={"size":
6}, xticklabels=Correlacion_feat.columns, yticklabels=Correlacion_feat.columns)

plt.xticks(fontsize=3)
plt.yticks(fontsize=3)
plt.show()

correlation
= Arousal_Valence_Features_sin_nombre['Valence'].corr(Arousal_Valence_Features_sin_nombre['loudn
ess_mean'])
print(correlation)

-0.6173935841576578
```

K-nn unidimensional

```
#cargamos las librerias necesarias para el calculo del Knn en sklearn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
#Separamos el conjunto de diseño (X_train, y_train) y de test (X_test, y_test): particiones 80/20
X_train, X_test, y_train, y_test = train_test_split(Features_scaled['loudness_mean'], valence, test_size =
0.2, random_state=2)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)

#Selección del hiper-parámetro haciendo uso del conjunto de diseño.
El criterio de elección es mínimo MSE

param_grid = {'n_neighbors': range(1,41)}
k_grid = np.array(range(1,41))
grid_knn = GridSearchCV(KNeighborsRegressor(), param_grid = param_grid, scoring =
'neg_mean_squared_error', cv=3, return_train_score=True)
grid_knn.fit(X_train.values.reshape(-1,1), y_train.values.reshape(-1,1))

mse_train = -1*np.array(grid_knn.cv_results_['mean_train_score'])
mse_val = -1*np.array(grid_knn.cv_results_['mean_test_score'])


plt.figure(figsize=(22,5))
plt.plot(k_grid,mse_train,'-o',label="train", linewidth=2)
plt.plot(k_grid,mse_val,'-*b',label="val", linewidth=2)
plt.xlabel('k',fontsize=14)
plt.ylabel('MSE',fontsize=14)
plt.legend()
```

```
plt.show()

print("Menor MSE en validación: {:.3f}".format(-grid_knn.best_score_))
print("Valor correspondiente para el hiperparámetro: {}".format(grid_knn.best_params_))

# Con el valor de k seleccionado u otro que se vea que funciona bien
(a ojo), construimos el modelo con el conjunto de diseño y evaluamos las prestaciones en test
n_neighbors = 8
modelo_knn_regres = KNeighborsRegressor(n_neighbors)
y_test_estimada = modelo_knn_regres.fit(X_train.values.reshape(-1,1), y_train.values.reshape(-
1,1)).predict(X_test.values.reshape(-1,1)) # Entrenamos y evaluamos en test

MSE_test_knn_1D = mean_squared_error(y_test.values.reshape(-1,1), y_test_estimada)
MAE_test_knn_1D = mean_absolute_error(y_test.values.reshape(-1,1), y_test_estimada)

print("MSE en el conjunto de test cuando se utiliza sólo la característica loudness_mean en k-
nn para predecir valencia")
print(MSE_test_knn_1D)

print("MAE en el conjunto de test cuando se utiliza sólo la característica loudness_mean en k-
nn para predecir valencia")
print(MAE_test_knn_1D)
```

Comparativa con distintos valores del hiperparametro k

```
T = np.linspace(0,1)[:, np.newaxis]
# generamos 600 puntos en el intervalo entre 0 y 200
y_ = modelo_knn_regres.predict(T)


# Representamos las estimaciones sobre el subconjunto de diseño
plt.figure
#plt.figure(figsize=(22,5))
plt.scatter(X_train.values.reshape(-1,1), y_train.values.reshape(-1,1), alpha=0.2) #,
label='observaciones')
plt.plot(T, y_,'g', label="estimacion")

plt.xlabel('loudness_mean',fontsize=14)
plt.ylabel('valence',fontsize=14)

#plt.xlabel('x',fontsize=14)
#plt.ylabel('y',fontsize=14)
plt.axis('tight')
plt.legend()
plt.title("KNeighborsRegressor (k = %i)" % (n_neighbors))
plt.show()

n_neighbors = 1
#n_neighbors = 600
modelo_knn_regres = KNeighborsRegressor(n_neighbors)
y_T_k1 = modelo_knn_regres.fit(X_train.values.reshape(-1,1), y_train.values.reshape(-1,1)).predict(T)
# Entrenamos y evaluamos en la malla 1D

