UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS DE LA DOCUMENTACIÓN

Máster en Data Science, Big Data & Business Analytics

Trabajo Fin de MÁSTER

¿Qué elementos hacen que una canción sea popular?

Un análisis de la data de Spotify

Alumno: Lucía Durbán Redondo
Tutores: Santiago Mota y Carlos Ortega

Madrid, 19 de Febrero del 2026

Agradecimientos

A mi familia, mis padres y mi hermana, por siempre ayudarme y animarme a hacer lo que de verdad me gusta, por apoyarme en cada paso de mi carrera profesional.

A Nacho, por estar ahí en el día a día, en los altos y los bajos, por ayudarme en todo lo puedes, por creer en mí y animarme cuando no encuentro las fuerzas.

A mis abuelos, que aunque ya no estén sé que estarían orgullosos de mí y de la persona en la que me he convertido.

A mis amigos, en especial a Ana y Marina, por aguantarme cuando solo puedo quejarme y ayudar a animarme como mejor se les da, escuchándome, dándome su opinión en las cosas que no tengo claras y con un buen matcha.

Gracias a todos.

Abstracto

Este trabajo investiga cuáles son los factores del sonido más importantes que hacen que una canción sea más popular que otras, basándose en datos recopilados desde 2010 hasta julio de 2025. Esta investigación se amplía aún más analizando si los elementos externos, reconocidos como "contexto", moderan la relación entre estas variables importantes y la popularidad. Para responder a la pregunta principal del trabajo, se han utilizando datos extraídos la API web de Spotify, teniendo como predictores una serie de variables basadas en el audio, y como respuesta la variable "popularidad". Debido a que la API de Spotify lleva meses cerrada y sin la posibilidad de utilizarla, y que estos datos ya estaban disponibles online gracias a Anna's Archive, fueron descargados con la ayuda de un torrent.

Se utiliza un LightGBM para evaluar cuáles de las variables son más importantes para calcular qué hace a una canción más popular que otra. Un primer análisis reveló que las dos variables más importantes eran la *loudness* y el *explicit*. Un segundo análisis más concreto realizado con LightGBM que incluyó valores de SHAP. Esto demostró que el *loudness, explicit* y *speechiness* son importantes y que su efecto se potencia con este modelo. También realizaremos un modelo de machine learning para ayudarnos a predecir qué canciones serán populares, dependiendo de las características de sonido de la canción.

Índice de contenidos

1. Introducción....4
1.1. Problema de investigación....4
1.2. Motivación empresarial....5
1.3. Motivación académica....5
2. Data....6
2.1. Recopilación de la data....6
2.2. Descripción de la data....7
3. Metodología....9
3.1. Análisis simple....9
3.2. Métodos Candidatos....11
3.3. Tabla resúmen de los métodos....14
3.4. Selección del modelo....14
4. Resultados....15
4.1. Análisis principal....15
4.2. Modelo predictor....16
5. Conclusión....17
5.1. Resumen de la investigación....17
5.2. Interpretación de los resultados....17
5.3. Implicaciones directivas....18
5.4. Limitaciones y futuras líneas de investigación....19
6. Bibliografía....20
6. Anexos....21
6.1. Anexo 1....21
6.2. Anexo 2....25
6.3. Anexo 3....27
6.4. Anexo 4....30

1. Introducción

1.1. Problema de investigación

En nuestra generación y las anteriores, siempre ha estado presente la música, y estamos expuestos a ella constantemente en nuestro día a día. Según un estudio realizado en 26 países con personas entre 16-64 años, escuchamos 20,7 horas de música grabada a la semana (IFPI, 2023), y se podría argumentar que más si tenemos en cuenta la música de fondo. Esta gran presencia de la música en nuestras vidas se puede asociar a la relación que existe entre el acto de escuchar música y la dopamina generada cuando lo hacemos. La evidencia causal de que la transmisión dopaminérgica modula tanto los componentes hedónicos (agrado) como motivacionales (deseo) de la recompensa musical en los seres humanos, así como sugiere que la dopamina desempeña un papel fundamental en la recompensa por estímulos abstractos (música), posiblemente a través de interacciones con procesos cognitivos (expectativa, memoria, aprendizaje) que conforman el placer musical (Ferreri et al., 2019).

Los ingresos por música grabada han crecido de forma continua durante una década, y sigue habiendo un gran potencial de crecimiento y desarrollo, a través de la innovación, las tecnologías emergentes y la inversión tanto en artistas como en las partes en evolución del creciente ecosistema musical mundial. En 2024, los ingresos totales en la industria musical fueron de 29.6 billones USD, siendo un crecimiento de 4,8% respecto al año anterior. Aunque sí se aprecian cambios en la forma de consumir la música, ya que los servicios de streaming tuvieron un crecimiento del 9,5% y las ventas físicas tuvieron un descenso del 3,1%, respecto al año anterior. Esto no quiere decir que dejemos de escuchar música, sino que la escuchamos de forma diferente, en el mayor caso en plataformas de streaming como Spotify, Apple Music o Amazon Music (IFPI, 2025).

Sabemos que para la industria musical y sus componentes, es importante saber qué hace a una canción popular, ya sea para recrearlo o entender qué es lo que hace que las masas quieran escuchar ciertas canciones por encima de otras. Para saber si qué hace a una canción un “hit”, el profesor Jonah Berger se centra en el “tú”, The Power of “You”, conectando nuestras experiencias personales a la canción, haciendo que esta nos parezca más identificable y emocionalmente resonante. Es decir, para los consumidores es importante lo que dice la canción, tanto como el ritmo o el tempo de esta, sentirse identificado con lo que está diciendo es gran parte de su popularidad (Berger, 2021).

En este trabajo, nos enfocaremos en cuales son los factores que más contribuyen para el éxito de una canción. Para ello, analizaremos las características de audio de las pistas para conocer sus características, extraídas a través de la API web de Spotify, gracias a la extracción por parte del grupo de Anna's Archive (2025), que extrajeron todos los datos de las canciones desde 2010 a Julio de 2025 y han sido descargado con la ayuda de un torrent. Refiriéndose a todos los elementos que componen una canción, los que están relacionados con el estado de ánimo de la canción (danceability, valence, energy, tempo, key), con sus

propiedades (loudness, speechiness, instrumentalness), y con su contexto (acousticness, liveness) (Spotify for Developers, 2025). Profundizaremos en el análisis para comprender cómo el contexto afecta a los predictores de popularidad y en qué medida los oyentes se ven influidos por factores externos. La variable dependiente será la variable "popularidad" proporcionada por el conjunto de datos, que en última instancia muestra el grado de popularidad de una canción en comparación con otras.

1.2. Motivación empresarial

Poder determinar qué características hacen que una canción alcance mayor popularidad que otra es una cuestión fuertemente impulsada por la propia industria musical (Borg & Hokkanen, 2011). El rendimiento de un tema puede influir de forma notable en los ingresos de sellos discográficos, plataformas de streaming y otros participantes del sector. Cuando se entienden los elementos que condicionan el éxito, los responsables de marketing y los ejecutivos pueden decidir con mayor criterio qué lanzamientos impulsar, qué estrategias de promoción aplicar y cómo distribuir recursos como campañas publicitarias o apoyo a giras. Esa misma información también sirve para detectar tendencias que están surgiendo, descubrir nuevos talentos y estimar con mayor precisión qué canciones tienen más posibilidades de destacar en el futuro. Para las grandes compañías y para quienes invierten en música, es clave mantener una actitud innovadora, ofrecer propuestas frescas y adaptarse con rapidez a los cambios del mercado. Además, analizar los factores asociados al éxito puede revelar oportunidades de mejora en productos y servicios. Por ejemplo, ciertos géneros o estilos conectan mejor con grupos demográficos específicos o con determinadas regiones, lo que permite ajustar la oferta y la comunicación para responder de manera más efectiva a las preferencias del público objetivo.

Actualmente, la industria musical depende en gran medida de las plataformas de streaming como Spotify, Apple Music o Amazon Music. Estas plataformas emplean sistemas algorítmicos para sugerir contenido musical a sus usuarios basándose en su comportamiento de escucha y gustos personales. Cuando los responsables del sector comprenden qué elementos determinan la popularidad de un tema, pueden ofrecer a estas plataformas datos más precisos sobre qué canciones tienen mayor potencial de éxito entre audiencias específicas o en zonas geográficas concretas.

1.3. Motivación académica

Entender el por qué ciertas anatomías de las canciones es interesante para los académicos para poder ayudar a recrearlas y saber las características que hacen a una canción un "hit". Muchos académicos se han centrado en analizar la anatomía de una canción para tratar de comprender cuáles son los elementos clave que impulsan su éxito, utilizando muchas técnicas y conjuntos de datos diferentes. Sin embargo, este estudio representará una visión más completa del tema.

Terroso-Saenz, Soto y Muñoz (2022) examinan los temas más escuchados en Spotify en 52 países, analizando atributos como bailabilidad, positividad e intensidad, lo cual les permite observar cómo dichas

características reflejan tendencias anímicas y cómo interactúan con factores contextuales que pueden influir en los hábitos de escucha de la población. También desarrollaron un modelo de series temporales multivariante para anticipar las preferencias musicales en esos países, hallando modificaciones significativas en dichos patrones como consecuencia de la pandemia.

En este trabajo se analizan las canciones con más de 55 puntos de popularidad, incluyendo 15 años de datos y todos los países que están presentes en la data de Spotify, así como todas las variables que tiene una canción dentro de esta misma data.

2. Data

2.1. Recopilación de la data

Para este trabajo he decidido trabajar con la API de Spotify, por el volúmen de datos que tienen a nuestra disposición y por la precisión de éstos. En mi caso, cuando fuí a extraer los datos con Spotify for Developers, el sistema estaba caído y no dejaban scrapear la data. Encontrándome este problema, decidí indagar en el tema y encontré que el grupo anónimo "Anna's Archive" había scrapeado toda la base de datos de Spotify hace poco y habían abierto al público la descarga de esos datos (Anna's Archive, 2025).

Spotify es la plataforma líder de streaming musical del mundo, con más de 670 millones de usuarios activos mensuales, un 40% de estos siendo usuarios de pago en el cuarto trimestre de 2024. La plataforma cuenta con más de 200 millones de canciones disponibles, teniendo que ser sumado otras formas de entretenimiento en audio, como los podcasts. Todo esto hace que Spotify tuviera unos ingresos de más de 15 billones de euros en 2024, superando con creces a sus principales competidores Tencent, Apple Music y Amazon, teniendo una cuota de mercado basada en el número de suscriptores del 32% (Statista, 2025).

Gracias al trabajo de Anna's Archive conseguí descargarme, por medio de torrent, todos los metadatos de todas las canciones en Spotify desde 2010 a julio de 2025, un total de 256.039.007 canciones/pistas. De los archivos descargados, dos nos son útiles para este trabajo, "spotify_clean.sqlite3" y "spotify_clean_audio_features.sqlite3", el primero siendo la información principal como nombre de canción, ID de canción, nombre del/los artista/s, ID del/los artista/s, y la popularidad, y el segundo teniendo las variables que hacen la canción como bailabilidad, valencia, y tempo. Cabe recalcar que para este trabajo estaré usando los nombres de las variables en inglés, igual que se usarán en la base de datos, así como en el código realizado.

Estas bases de datos originales han sido limpiadas y unidas por medio de SQL, ya que trabajar y compartir una base de datos de más de 256 millones de pistas sería complicado, así que se ha decidido acortar esas bases de datos a las 100.000 canciones más populares en ellas. También se han juntado ambos datasets, dado que cada uno de ellos contienen diferentes tipos de datos que nos interesan para este análisis, por una parte la

información general de una canción, y por otra las características acústicas de esta, acabando con un total de 98.960 canciones con todas las variables necesarias para nuestro análisis en un mismo dataset.

2.2. Descripción de la data

La base de datos se compone de 13 variables y 7 identificadores por canción. A continuación podemos ver la *Tabla 1* con una descripción detallada de cada variable, y la *Tabla 2*, donde encontramos estadísticas descriptivas de las variables numéricas.

	Nombre	**Tipo**	**Descripción**
Identificadores	*track_id*	text	Identificador único de canción
	track_name	text	Nombre de la canción
	artist_name	text	Nombre del artista o artistas
	artist_id	text	Identificador único del artista o artistas
	isrc	text	Código Internacional Normalizado de Grabación
	album_name	text	Nombre del álbum
	album_type	text	Tipo de álbum, puede ser álbum o single
Variables basadas en audio	*explicit*	integer	Indica contenidos como lenguaje soez, referencias sexuales, violencia o referencias a drogas que se consideren inapropiadas para los niños. Valores entre 0 y 1
	key	integer	La clave en la que se encuentra la pista. Valores de 0 a 11
	mode	integer	Indica la modalidad (mayor o menor) de una pista. Mayor es 1 y menor es 0
	danceability	numeric	Descripción de la idoneidad de una pista para bailar, basada en la combinación de muchos elementos, incluyendo el tempo y la estabilidad del ritmo. Valores de 0,0 a 1,0
	loudness	numeric	Volumen general en decibelios (dB) de una canción. Valores de -60 a 0
	speechiness	numeric	Detecta la presencia de palabras habladas en una pista. Valores de 0,0 a 1,0. Cuanto más se acerque a 1, más probable es que la grabación esté compuesta íntegramente por palabras
	acousticness	numeric	Medida de confianza sobre si la pista es acústica. Valores de 0,0 a 1,0
	energy	numeric	Representación de la medida perceptiva de intensidad y actividad. Valores de 0,0 a 1,0
	valence	numeric	Descriptor de la positividad musical de una grabación. Valores de 0,0 a 1,0. Una valencia más cercana a 1 indica canciones más alegres y felices, mientras que lo contrario

			indica canciones tristes y enfadadas
	tempo	numeric	Tempo estimado de una grabación en pulsaciones por minuto (BPM). Valores de 0 a 230
	instrumentalness	numeric	Predice si una grabación no contiene voces, con valores entre 0,0 y 1,0. Cuanto más se acerque a 1,0, más instrumental será la grabación
	liveness	numeric	Detector de público en la grabación. Cuanto más alto sea el valor, más probable es que la canción se haya grabado en directo. Valores de 0,0 a 1.0
Variable de interés	*popularity*	integer	La popularidad de una canción es un valor entre 0 y 100, siendo 100 la más popular. La popularidad se calcula mediante un algoritmo y se basa, en su mayor parte, en el número total de reproducciones que ha tenido la canción y en la antigüedad de dichas reproducciones. En términos generales, las canciones que se reproducen mucho en la actualidad tendrán una mayor popularidad que las canciones que se reproducían mucho en el pasado.

Tabla 1. Descripción de las variables e identificadores incluidos en el conjunto de datos, basada en la documentación de Spotify (2025).

En esta primera tabla, el primer grupo representa los identificadores de las canciones, nombres o ID's, que son *track_id, track_name, artist_name, artist_id, isrc, album_name* y *album_type.* Los nombres son los identificadores generales de las pistas, debido a que los nombres de canciones, álbumes y artistas se pueden repetir necesitamos los ID's, el más importante siendo el de la canción, ya que el del artista se repite en todas las canciones de este. El segundo grupo representa las variables de audio de las pistas, estas son las principales que se van a analizar en el siguiente punto, ya que lo que queremos ver es cuáles son más importantes y pesadas para la popularidad de una canción, siendo esta, la *popularity*, nuestra variable principal. Un valor numérico entre el 0 y el 100 que mide la popularidad de la pista, siendo calculada gracias a una fórmula interna de Spotify, con las anteriores variables y algunas más como escuchas recientes.