#plt.figure(figsize=(22,5))
plt.scatter(X_train.values.reshape(-1,1), y_train.values.reshape(-1,1), alpha=0.2,
label='observaciones')
plt.plot(T, y_T_k1, c='g', label='estimacion')
```

```
plt.xlabel('loudness_mean',fontsize=14)
plt.ylabel('valence',fontsize=14)
plt.title("KNeighborsRegressor (k = 1)")
plt.axis('tight')
plt.legend()
plt.show()

n_neighbors = 80

modelo_knn_regres = KNeighborsRegressor(n_neighbors)
y_T_k1 = modelo_knn_regres.fit(X_train.values.reshape(-1,1), y_train.values.reshape(-1,1)).predict(T)
# Entrenamos y evaluamos en la malla 1D

#plt.figure(figsize=(22,5))
plt.scatter(X_train.values.reshape(-1,1), y_train.values.reshape(-1,1), alpha=0.2,
label='observaciones')
plt.plot(T, y_T_k1, c='g', label='estimacion')
plt.xlabel('loudness_mean',fontsize=14)
plt.ylabel('valence',fontsize=14)
plt.title("KNeighborsRegressor (k = 80)")
plt.axis('tight')
plt.legend()
plt.show()
```

K-nn multivariante

Estimación de la Valencia utilizando todas las características

X_all = Features_scaled

```
Y_all = valence
print(X_all.shape)
print(Y_all.shape)

X_all_train, X_all_test, y_all_train, y_all_test = train_test_split(X_all, Y_all, test_size =
0.2, random_state=2)
print(X_all_train.shape)
print(y_all_train.shape)
print(X_all_test.shape)
print(y_all_test.shape)

param_grid = {'n_neighbors': range(1,31,1)}
k_grid = np.array(range(1,31,1))
grid_knn = GridSearchCV(KNeighborsRegressor(), param_grid = param_grid, scoring =
'neg_mean_squared_error', cv=3, return_train_score=True)
grid_knn.fit(X_all_train, y_all_train)

mse_knn_all_train = -1*np.array(grid_knn.cv_results_['mean_train_score'])
mse_knn_all_val = -1*np.array(grid_knn.cv_results_['mean_test_score'])

plt.figure(figsize=(22,5))
plt.plot(k_grid,mse_knn_all_train,'-o',label="train", linewidth=2)
plt.plot(k_grid,mse_knn_all_val,'-*b',label="val", linewidth=2)
plt.xlabel('k',fontsize=14)
plt.ylabel('MSE con todas las características',fontsize=14)
plt.legend()
plt.show()
```

```
print("Menor MSE en validación: {:.3f}".format(-grid_knn.best_score_))
print("Valor correspondiente para el hiperparámetro: {}".format(grid_knn.best_params_))


n_neighbors = 5
modelo_knn_all_regres = KNeighborsRegressor(n_neighbors)
y_test_knn_all_estimada = modelo_knn_all_regres.fit(X_all_train, y_all_train).predict(X_all_test)
# Entrenamos y evaluamos en test

MSE_test_knn_all = mean_squared_error(y_all_test.values.reshape(-1,1), y_test_knn_all_estimada)
MAE_test_knn_all = mean_absolute_error(y_all_test.values.reshape(-1,1), y_test_knn_all_estimada)

print("Número de vecinos considerados:")
print(n_neighbors)

print("MSE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-nn:")
print(MSE_test_knn_all)

print("MAE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-
nn:")
print(MAE_test_knn_all)


Menor MSE en validación: 0.144
Valor correspondiente para el hiperparámetro: {'n_neighbors': 8}
Número de vecinos considerados:
5

# Gráfico de dispersión entre los valores reales y estimados
plt.figure(figsize=(8, 6))
plt.scatter(y_all_test, y_test_knn_all_estimada, alpha=0.5, label="Valores estimados")
plt.plot([min(y_all_test), max(y_all_test)], [min(y_all_test), max(y_all_test)], color='red',
label="Valores reales", linestyle='--')