	Minimum	**Maximum**	**Mean**	**Median**	**Std. Dev.**
explicit	0	1	0,21490	0	0,410756
key	0	11	5,279091	5	3,582641
mode	0	1	0,602724074	1	0,489334001
danceability	0	0,998	0,613187917	0,636	0,177239662
loudness	-54,341	1,16	-8,472065796	-6,669	6,715578722
speechiness	0	0,967	0,095177444	0,052	0,102241976
acousticness	0	0,996	0,318096082	0,217	0,304019448
energy	0	1	0,618150761	0,655	0,23364819

valence	0	1	0,502627282	0,504	0,253780287
tempo	0	215,962	121,2382386	120,944	29,36734237
instrumentalness	0	1	0,100262623	0,00000436	0,263016814
liveness	0	0,997	0,189790108	0,124	0,159628129
popularity	55	100	61,93753033	60	5,98333845

Tabla 2. Estadísticas descriptivas de las variables incluidas en el conjunto de datos después de la limpieza.

Viendo la *Tabla 2*, se pueden apreciar números generales de nuestras variables numéricas, teniendo una idea clara de cómo se miden estas variables , cómo las tenemos representadas en nuestros datos y sus promedios, medianas y desviación estándar, ayudándonos a visualizar una imagen general de nuestros datos. Como ejemplo se aprecia que la media y mediana de la *popularity* está alrededor de los 60 puntos, lo que nos quiere decir que aunque tenemos un conjunto de datos de casi 100.000 pìstas con popularidad entre los 55 y 100 puntos, la mayoría de éstas están por la baja de ese rango. Este resultado tiene sentido, ya que solo hay 62 pistas con 90 puntos de popularidad o más, teniendo el corte de popularidad de las 10.000 pistas más escuchadas en 70 puntos.

3. Metodología

3.1. Análisis simple

Para abordar nuestra pregunta central de investigación, primero debemos determinar si las variables de audio presentes en nuestro conjunto de datos mantienen alguna relación con la variable de interés, la popularidad. Antes de avanzar hacia un análisis más elaborado, es importante obtener información preliminar sobre nuestros datos, lo que también nos permitirá verificar si estos hallazgos iniciales coinciden o difieren de lo reportado en estudios previos. Con este propósito, optamos por aplicar una regresión lineal múltiple. Este método estadístico permite evaluar si existe una relación entre múltiples variables independientes y una variable dependiente, constituyendo así una extensión de la regresión lineal simple. La expresión matemática correspondiente es la siguiente:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n + \varepsilon$$

Tenemos la Y, que es la variable dependiente, β_0 la constante, β_n que son los coeficientes de las variables independientes, y x_n, que son las variables independientes. Este modelo tiene como objetivo reducir los errores cuadráticos ajustando una línea lineal a través de los datos, con un proceso denominado OLS (ordinary least squares, mínimos cuadrados ordinarios), asumiendo que la relación entre las variables es lineal (Echambadi & Hess, 2007). Para tener una visión general óptima de todas las variables, incluimos todas ellas para evaluar su relación con la popularidad, en nuestro caso todas las variables numéricas del

anterior punto. La *Tabla 3* muestra los resultados de la fórmula denominada, siendo *popularity* la variable dependiente (Anexo 2):

	Coeficiente Error	**Error Estándar**
const	63,769***	(0,198)
explicit	0,857***	(0,052)
key	0,001	(0,005)
mode	0,066*	(0,040)
danceability	-0,513***	(0,148)
loudness	0,026***	(0,006)
speechiness	-3,792***	(0,204)
energy	-0,457***	(0,153)
valence	-0,264***	(0,096)
acousticness	-0,874***	(0,090)
tempo	-0,001**	(0,001)
instrumentalness	-1,400***	(0,106)
liveness	-0,821***	(0,124)

Tabla 3. Resultados de la línea de regresión.

Los resultados de la regresión lineal múltiple muestran un bajo poder explicativo ($R^2 = 0.014$), lo que indica que las características acústicas analizadas explican solo una pequeña parte de la variabilidad en la popularidad de las canciones.

Vemos algunas perspectivas interesantes, ya que vemos algunas diferencias con las pruebas sin modelo descritas en la sección anterior. Podemos ver en la tabla que los coeficientes más significativos y mayor impacto son *speechiness, acousticness* y *instrumentalness*, todos con efecto negativo, sugiriendo que las canciones más habladas, instrumentales y acústicas tienden a afectar negativamente a la popularidad de las pistas. La mayor diferencia la marca el coeficiente de *speechiness*, ya que un aumento de 1 punto en la capacidad de hablar conduce, en promedio, a una reducción de casi 4 puntos en la popularidad. Además, si una canción es más instrumental y es más acústica, la popularidad se reduce en 1,4 y 0,8 puntos, respectivamente. Las variables de *danceability, energy, valence* y *liveness* tienen coeficientes negativos pero moderados, lo que hace que afectan negativamente a la popularidad pero en menor nivel. Hay dos variables, *key* (0,001) y *tempo* (-0,001) que tienen un efecto casi nulo, estando tan cerca del 0. Las tres variables que afectan positivamente a la popularidad son *explicit, loudness* y *mode*, que aunque no sea un número muy grande, son las únicas con coeficiente positivo, de estas las que afecta más a nuestra variable principal es *explicit*.

La regresión lineal múltiple puede resultar muy útil para realizar un análisis exploratorio inicial, ya que ofrece resultados inmediatos y facilita interpretaciones directas. No obstante, aplicar este método a cualquier

conjunto de datos, y particularmente al que estamos analizando, presenta ciertas limitaciones. En primer lugar, la regresión lineal no gestiona adecuadamente los datos desbalanceados. En nuestro caso, aunque no existen valores ausentes, tenemos un conjunto de datos entre los 55 y 100 puntos de popularidad, lo cual puede perjudicar la capacidad del modelo para identificar con precisión las relaciones entre variables, ya que no tenemos el rango entero entre 0 y 100. Adicionalmente, el conjunto de datos incluye canciones publicadas en distintos años y períodos, cuyas características no son directamente comparables al mismo nivel, algo que la regresión lineal asume por defecto. Por ejemplo, si ocurren cambios significativos en los patrones de consumo musical a lo largo del tiempo, la regresión lineal no los considera en su estimación (Detienne et al., 2003).

En la siguiente sección presentaremos algunos métodos que podrían ser utilizados para responder a nuestras principales preguntas de investigación. Describiremos las ventajas y desventajas de cada uno y, al final, elegiremos el más adecuado para nuestro conjunto de datos y nuestra investigación.

3.2. Métodos Candidatos

ARIMA

ARIMA, Autoregressive Integrated Moving Average o Media Móvil Integrada Autorregresiva, es una familia de modelos que explica el comportamiento de una serie temporal a partir de sus propios valores pasados dados por los retrasos (autorregresivos) y de los errores retrasados (media móvil), considerando además la estacionariedad que se logra mediante diferenciación, operación inversa a la integración. Dicho de otro modo, ARIMA parte del supuesto de que la serie temporal puede describirse mediante las autocorrelaciones presentes en los datos, sin depender directamente de tendencias o patrones estacionales. El modelo está dividido en tres partes. Autoregressive (AR) representa la relación dependiente entre una observación y un número determinado de observaciones retrasadas. Integrated (I) se refiere a hacer que la serie temporal sea estacionaria, es decir, una serie temporal cuyas propiedades estadísticas (media, mediana, etc.) no varían con el tiempo. Y el Moving Average (MA) se refiere al modelo que utiliza la dependencia entre una observación y un error residual de un modelo de media móvil aplicado a observaciones retardadas (Students of PhD at the Asian Institute of Management, 2020).

ARIMA hace un excelente trabajo a la hora de comprender los datos de series temporales y extraer información valiosa sobre las tendencias futuras, lo cual es muy importante para comprender hacia dónde se dirigen las tendencias. Sin embargo, el modelo no es adecuado para nuestra investigación, ya que no identifica cuáles son las variables más importantes.

Balanced Random Forest

Como proponen Chen, Liaw y Breiman (2004) el método tradicional, Random Forest, se puede sufrir la maldición de aprender a partir de un conjunto de datos de entrenamiento extremadamente desequilibrado.

Dado que está diseñado para minimizar la tasa de error global, tenderá a centrarse más en la precisión de la predicción de la clase mayoritaria, lo que a menudo da lugar a una precisión deficiente para la clase minoritaria. Los árboles de decisión aleatorios equilibrados introducen modificaciones en el criterio de división teniendo en cuenta el desequilibrio de clases, con una combinación de sobre muestreo aleatorio de la clase minoritaria en cada nodo de decisión. El método Balanced Random Forest (BRF) es un algoritmo de aprendizaje automático utilizado para tareas de clasificación y regresión, que consiste en un conjunto de árboles de decisión. Los árboles de decisión son estructuras similares a diagramas de flujo en las que los nodos internos representan características, las ramas representan reglas de decisión y los nodos finales representan etiquetas de clase o valores de salida. Los bosques aleatorios mejoran la precisión al combinar muchos árboles de decisión. Cada árbol se entrena con un subconjunto aleatorio de datos y un subconjunto aleatorio de características, y la predicción final en un bosque aleatorio se obtiene agregando las predicciones individuales de todos los árboles de decisión (Ali et al., 2012).

Esta técnica abordará el problema de contar con una clase predominantemente mayor que las demás; en nuestro caso, el elevado número de canciones con un valor de popularidad igual a 55. No obstante, los bosques aleatorios balanceados no son especialmente apropiados para trabajar con datos de series temporales. Esto implica que no capturan de manera explícita las variaciones de tendencia a lo largo del tiempo, aspecto que resulta relevante al analizar canciones publicadas durante un periodo de 15 años. La razón es que este método trata cada observación de forma independiente, sin considerar las posibles correlaciones temporales entre ellas.

LightGBM

LightGBM es una implementación de árboles de decisión potenciados por gradiente (gradient boosted decision trees, GBDT), lo que significa que es un modelo que utiliza aprendices más débiles, en este caso árboles de decisión, para crear un modelo conjunto más fuerte. El framework comienza con un modelo inicial (clasificador débil) que se entrena con los datos. A partir de este, se calculan las diferencias entre los valores observados y las predicciones generadas, que representan los errores que el modelo no logró capturar. El algoritmo de descenso de gradiente se encarga de identificar la mejor dirección para optimizar el modelo, determinando cómo deben ajustarse las predicciones para minimizar dichos errores. En cada iteración se incorporan nuevos clasificadores base, pero ahora son entrenados para predecir el gradiente negativo de la función de pérdida en lugar de los valores originales, buscando así reducir los residuos acumulados. Al final, todas estas predicciones se combinan para conformar el modelo conjunto, el cual se emplea para realizar predicciones sobre datos nuevos. Estas predicciones se ponderan según el desempeño de cada clasificador durante el entrenamiento. Para prevenir el sobreajuste, el algoritmo incorpora técnicas de regularización como la reducción de la tasa de aprendizaje (shrinkage) y el submuestreo de características, mecanismos que controlan la influencia de cada clasificador débil en el modelo final (Ke et al., 2017).

LightGBM pertenece a la familia de algoritmos de impulso por gradiente (gradient boosting), que incluye otros frameworks conocidos como XGBoost y AdaBoost. LightGBM comparte numerosas ventajas con XGBoost, tales como la optimización dispersa, el entrenamiento en paralelo, el soporte de múltiples funciones de pérdida, la regularización, el bagging y la detención anticipada (early stopping). Como mencionan Ke, Meng, Finley y otros (2017), la principal distinción entre ambos métodos reside en la forma de construir los árboles. Mientras que XGBoost crece nivel por nivel (depth-wise), LightGBM utiliza una estrategia de crecimiento hoja por hoja (leaf-wise), seleccionando en cada paso la hoja que, según el algoritmo, generará la mayor reducción en la pérdida. Además, LightGBM no busca el mejor punto de división recorriendo todos los valores ordenados de las características, sino que implementa un algoritmo basado en histogramas altamente optimizado para el aprendizaje de árboles de decisión, lo que aporta importantes ventajas tanto en eficiencia computacional como en uso de memoria.

LightGBM resulta un método apropiado para nuestro análisis por diversas razones. En primer lugar, está diseñado para trabajar eficazmente con datos desbalanceados. El método asigna pesos mayores a las clases minoritarias y menores a las mayoritarias, compensando así cualquier desequilibrio en la distribución. Además, permite implementar bagging balanceado, que ajusta la distribución de clases en cada iteración, y su algoritmo asigna gradientes más elevados a las observaciones mal clasificadas de la clase minoritaria, facilitando que el modelo capture mejor los patrones característicos de dicha clase.

Regresión LASSO

LASSO (Least Absolute Shrinkage and Selection Operator) es una técnica de regresión lineal que realiza la selección de variables y la regularización, y se introdujo originalmente para abordar las limitaciones de la regresión lineal tradicional, específicamente cuando se trata de manejar grandes conjuntos de datos con muchas observaciones. La regresión LASSO minimiza la suma de los errores al cuadrado añadiendo un término de penalización adicional, según la fórmula:

$$\sum_{i=1}^{n} (y_i - \sum_j x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

donde y es el vector de valores observados de la variable dependiente, x es la variable predictora, β es el vector de coeficientes que se va a estimar, λ es el parámetro que regula la fuerza del término de penalización, y $|\beta_j|$ representa la suma de los valores absolutos de los coeficientes. El término de penalización reduce algunos de los coeficientes hacia 0, lo que da como resultado que algunos de los predictores sean exactamente cero y, por lo tanto, se excluyan del modelo. Este proceso se conoce como selección de variables. El término de penalización impone una restricción a la magnitud de los coeficientes. A medida que λ aumenta, más coeficientes se reducen a cero, lo que da lugar a un modelo más simple con menos predicciones (Tibshirani, 1996).

LASSO es un método de fácil interpretación y capaz de procesar grandes volúmenes de datos. Se ajusta a nuestro objetivo porque elimina automáticamente las variables que considera irrelevantes, en este caso aquellas que no contribuyen a aumentar o disminuir la popularidad. Al hacerlo, también puede capturar diferencias temporales y evidenciar si el momento de publicación tiene influencia o no, siempre que se realice un pre procesamiento adecuado de los datos (Tibshirani, 1996). Sin embargo, la limitación de este modelo en nuestro contexto es que no maneja eficientemente los datos desbalanceados. A diferencia de LightGBM y los árboles de decisión balanceados, donde la ponderación de clases se aplica de forma automática, en LASSO tendríamos que implementar manualmente dicha ponderación, lo que añade complejidad y reduce su idoneidad para nuestras necesidades específicas.

3.3. Tabla resúmen de los métodos

La *Tabla 4* enseña un resumen de los métodos recién descritos, estos son adecuados para nuestra principal pregunta y abordan el problema que teníamos con la regresión linear.

Modelo	**Desequilibrio de clases**	**Viabilidad**	**Interpretabilidad**	**Efectos de interacción**
ARIMA	No apto	Requiere preprocesamiento de datos	No apto	Si
Balanced Random Forest	Apto	Muy fácil de implementar y ejecutar	Muy fácil	Si
LightGBM	Apto	Fácil de implementar y ejecutar	Depende de la complejidad del modelo	Si
LASSO	No apto	Más complejo debida al procesamiento de datos	Fácil	Si

Tabla 4. Comparación de los métodos candidatos para el análisis principal.

3.4. Selección del modelo

Después de comparar los cuatro modelos anteriores, usaremos LightGBM para nuestro análisis. Es el modelo que mejor destacará las variables que más afectan a la popularidad, ya que este es el principal objetivo de la investigación. Mientras se cumple esto, también tendrá en cuenta otros factores como los datos desequilibrados, estando diseñado para este tipo de desequilibrios, en este caso se ocupará del hecho de que hay muchas observaciones con un valor de popularidad de 55. Por último, dado que la relación entre las características acústicas y la popularidad musical puede ser no lineal y presentar interacciones complejas, este es el método que mejor se adapta, permitiendo capturar dichas relaciones sin imponer supuestos funcionales estrictos, mejorando la capacidad predictiva al respecto a modelos lineales.