# Etiquetas de los ejes
plt.xlabel("Valence Real", fontsize=14)
plt.ylabel("Valence Estimado", fontsize=14)

# Título y leyenda
plt.title("Gráfico de dispersión: Valence Real vs Estimado", fontsize=16)


# Mostrar el gráfico
plt.show()
```

Estimación del Arousal utilizando todas las características

```
X_all = Features_scaled
Y_all = Arousal_Valence_Features[['Arousal']]
print(X_all.shape)
print(Y_all.shape)

X_all_train, X_all_test, y_all_train, y_all_test = train_test_split(X_all, Y_all, test_size =
0.2, random_state=2)
print(X_all_train.shape)
print(y_all_train.shape)
print(X_all_test.shape)
```

```
print(y_all_test.shape)

param_grid = {'n_neighbors': range(1,31,1)}
k_grid = np.array(range(1,31,1))
grid_knn = GridSearchCV(KNeighborsRegressor(), param_grid = param_grid, scoring =
'neg_mean_squared_error', cv=3, return_train_score=True)
grid_knn.fit(X_all_train, y_all_train)

mse_knn_all_train = -1*np.array(grid_knn.cv_results_['mean_train_score'])
mse_knn_all_val = -1*np.array(grid_knn.cv_results_['mean_test_score'])

plt.figure(figsize=(22,5))
plt.plot(k_grid,mse_knn_all_train,'-o',label="train", linewidth=2)
plt.plot(k_grid,mse_knn_all_val,'-*b',label="val", linewidth=2)
plt.xlabel('k',fontsize=14)
plt.ylabel('MSE con todas las características',fontsize=14)
plt.legend()
plt.show()

print("Menor MSE en validación: {:.3f}".format(-grid_knn.best_score_))
print("Valor correspondiente para el hiperparámetro: {}".format(grid_knn.best_params_))


n_neighbors = 4
modelo_knn_all_regres = KNeighborsRegressor(n_neighbors)
y_test_knn_all_estimada = modelo_knn_all_regres.fit(X_all_train, y_all_train).predict(X_all_test)
# Entrenamos y evaluamos en test



MSE_test_knn_all = mean_squared_error(y_all_test.values.reshape(-1,1), y_test_knn_all_estimada_pred)
MAE_test_knn_all = mean_absolute_error(y_all_test.values.reshape(-
1,1), y_test_knn_all_estimada_pred)

print("Número de vecinos considerados:")
print(n_neighbors)

print("MSE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-nn:")
print(MSE_test_knn_all)

print("MAE en el conjunto de test cuando se utilizan todas las características (normalizadas) en k-
nn:")
print(MAE_test_knn_all)


# Gráfico de dispersión entre los valores reales y estimados
plt.figure(figsize=(8, 6))
plt.scatter(y_all_test, y_test_knn_all_estimada, alpha=0.5, label="Valores estimados")
plt.plot([min(y_all_test['Arousal']), max(y_all_test['Arousal'])],
      [min(y_all_test['Arousal']), max(y_all_test['Arousal'])],
      color='red', label="Valores reales", linestyle='--')