El modelo será implementado de la siguiente forma. Primero, prepararemos los datos dividiendo los datos en train y test splits, siendo necesario para evaluar el rendimiento del modelo. Los datos se transformarán al formato "lgb.Dataset", que es el formato preferido para LightGBM, ya que está optimizado tanto para la eficiencia de la memoria como para la velocidad de entrenamiento. El modelo tiene un conjunto de parámetros, como el número de rondas de refuerzo y las rondas de detención temprana. Para asegurarnos de obtener los mejores resultados, necesitamos ajustar estos parámetros, lo que podremos hacer probando muchas combinaciones diferentes en el propio conjunto de datos. Una vez obtenidos los resultados del modelo, generamos el gráfico de importancia de características, lo que nos proporcionará una visualización clara de las variables relevantes y nos ayudará a dar respuesta a la pregunta central de investigación de este trabajo. Posteriormente, exploramos los efectos de interacción para identificar si existen otras variables que actúan como moderadoras en la relación entre la popularidad y las variables principales que influyen en su valor (Anexo 3).

A parte de todos los pasos anteriores sobre la utilización del modelo LightGBM, sus gráficos e interpretar los resultados con SHAP, iremos un paso más allá con una simulación contrafactual creando un modelo predictivo, empleando machine learning. Primero crearemos canciones aleatorias ficticias, con el motivo de poner a prueba nuestro modelo predictivo. Con el modelo entrenado de LightGBM anteriormente, haremos la predicción de la popularidad predicha de nuestras canciones simuladas, descargandolas en un CSV para poder visualizar las canciones, sus variables y la popularidad predicha (Anexo 4). Todo esto se presentará con mayor detalle en el apartado siguiente, correspondiente a los resultados.

4. Resultados

4.1. Análisis principal

El modelo LightGBM nos ayuda a señalar la importancia de cada una de las variables analizadas, haciendo un ranking de éstas mismas, como enseña la *Figura 1*, siendo necesario para saber cuáles de estas características tendremos en cuenta para comprender qué es lo que impulsa la popularidad de las canciones.

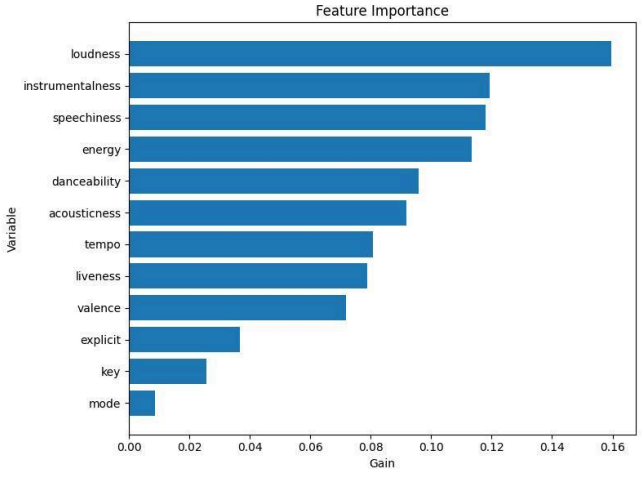


Figura 1. Clasificación de importancia de las características.

Por otro lado, la *Figura 2* muestra los valores SHAP de las variables de nuestro modelo de LightGBM, los cuales utilizaremos para comprender cuáles de los predictores relevantes tienen un efecto positivo en la popularidad. Un valor positivo significa que los predictores tienen un efecto positivo en las predicciones, en este caso en el valor de la popularidad, y, a la inversa, los negativos tienen un efecto negativo.

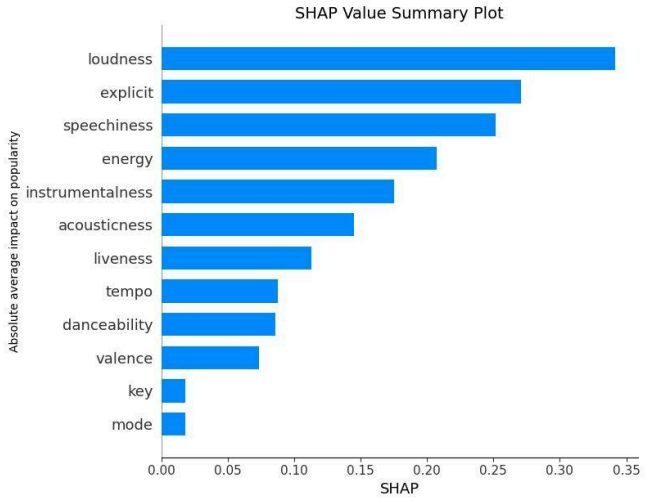


Figura 2. Valores SHAP del modelo LightGBM.

El gráfico SHAP muestra la importancia global de cada variable, calculada como la media del valor absoluto de los valores SHAP, facilitando la compresión directa entre características. Viendo el gráfico, podemos apreciar que las variables que tienen un efecto positivo sobre *popularity* son todas, esto no es algo normal, es debido a que nuestro conjunto de datos tiene un rango de popularidad popularidad de 55 puntos o mayor, por lo que vamos a centrarnos en las que tienen el SHAP del 20% o mayor que serían *loudness, explicit, speechiness* y *energy*. Podemos comprobar que los resultados son diferentes a los del modelo de regresión lineal, donde solo teníamos positivas *explicit, loudness* y *mode*, aunque las dos primeras son iguales, vemos que *mode* en con SHAP está el último. Y en el caso de *explicit* y *loudness* se han intercambiado puestos, mientras con el primer método *explicit* era la que más positivamente afectaba, ahora lo es *loudness* con más de 5% de diferencia. Otro caso a tener en cuenta es el de *speechiness*, con el método de regresión linear era el la variable que más negativamente afectaba a la popularidad, ahora es la tercera que mejor la afecta. Con este análisis podemos ver lo importante que es elegir un buen método para nuestro tipo de datos.

4.2. Modelo predictor

Con el modelo de machine learning, lo que se quiere conseguir es predecir cómo de popular puede ser una canción antes de que salga en las plataformas, en este caso Spotify. Nosotros lo que hemos hecho en este punto, como no tenemos canciones reales que una discográfica, es crear unas canciones simuladas, con ayuda del método LightGBM entrenado del punto anterior, permitiendo simular la popularidad esperada de nuevas

producciones musicales en función de sus características acústicas. Una vez generadas 100 pistas ficticias ejecutamos el modelo predictivo y su popularidad predicha. Con este tipo de modelo predictor, una discográfica podría coger una canción antes de lanzamiento y con sus variables iguales que las de Spotify, predecir cómo de popular va a ser ésta. Al igual que una canción recién o ya sacada en Spotify y hacer la predicción. Esta forma de actuar sería muy buena si lo que busca la discográfica o el mismo artista es que su canción sea el próximo "hit".

5. Conclusión

5.1. Resumen de la investigación

En este trabajo nos hemos enfocado en el objetivo principal de saber qué hace a una canción popular. El ángulo sobre el que hemos realizado el análisis son los elementos acústicos por la anatomía de una canción. Las contribuciones aportadas son tanto empresariales como académicas, entendiendo qué hace a una canción popular es importante para un negocio en la industria musical, como una discográfica o artista, y les puede ayudar a realizar decisiones que resulten más rentables sobre el lanzamientos de canciones, álbumes o incluso artistas.

Los elementos analizados son variables basadas en audio, que pueden ser vinculadas con el estado de ánimo, con las propiedades y con el contexto de las canciones (Spotify for Developers, 2025). Los datasets utilizados fueron descargados por Anna's Archive (2025) mediante la API de Spotify, la cual habilita la posibilidad de acceder a las databases de canciones, álbumes y podcasts. Los datasets obtenidos gracias a Anna's Archive eran de la totalidad de canciones de Spotify, desde el 2010 hasta julio de 2025, un total de más de 256 millones de pistas. Estos fueron acortadas a las 100.000 más populares de esa lista, limpiadas para eliminar las pistas sin características acústicas y unidos para tener las variables necesarias para nuestro análisis en un solo dataset, quedándonos con casi 99.000 pistas.

Empezamos haciendo el método de regresión lineal, donde las variables más importantes fueron *explicit* y *loudness*, siendo las dos variables positivas más altas, es decir, que más afectan a la popularidad de una canción. Con este método nos dimos cuenta que no era la mejor idea para nuestro análisis, así que comparamos cuatro métodos candidatos, reflejados en la *Tabla 4*, y nos decantamos por usar el método de machine learning LightGBM ya que es el que mejor se adapta a nuestro análisis, ayudándonos también de SHAP. En estos encontramos que las variables que mejor afectan a la popularidad son l*oudness, speechiness, energy* y *explicit*.

5.2. Interpretación de los resultados

La variable que hemos encontrado más importante en este trabajo es *loudness*, ya que en todos los métodos utilizados es la única siempre ha tenido una relación positiva con la popularidad, lo que quiere decir que las canciones que son grabadas con un sonido más fuerte tienden a ser más populares, al menos con este dataset.

Esta ha sido la tendencia en las últimas décadas, apareciendo el término "loudness war" que hace referencia a que en los últimos 60 años las discográficas han competido para hacer las canciones con un sonido más fuerte, a menudo reduciendo el rango dinámico con compresión (Bunning, 2018). Este fenómeno se debe a varios factores. Por una parte, los vinilos tienen límites físicos, los surcos más profundos del disco requieren más espacio y reducen el número de canciones que caben en cada una de sus caras, por lo que el vinilo impuso un límite práctico al volúmen hasta la llegada de los métodos digitales. Cuando llegaron éste tipo de formato, los ingenieros musicales acercaron los niveles medios al límite, utilizando la compresión para aumentar el volumen medio. Y en los últimos años con las plataformas digitales, no hay límites, y las canciones con un sonido más fuerte tienden a ser más populares (Bunning, 2018).

La otra variable con mayor peso positivo en los modelos es *energy*, siendo canciones con mayor emoción e intensidad física, sintiéndose rápidas, fuertes y ruidosas. Como plantea Kim (2021) esto ya fue tendencia entre 2011 y 2015, donde las canciones muy energéticas estaban muy relacionadas con la popularidad. También se sugiere, que las características *loudness* y *energy* van muy de la mano, al igual que pasa en nuestro análisis. Las dos variables van muy de la mano junto con los latidos por minuto ("beats per minute", bpm) de una canción, debido a que las personas sentimos más energía con unos bpm más altos, y latidos y sonidos más fuertes (Kim, 2021).

5.3. Implicaciones directivas

Hay muchos tipos de personas que trabajan en la industria musical para las que los resultados de esta investigación tienen implicaciones y valor. Las principales son los ejecutivos musicales, los artistas, los productores y las plataformas de streaming, como Spotify.

No existe una fórmula puramente técnica para crear un éxito musical, ya que el rendimiento de una canción depende en gran medida de la marca del artista, el marketing que emplee su discográfica para él/ella, la inclusión en las listas por parte de las plataformas de streaming, las tendencias culturales y las redes sociales. Existen artistas que saquen lo que saquen, va a ser popular, como lo puede ser Taylor Swift o Harry Styles, teniendo ya un grupo de fans dedicados es muy fácil para ellos llegar a las listas de éxitos y que sus canciones estén entre las más escuchadas mundialmente, gracias a su imagen como artistas y el marketing empleado por sus discográficas. Hablando de las redes sociales, y con la nueva generación de creación de contenido, muchos artistas ven sus canciones siendo "trend" en redes sociales y dependen de ese marketing orgánico para tener sus "hits".

Sin embargo, los modelos de machine learning como el creado en este trabajo, pueden utilizarse como herramientas de filtrado de demos, simuladores de potencial comercial, y sistemas de apoyo a decisiones de A&R (Artists and Repertoire), siendo la división de las compañías encargada de buscar talentos, firmar a nuevos artistas y de supervisar el desarrollo creativo. En este sentido, el análisis de datos no sustituye la intuición creativa, pero puede aportar una ventaja competitiva a la hora de tomar decisiones.

5.4. Limitaciones y futuras líneas de investigación

Limitaciones del estudio

Esta investigación está centrada en los elementos que más contribuyen a qué hace a una canción popular, por medio del análisis de las variables basadas en audio de Spotify. Esto provoca que toda la investigación se base en la precisión y qué tipo de datos el equipo de datos de Spotify genera y pone a disposición del público, siendo un punto a tener en cuenta como una debilidad de la investigación, ya que en su web no se especifica cómo y con qué precisión se generan. Además contamos con una base de datos recortada a 100.000 observaciones, que puede considerarse como un tamaño de data limitado.

Asimismo, el modelo se ha construido únicamente con características acústicas de las canciones, dejando fuera variables estratégicas clave como la popularidad del artista, el número de seguidores, la duración y años de publicación de las canciones, el presupuesto de marketing y el género musical. La ausencia de estos factores determinantes en el éxito comercial, explica el bajo R^2 observado en el modelo lineal. La popularidad de una canción se crea en la intersección de dimensiones creativas, sociales, culturales y estratégicas, haciendo referencia a la naturaleza multifactorial del éxito musical. Un modelo solo basado en el audio no puede capturar plenamente esta complejidad, lo que limita la capacidad predictiva y explicativa de los resultados.

Además, la variable *popularity* de Spotify es un índice interno cuyo cálculo exacto no está en nuestras manos, haciendo que esta opacidad introduce una incertidumbre en la interpretación y generalización de los hallazgos, ya que no se conoce con precisión como la plataforma pondera distintos factores que influyen en dicho índice.

Futuras líneas de investigación

A partir de las limitaciones detectadas, se proponen nuevas líneas de investigación posibles. Una sería incorporar otro tipo de variables de la canción, como la fecha de lanzamiento, la duración de la canción, el idioma o el artista. Junto con este último, se podrían incluir los seguidores, el número de reproducciones totales, o si son colaboraciones o no, lo que permitiría distinguir el efecto "marca del artista" del efecto estrictamente musical. También se podrían analizar la inclusión en playlist creadas por la misma plataforma de streaming, como "Top Hits", o la viabilidad en redes sociales.

Otro punto interesante a investigar es el análisis por género musical, segmentando el estudio por géneros (pop, rock, hip-hop, rap, o clásica entre otros) abriría un análisis más completo y ayudaría a capturar cómo varía la relación entre las características acústicas y la popularidad en cada estilo. Por último, analizar la evolución temporal de las canciones populares, tendencias musicales, cambios en la producción y el impacto de nuevas plataformas o formatos, permitiría identificar patrones dinámicos y anticipar cambios en el éxito futuro.

6. Bibliografía

Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012, Septiembre). Random Forests and Decision Trees. *International Journal of Computer Science Issues*, *9*(5), 272-278. https://ijcsi.org/papers/IJCSI-9-5-3-272-278.pdf

Anna's Archive. (2025, December 20). *Backing up Spotify*. https://web.archive.org/web/20260108103928/https://annas-archive.li/torrents/spotify

Berger, J. (2021). *What Makes Songs Popular? It's All About 'You'*. Knowledge at Wharton. https://knowledge.wharton.upenn.edu/podcast/knowledge-at-wharton-podcast/what-makes-songs-popular-its-all-about-you/

Borg, N., & Hokkanen, G. (2011). *What makes for a hit pop song?* https://cs229.stanford.edu/proj2011/BorgHokkanen-WhatMakesForAHitPopSong.pdf

Bunning, J. (2018, Octubre 26). The Loudness Wars. *USC Illumin Magazine*, *XVIII*(II). https://illumin.usc.edu/the-loudness-wars/

Chen, C., Liaw, A., & Breiman, L. (2004). *Using Random Forest to Learn Imbalanced Data*. https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

Detienne, K. B., Detienne, D. H., & Joshi, S. A. (2003). Neural Networks as Statistical Tools for Business Researchers. *Organizational Research Methods*, (6), 236-259. https://scispace.com/pdf/neural-networks-as-statistical-tools-for-business-2vbej8a2sw.pdf

Echambadi, R., & Hess, J. D. (2007). Mean-Centering Does Not Alleviate Collinearity Problemsin Moderated Multiple Regression Models. In *Marketing Science* (pp. 438-444). INFORMS.