# Etiquetas de los ejes
plt.xlabel("Arousal Real", fontsize=14)
plt.ylabel("Arousal Estimado", fontsize=14)

# Título y leyenda
plt.title("Gráfico de dispersión: Arousal Real vs Estimado", fontsize=16)
```

```
# Mostrar el gráfico
plt.show()
```

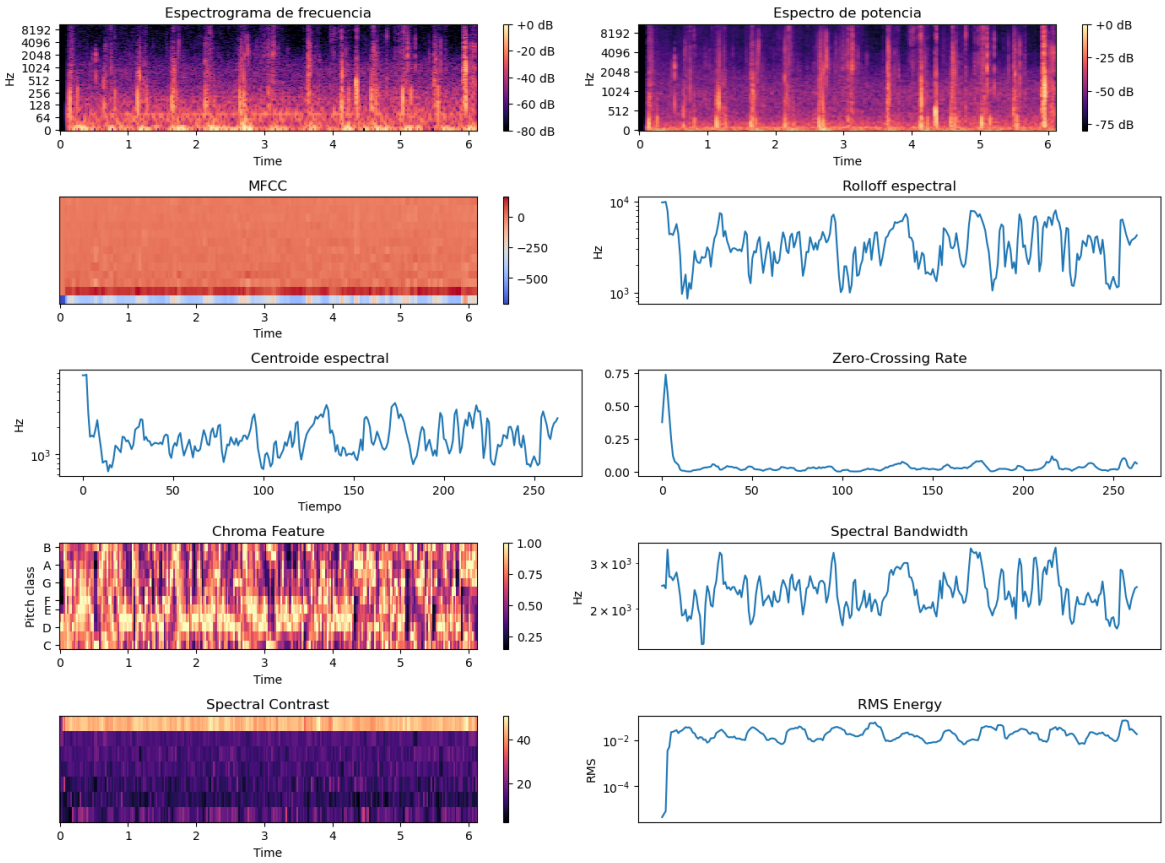
Estimación de valencia y arousal frente a todas las caracteristicas

```
X_all = Features_scaled
y_all = Arousal_Valence_Features[['Arousal', 'Valence']]


# Dividimos los datos en conjuntos de entrenamiento y prueba
X_all_train, X_all_test, y_all_train, y_all_test = train_test_split(X_all, y_all, test_size=0.2, random_state=2)


# Definir la malla de hiperparámetros para k-NN
param_grid = {'n_neighbors': range(1, 31, 1)}
k_grid = np.array(range(1, 31, 1))

# Búsqueda en la malla utilizando validación cruzada
grid_knn = GridSearchCV(KNeighborsRegressor(), param_grid=param_grid,
scoring='neg_mean_squared_error', cv=3, return_train_score=True)
grid_knn.fit(X_all_train, y_all_train)

mse_knn_all_train_std = -1 * np.array(grid_knn.cv_results_['mean_train_score'])
mse_knn_all_val_std = -1 * np.array(grid_knn.cv_results_['mean_test_score'])

# Gráfico de MSE en función del número de vecinos
plt.figure(figsize=(22, 5))
plt.plot(k_grid, mse_knn_all_train_std, '-o', label="train", linewidth=2)
plt.plot(k_grid, mse_knn_all_val_std, '-*b', label="val", linewidth=2)
plt.xlabel('k', fontsize=14)
plt.ylabel('MSE con cada característica de entrada normalizada', fontsize=14)
plt.legend()
plt.show()

print("Menor MSE en validación: {:.3f}".format(-grid_knn.best_score_))
print("Valor correspondiente para el hiperparámetro: {}".format(grid_knn.best_params_))

# Entrenamiento y evaluación en el conjunto de prueba
#n_neighbors = grid_knn.best_params_['n_neighbors']

n_neighbors = 5
modelo_knn_all_std_regres = KNeighborsRegressor(n_neighbors)
y_test_knn_all_std_estimada = modelo_knn_all_std_regres.fit(X_all_train, y_all_train).predict(X_all_test)



# Calcular las métricas de evaluación para cada variable por separado
MSE_test_knn_arousal = mean_squared_error(y_all_test['Arousal'], y_test_knn_all_std_estimada[:, 0])
MAE_test_knn_arousal = mean_absolute_error(y_all_test['Arousal'], y_test_knn_all_std_estimada[:, 0])
MSE_test_knn_valence = mean_squared_error(y_all_test['Valence'], y_test_knn_all_std_estimada[:, 1])
MAE_test_knn_valence = mean_absolute_error(y_all_test['Valence'], y_test_knn_all_std_estimada[:, 1])


print("Número de vecinos considerados:")
print(n_neighbors)
```

```
print("MSE en el conjunto de test para 'valence':")
print(MSE_test_knn_valence)
print("MAE en el conjunto de test para 'valence':")
print(MAE_test_knn_valence)

print("MSE en el conjunto de test para 'arousal':")
print(MSE_test_knn_arousal)
print("MAE en el conjunto de test para 'arousal':")
print(MAE_test_knn_arousal)


from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score  # Asegúrate de importar r2_score

X_all = Features_scaled
y_all = Arousal_Valence_Features[['Arousal', 'Valence']]

# Dividir los datos en conjuntos de entrenamiento y prueba
X_all_train, X_all_test, y_all_train, y_all_test = train_test_split(X_all, y_all, test_size=0.2, random_state=2)

# Definir la malla de hiperparámetros para k-NN
param_grid = {'n_neighbors': range(1, 31, 1)}
k_grid = np.array(range(1, 31, 1))

# Búsqueda en la malla utilizando validación cruzada
grid_knn = GridSearchCV(KNeighborsRegressor(metric='l2'), param_grid=param_grid,
scoring='neg_mean_squared_error', cv=3, return_train_score=True)
grid_knn.fit(X_all_train, y_all_train)

# Calcular MSE para el conjunto de entrenamiento y validación
mse_knn_all_train_std = -1 * np.array(grid_knn.cv_results_['mean_train_score'])
mse_knn_all_val_std = -1 * np.array(grid_knn.cv_results_['mean_test_score'])

# Gráfico de MSE en función del número de vecinos
plt.figure(figsize=(22, 5))
plt.plot(k_grid, mse_knn_all_train_std, '-o', label="train", linewidth=2)
plt.plot(k_grid, mse_knn_all_val_std, '-*b', label="val", linewidth=2)
plt.xlabel('k', fontsize=14)
plt.ylabel('MSE con cada característica de entrada normalizada', fontsize=14)
plt.legend()
plt.show()

print("Menor MSE en validación: {:.3f}".format(-grid_knn.best_score_))
print("Valor correspondiente para el hiperparámetro: {}".format(grid_knn.best_params_))

# Entrenamiento y evaluación en el conjunto de prueba
n_neighbors = grid_knn.best_params_['n_neighbors']  # Obtener el mejor número de vecinos
modelo_knn_all_std_regres = KNeighborsRegressor(n_neighbors=n_neighbors, weights='distance',
metric='cosine')

# Ajustar el modelo con los datos de entrenamiento y predecir con los de prueba
y_test_knn_all_std_estimada = modelo_knn_all_std_regres.fit(X_all_train, y_all_train)
y_test_knn_all_std_estimada_predict = y_test_knn_all_std_estimada.predict(X_all_test)

# Calcular las métricas de evaluación para cada variable por separado
MSE_test_knn_arousal = mean_squared_error(y_all_test['Arousal'], y_test_knn_all_std_estimada_predict
[:, 0])
```

```
MAE_test_knn_arousal = mean_absolute_error(y_all_test['Arousal'], y_test_knn_all_std_estimada_predic
t[:, 0])
MSE_test_knn_valence = mean_squared_error(y_all_test['Valence'], y_test_knn_all_std_estimada_predict
[:, 1])
MAE_test_knn_valence = mean_absolute_error(y_all_test['Valence'], y_test_knn_all_std_estimada_predic
t[:, 1])

# Calcular R^2 para las predicciones
r2_arousal = r2_score(y_all_test['Arousal'], y_test_knn_all_std_estimada_predict[:, 0])
r2_valence = r2_score(y_all_test['Valence'], y_test_knn_all_std_estimada_predict[:, 1])