Ferreri, L., Mas-Herrero, E., J. Zatorre, R., Ripollés, P., Gomez-Andres, A., Alicart, H., Olivé, G., Marco-Pallarés, J., Antonijoan, R. M., Valle, M., Riba, J., & Rodriguez-Fornells, A. (2019). *Dopamine modulates the reward experiences elicited by music*. https://www.pnas.org/doi/full/10.1073/pnas.1811878116

IFPI. (2023). *IFPI's global study finds we're listening to more music in more ways than ever*. https://www.ifpi.org/ifpis-global-study-finds-were-listening-to-more-music-in-more-ways-than-ever/

IFPI. (2025). *Global Music Report 2025*. https://www.ifpi.org/wp-content/uploads/2024/03/GMR2025_SOTI.pdf

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., UE, Q., & Liu, T.-Y. (2017, December). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, *30*.

Kim, J. (2021, Octubre 29). Music Popularity Prediction Through Data Analysis ofMusic's Characteristics. *nternational Journal of Science, Technology and Society*, *9*(5).

Spotify for Developers. (2025). *Spotify for Developers*. Spotify for Developers. Retrieved 2026, from https://developer.spotify.com/community

Statista. (2025, December 17). *Spotify - statistics & facts*. Statista. https://www.statista.com/topics/2075/spotify/?srsltid=AfmBOoo5L6RSEOmIb_TCyuYTBSN1so960NAZ3PXgMFf6LmF4uz9GcgSE#topicOverview

Students of PhD at the Asian Institute of Management. (2020). *Chapter 1: AutoRegressive Integrated Moving Average (ARIMA)*. GitHuN. https://phdinds-aim.github.io/time_series_handbook/Preface/Preface.html

Terroso-Saenz, F., Soto, J., & Muñoz, A. (2022). Evolution of global music trends: An exploratory and predictive approach based on Spotify data. *Entertainment Computing*, *44*(3).

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267-288.

6. Anexos

Antes de meternos a los anexos, con los scripts y su desarrollo, quiero adjuntar el link del GitHub donde se pueden encontrar la base de datos principal con la que se desarrolla esta investigación, siendo esta la cortada y limpiada como explicamos en el *Anexo 1*, así como todos los scripts en sus formatos, aunque en los anexos también los tenéis. También están algunos documentos CSV, siendo los resultados de algunos scripts. En general, están todos los documentos y respuestas de este trabajo.

6.1. Anexo 1

Tenemos dos bases de datos que hay que juntar para poder trabajar con una única base de datos, haciendo más fácil la labor del análisis de los datos y nuestro modelo, así que para ello utilizaremos SQL, en mi caso, y por el gran volumen de datos que tenemos en un primer momento, he optado por usar DataGrip ya que me permite subir archivos grandes y con gran cantidad de datos al programa sin problema y juntar las dos bases de datos.

Las dos bases de datos principales son demasiado grandes como para compartirlas, pero en este trabajo se han compartido links para que puedan descargarlas, aún así se compartirán la base de datos limpia y unida, que será la base de los análisis posteriores.

Para limpiar los datos, cargamos ambas bases de datos en DataGrip. Primero trabajaremos en la consola de la base de datos, "spotify_clean.sqlite3", en esta seleccionaremos las variables que nos son útiles para crear nuestra nueva base de datos (name, id, popularity, duration_ms, etc.), obtendremos todos los nombres e ID's de artistas como matriz JSON de objetos, también obtendremos todos los ID's de pista con el mismo ISRC (duplicados) como matriz JSON, creamos un inner join entre tracks y albums usando el ID del álbum, filtraremos solo las pistas que tienen un código ISRC asignado, y por último, ordenaremos por popularidad descendente y limitamos los resultados a 100.000. Este último paso lo he realizado para acortar la base de datos, que sea más manejable y solo tengamos canciones con la popularidad de 55 o mayor número. La respuesta al código, guardada como CSV, también se ilustra a continuación.

```
-- Seleccionamos todas las columnas que nos son necesarias por la información
que tienen
SELECT
 tracks.rowid,
 tracks.external_id_isrc AS isrc,
 tracks.name AS track_name,
 tracks.id AS track_id,
 tracks.popularity,
 tracks.duration_ms,
 tracks.explicit,
 albums.name AS album_name,
```

```
 albums.album_type,
 albums.release_date,
 albums.rowid AS album_rowid,
 albums.id AS album_id,
 -- Obtener todos los nombres e ID de artistas como matriz JSON de objetos
 (SELECT json_group_array(json_object('name', artists.name, 'id', artists.id))
  FROM track_artists
  JOIN artists ON artists.rowid = track_artists.artist_rowid
  WHERE track_artists.track_rowid = tracks.rowid) AS artist_names,
 -- Obtener todos los ID de pista con el mismo ISRC (duplicados) como matriz
JSON
 (SELECT json_group_array(t2.id)
  FROM tracks t2
  WHERE t2.external_id_isrc = tracks.external_id_isrc) AS copies
FROM tracks
-- Hacemos un inner join entre tracks y albums usando el ID del álbum
INNER JOIN albums ON tracks.album_rowid = albums.rowid
-- Filtrar solo las pistas que tienen un código ISRC asignado
WHERE tracks.external_id_isrc IS NOT NULL
-- Ordena por popularidad descendente (los más populares primero)
ORDER BY tracks.popularity DESC
-- Limita el resultado a 100.000 registros
LIMIT 100000;
```

Script 1. Primer script SQL, sobre la base de datos "spotify_cleaned", realizado en DataGrip.

	rowid	isrc	track_name	track_id	popularity	duration_ms	explicit	album_name	album_type	release_date
1	1178	USUM72409273	Die With A Smile	2plbrEY59IikOBgBGLjaoe	100	251667	0	Die With A Smile	single	2024-08-16
2	2998	QMFMF2447070	DtMF	3sK8wGT43QFpWrvNQsrQya	98	237117	1	DeBÍ TiRAR MáS FOToS	album	2025-01-05
3	269	USUM72401994	BIRDS OF A FEATHER	6d0tVTDdiauQNBQED0tlAB	98	210373	0	HIT ME HARD AND SOFT	album	2024-05-17
4	87640805	NLC8M2000219	Clean Baby Sleep White Noise (Loopable)	0zirWZTcXBBwGsevrsIpvT	97	142222	0	Best White Noise For Sleeping Baby	single	2020-04-29
5	2985	QMFMF2447057	BAILE INoLVIDABLE	2lTm559tuIvatlT1u0JY02	96	367725	1	DeBÍ TiRAR MáS FOToS	album	2025-01-05
6	2779	USUG12400910	Not Like Us	6AI3ezQ4o3HUoP6Dhudph3	96	274192	1	Not Like Us	single	2024-05-04
7	290	USUG12406903	That's So True	7ne4VBA60CxGM75vw0EYad	96	166300	1	The Secret of Us (Deluxe)	album	2024-10-18
8	266676	USAT22409172	APT.	5vNRhkKd0yEAg8suGBpjeY	95	169917	0	APT.	single	2024-10-18
9	4337	USUM71713947	All The Stars (with SZA)	3GCdLUSnKSMJhs4Tj6CV3s	95	232186	1	Black Panther The Album Music From And Inspired By	album	2018-02-09
10	3156	USHM92438095	Sailor Song	2262bWmqomIaJXWcRHr13j	95	211978	0	Sailor Song	single	2024-07-26
11	270	USUM72401993	WILDFLOWER	3QaPy1KgI7nu9FJEQUgn6h	95	261466	0	HIT ME HARD AND SOFT	album	2024-05-17
12	41772	USUG12408496	luther (with sza)	2C6NAOSuO1MEFCbBRgUzjd	94	177598	0	GNX	album	2024-11-21
13	3791	USUG12408493	tv off (feat. lefty gunplay)	0aB0v4027ukVziUGwVGYpG	94	220690	1	GNX	album	2024-11-22
14	2983	QMFMF2447055	NUEVAYoL	5TFD2bmFK6hoCRbX61nXY5	94	183685	0	DeBÍ TiRAR MáS FOToS	album	2025-01-05
15	2590	USAT22500463	Ordinary	6qqrTXSdwiJaq8SO0X2lSe	94	186964	0	Ordinary	single	2025-02-07
16	265	USUG12401967	Good Luck, Babe!	0WbMK4wrZ1wFSty9F7FCgu	94	218423	0	Good Luck, Babe!	single	2024-04-05
17	268793	USUM72412581	Abracadabra	5ZLUm9eab8y3tqQ1OhQSHI	93	223398	0	Abracadabra	single	2025-02-03
18	3787	USUG12408494	luther (with sza)	45J4avUb9NiObnETYaYFVJ	93	177598	0	GNX	album	2024-11-22
19	2997	QMFMF2447069	Eo0	6J5kc128W5HuP3d7C3vvx8	93	204768	1	DeBÍ TiRAR MáS FOToS	album	2025-01-05
20	2988	QMFMF2447060	VeLDÁ	7d6yK8v8J484SWH5prIQiE	93	235336	1	DeBÍ TiRAR MáS FOToS	album	2025-01-05
21	2984	QMFMF2447056	VOY A LLeVARTE PA PR	59D4DOkspUbWyMmbAPQkxZ	93	156304	1	DeBÍ TiRAR MáS FOToS	album	2025-01-05
22	2620	USUG12406536	Timeless (feat Playboi Carti)	0FIDCNYYjNvPVimzSicugS	93	256000	1	Hurry Up Tomorrow	album	2025-01-31
23	2572	USQX92500261	like JENNIE	0fK71e6XwGxQTIkpFoWkd1	93	123517	1	Ruby	album	2025-03-07
24	189335	USUG12303881	One Of The Girls (with JENNIE, Lily Rose Depp)	7CyPwkpOoE8Ro9Dd5CUDjW	92	244084	0	The Idol Episode 4 (Music from the HBO Original Series)	single	2023-06-23