# Imprimir los resultados
print("Número de vecinos considerados:")
print(n_neighbors)

print("MSE en el conjunto de test para 'valence':")
print(MSE_test_knn_valence)
print("MAE en el conjunto de test para 'valence':")
print(MAE_test_knn_valence)

print("MSE en el conjunto de test para 'arousal':")
print(MSE_test_knn_arousal)
print("MAE en el conjunto de test para 'arousal':")
print(MAE_test_knn_arousal)

# Imprimir precisión
print("Precisión del modelo (R^2) para 'Arousal': {:.3f}".format(r2_arousal))
print("Precisión del modelo (R^2) para 'Valence': {:.3f}".format(r2_valence))


#Se guarda el modelo entrenado para utilizarlo despues para predecir
joblib.dump(y_test_knn_all_std_estimada, 'modelo_entrenado.pkl')


# Crear gráficos de dispersión
plt.figure(figsize=(12, 6))

# Gráfico de dispersión para Arousal
plt.subplot(1, 2, 1)
plt.scatter(y_all_test['Arousal'], y_test_knn_all_std_estimada[:, 0], color='blue', alpha=0.5)
plt.plot([min(y_all_test['Arousal']), max(y_all_test['Arousal'])], [min(y_all_test['Arousal']),
max(y_all_test['Arousal'])], 'r--')  # Línea de referencia
plt.xlabel('Arousal Verdadero')
plt.ylabel('Arousal Predicho')
plt.title('Arousal: Verdadero vs Predicho')

# Gráfico de dispersión para Valence
plt.subplot(1, 2, 2)
plt.scatter(y_all_test['Valence'], y_test_knn_all_std_estimada[:, 1], color='green', alpha=0.5)
plt.plot([min(y_all_test['Valence']), max(y_all_test['Valence'])], [min(y_all_test['Valence']),
max(y_all_test['Valence'])], 'r--')  # Línea de referencia
plt.xlabel('Valence Verdadero')
plt.ylabel('Valence Predicho')
plt.title('Valence: Verdadero vs Predicho')

# Mostrar los gráficos
plt.tight_layout()
plt.show()
```

ANEXO 3: Ejemplo de extracción de características



ANEXO 4: Resumen estadístico de la base de datos de características

	Media	Mediana	Desv. estándar	Varianza	Curtosis	Coef.de asimetría
rms_mean	0.047	0.024	0.077	0.006	47.448	5.519
rms_std	0.063	0.035	0.082	0.007	22.538	3.528
decreaseslope_mean	0.167	0.138	0.107	0.011	8.216	2.437
fluctuation_max	0.063	0.046	0.052	0.003	100.799	7.146
eventdensity_mean	0.601	0.678	0.355	0.126	-1.566	-0.238
zerocross_mean	0.134	0.107	0.111	0.012	14.477	2.958
zerocross_std	0.250	0.237	0.078	0.006	15.274	2.159
rolloff_mean	0.296	0.281	0.169	0.029	1.732	1.106
rolloff_std	0.172	0.143	0.119	0.014	4.390	1.601
brightness_mean	0.435	0.427	0.195	0.038	-0.212	0.208
brightness_std	0.318	0.299	0.138	0.019	1.667	0.767
centroid_mean	0.207	0.189	0.127	0.016	5.623	1.730
centroid_std	0.179	0.114	0.168	0.028	1.693	1.470
spread_mean	0.342	0.340	0.170	0.029	0.518	0.736
spread_std	0.189	0.169	0.117	0.014	6.119	1.858
skewness_mean	0.311	0.275	0.137	0.019	1.666	0.978
skewness_std	0.186	0.151	0.125	0.016	6.668	2.053
kurtosis_mean	0.110	0.068	0.107	0.011	8.931	2.399
kurtosis_std	0.076	0.045	0.095	0.009	23.254	3.976
flatness_mean	0.235	0.210	0.150	0.023	1.951	1.111
flatness_std	0.206	0.185	0.144	0.021	3.192	1.472
entropy_mean	0.654	0.673	0.151	0.023	2.324	-1.143
entropy_std	0.257	0.224	0.151	0.023	1.753	1.245
mfcc_mean_1	0.561	0.558	0.111	0.012	1.241	-0.223
mfcc_mean_2	0.621	0.623	0.095	0.009	2.490	-0.224
mfcc_mean_3	0.521	0.520	0.121	0.015	0.874	-0.011
mfcc_mean_4	0.529	0.541	0.124	0.015	0.445	-0.228
mfcc_mean_5	0.466	0.466	0.103	0.011	1.702	-0.101
mfcc_mean_6	0.500	0.501	0.113	0.013	1.975	0.028
mfcc_mean_7	0.539	0.540	0.089	0.008	3.193	-0.267
mfcc_mean_8	0.657	0.664	0.098	0.010	2.870	-0.560
mfcc_mean_9	0.568	0.573	0.084	0.007	3.865	-0.358
mfcc_mean_10	0.433	0.438	0.081	0.007	5.384	-0.402
mfcc_mean_11	0.409	0.416	0.079	0.006	5.451	0.494
mfcc_mean_12	0.527	0.538	0.076	0.006	5.237	-0.701
mfcc_mean_13	0.524	0.530	0.092	0.008	4.609	-0.830
mfcc_std_1	0.226	0.205	0.113	0.013	5.329	1.563
mfcc_std_2	0.200	0.169	0.144	0.021	7.463	2.214
mfcc_std_3	0.184	0.177	0.101	0.010	9.187	1.973

mfcc_std_4	0.209	0.191	0.115	0.013	7.023	1.930
mfcc_std_5	0.220	0.195	0.134	0.018	3.427	1.361
mfcc_std_6	0.280	0.254	0.126	0.016	4.137	1.477
mfcc_std_7	0.261	0.233	0.105	0.011	7.796	2.169
mfcc_std_8	0.279	0.260	0.100	0.010	7.942	2.364
mfcc_std_9	0.264	0.236	0.101	0.010	7.246	2.030
mfcc_std_10	0.240	0.216	0.098	0.010	12.637	2.588
mfcc_std_11	0.177	0.158	0.071	0.005	20.130	2.941
mfcc_std_12	0.190	0.172	0.071	0.005	26.533	3.576
mfcc_std_13	0.240	0.224	0.079	0.006	18.162	2.867
flux_mean	0.048	0.024	0.076	0.006	52.690	5.873
flux_std	0.074	0.034	0.099	0.010	16.167	3.191
lowenergy	0.627	0.606	0.116	0.013	-0.025	0.133
novelty_mean	0.309	0.288	0.146	0.021	0.318	0.622
novelty_std	0.361	0.349	0.188	0.035	-0.309	0.328
pitch_mean	0.215	0.208	0.097	0.009	5.546	1.204
pitch_std	0.449	0.461	0.196	0.038	-0.157	-0.479
chromagram_center_mean_1	0.547	0.560	0.125	0.016	0.805	-0.524
chromagram_center_mean_2	0.493	0.506	0.123	0.015	3.393	-0.840
chromagram_center_mean_3	0.420	0.427	0.104	0.011	3.638	0.125
chromagram_center_mean_4	0.329	0.325	0.095	0.009	3.691	0.710
chromagram_center_mean_5	0.394	0.350	0.153	0.023	3.660	1.757
chromagram_center_mean_6	0.318	0.294	0.114	0.013	4.368	1.559
chromagram_center_mean_7	0.376	0.361	0.103	0.011	5.829	1.542
chromagram_center_mean_8	0.360	0.353	0.115	0.013	2.126	0.539
chromagram_center_mean_9	0.469	0.454	0.122	0.015	1.489	0.344
chromagram_center_mean_10	0.447	0.437	0.115	0.013	2.590	0.711
chromagram_center_mean_11	0.567	0.592	0.158	0.025	0.434	-0.486
chromagram_center_mean_12	0.595	0.609	0.131	0.017	0.678	-0.313
chromagram_center_std_1	0.726	0.769	0.151	0.023	4.201	-1.876
chromagram_center_std_2	0.713	0.756	0.152	0.023	5.060	-2.085
chromagram_center_std_3	0.724	0.777	0.167	0.028	3.799	-1.862
chromagram_center_std_4	0.716	0.759	0.159	0.025	4.065	-1.865
chromagram_center_std_5	0.722	0.763	0.155	0.024	4.275	-1.841
chromagram_center_std_6	0.678	0.726	0.157	0.025	2.982	-1.581
chromagram_center_std_7	0.706	0.750	0.169	0.029	2.574	-1.573
chromagram_center_std_8	0.666	0.709	0.180	0.033	1.840	-1.433
chromagram_center_std_9	0.452	0.472	0.121	0.015	1.402	-0.835
chromagram_center_std_10	0.727	0.773	0.157	0.025	2.480	-1.505
chromagram_center_std_11	0.715	0.748	0.138	0.019	4.833	-1.868
chromagram_center_std_12	0.722	0.756	0.131	0.017	5.368	-1.894
key_mean	0.542	0.540	0.102	0.010	3.236	0.568

key_std	0.757	0.786	0.109	0.012	10.155	-2.812
mode_mean	0.441	0.434	0.099	0.010	12.925	2.340
mode_std	0.726	0.740	0.099	0.010	11.046	-2.351
hcdf_mean	0.447	0.447	0.095	0.009	4.385	-0.038
hcdf_std	0.295	0.262	0.120	0.014	5.897	1.957
inharmonicity_mean	0.841	0.885	0.121	0.015	5.767	-2.037
inharmonicity_std	0.222	0.183	0.144	0.021	1.717	1.170
chromagram_mean_1	0.630	0.673	0.166	0.027	1.155	-1.139
chromagram_mean_3	0.558	0.597	0.155	0.024	1.487	-1.048
chromagram_mean_4	0.527	0.562	0.151	0.023	1.259	-0.990
chromagram_mean_5	0.594	0.615	0.169	0.028	0.934	-0.468
chromagram_mean_6	0.493	0.512	0.139	0.019	1.234	-0.446
chromagram_mean_7	0.474	0.487	0.152	0.023	1.130	-0.454
chromagram_mean_8	0.465	0.485	0.152	0.023	0.971	-0.659
chromagram_mean_9	0.496	0.513	0.162	0.026	0.880	-0.383
chromagram_mean_10	0.613	0.644	0.140	0.019	2.104	-1.110
chromagram_mean_11	0.673	0.724	0.173	0.030	2.296	-1.418
chromagram_mean_12	0.668	0.708	0.160	0.026	2.160	-1.336
chromagram_std_1	0.576	0.557	0.139	0.019	0.939	-0.140
chromagram_std_2	0.599	0.584	0.133	0.018	2.079	-0.400
chromagram_std_3	0.520	0.504	0.120	0.014	2.023	-0.256
chromagram_std_4	0.554	0.543	0.113	0.013	3.714	-0.649
chromagram_std_5	0.557	0.539	0.129	0.017	1.500	-0.238
chromagram_std_6	0.547	0.527	0.128	0.016	1.724	-0.190
chromagram_std_7	0.491	0.474	0.120	0.014	1.512	-0.165
chromagram_std_8	0.569	0.552	0.150	0.022	1.153	0.125
chromagram_std_9	0.517	0.504	0.133	0.018	1.078	-0.148
chromagram_std_10	0.564	0.538	0.143	0.021	0.679	0.228
chromagram_std_11	0.601	0.581	0.152	0.023	0.834	-0.085
chromagram_std_12	0.548	0.536	0.131	0.017	0.824	-0.080
loudness_mean	0.157	0.133	0.105	0.011	9.246	2.226
loudness_std	0.213	0.178	0.147	0.022	3.658	1.591
energy_mean	0.047	0.024	0.077	0.006	47.453	5.519
energy_std	0.064	0.035	0.082	0.007	22.237	3.516
perceptual_sharp_mean	0.287	0.281	0.098	0.010	6.255	1.351
perceptual_sharp_std	0.218	0.208	0.080	0.006	11.161	1.642
spectral_slope_mean	0.211	0.198	0.110	0.012	7.165	1.864
spectral_slope_std	0.246	0.231	0.080	0.006	17.408	2.338
spectral_var_mean	0.420	0.435	0.110	0.012	2.504	-0.839
spectral_var_std	0.227	0.211	0.097	0.009	3.302	1.016