Figura 3. Resultados del primer código SQL, realizado en DataGrip.

Con el fin de centralizar la información, los resultados obtenidos serán persistidos temporalmente en un archivo plano (CSV) para su posterior migración a la instancia principal de SQLite. Al integrar los datos en una única unidad lógica, eliminamos la necesidad de utilizar comandos ATTACH DATABASE, lo que

previene la degradación del rendimiento asociada al manejo de múltiples descriptores de archivo y mejora la velocidad de indexación durante el análisis.

Sobre la base de datos unificada “spotify_clean_audio_features”, lanzaremos la siguiente query, el cual nos permitirá crear índices para evitar que se recorran todas las líneas y agilizar, estos índices se tienen que ejecutar antes que el resto del código, y es la parte en la que más se nota la optimización del tiempo del anterior paso. Limpiamos la variable de artist_names, extrayendo y concatenando los nombres y los ID’s, separándolos en dos columnas distintas, para que nos quede un dataset más limpio. Mediante una sentencia de unión (JOIN), se integraron las tablas spotify_recleaned (denominada spotify) y spotify_clean_audio_features (denominada features). Este proceso de selección y cruce de variables permite consolidar toda la información relevante en un único dataset de trabajo. Solo incluimos pistas con los valores de danceability y energy conocido, este paso es para asegurarnos de solo incluir pistas con las variables necesarias conocidas, ya que en el dataset original nos encontramos con pistas en las cuales estas variables necesarias están todas en blanco. Finalmente, ordenamos el resultado por popularidad descendente y limitamos el resultado a 100.000 pistas, aunque este último no sería necesario.

Con estos pasos tendremos una nueva base de datos, ya limpia, cortada y unida, a la cual hemos llamado “spotify_with_features” y guardada en CSV. Como resultado de este flujo de trabajo, se ha generado un dataset consolidado denominado “spotify_with_features”. Este conjunto de datos, tras haber sido sometido a procesos de limpieza, filtrado de atributos y unión relacional, se ha persistido en formato CSV para facilitar su portabilidad y consumo en las siguientes etapas de análisis y modelado.

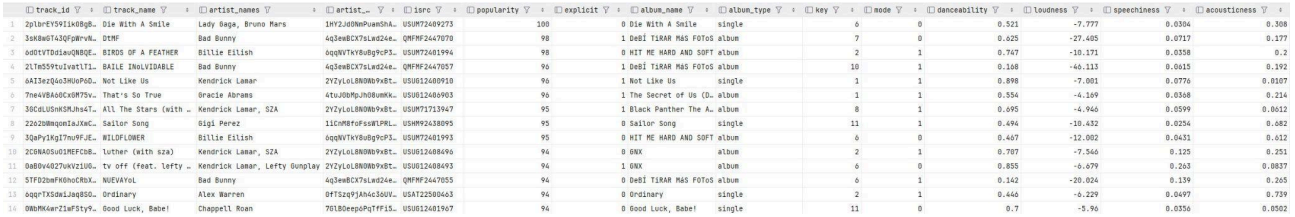
```
-- Creamos index para evitar que se recorran todas las líneas de la base de
datos y agilizar el proceso
CREATE INDEX IF NOT EXISTS idx_audio_track_id
ON track_audio_features(track_id);

CREATE INDEX IF NOT EXISTS idx_spotify_top_track_id
ON spotify_recleaned(track_id);

SELECT
   spotify.track_id,
   spotify.track_name,
   -- Extrae y concatena los nombres de los artistas de la pista en un solo
string, separados por comas
   (
       SELECT group_concat(json_extract(value, '$.name'), ', ')
       FROM json_each(spotify.artist_names)
   ) AS artist_names,
```

```
    -- Extrae y concatena los IDs de los artistas asociados a la pista en un solo
string, separados por comas
    (
        SELECT group_concat(json_extract(value, '$.id'), ', ')
        FROM json_each(spotify.artist_names)
    ) AS artist_id,
    spotify.isrc,
    spotify.popularity,
    spotify.explicit,
    spotify.album_name,
    spotify.album_type,
    features.key,
    features.mode,
    REPLACE(CAST(features.danceability AS TEXT), '.', ',') AS danceability,
    REPLACE(CAST(features.loudness AS TEXT), '.', ',') AS loudness,
    REPLACE(CAST(features.speechiness AS TEXT), '.', ',') AS speechiness,
    REPLACE(CAST(features.acousticness AS TEXT), '.', ',') AS acousticness,
    REPLACE(CAST(features.energy AS TEXT), '.', ',') AS energy,
    REPLACE(CAST(features.valence AS TEXT), '.', ',') AS valence,
    REPLACE(CAST(features.tempo AS TEXT), '.', ',') AS tempo,
    REPLACE(CAST(features.instrumentalness AS TEXT), '.', ',') AS instrumentalness,
    REPLACE(CAST(features.liveness AS TEXT), '.', ',') AS liveness
-- Renombramos la tabla de spotify_recleaned como spotify
FROM spotify_recleaned spotify
-- Unimos con la tabla track_audio_features y renombramos la tabla de
track_audio_features como features
JOIN track_audio_features features
    ON spotify.track_id = features.track_id
-- Solo incluimos pistas con valor de "danceability" y "energy" conocido
WHERE
    features.danceability IS NOT NULL
    AND features.energy IS NOT NULL
-- Ordenamos el resultado por popularidad (los más populares primero)
ORDER BY spotify.popularity DESC
-- Limitamos el resultado a un máximo de 11,000 filas
LIMIT 100000;
```

Script 2. Segundo script SQL, sobre la base de datos "spotify_clean_audio_features.sqlite3", realizado en DataGrip.



	track_id	track_name	artist_names	artist_...	isrc	popularity	explicit	album_name	album_type	key	mode	danceability	loudness	speechiness	acousticness
1	2plbrEY59IikOBgB…	Die With A Smile	Lady Gaga, Bruno Mars	1HYZJdONmPuamShA…	USUM72409273	100	0	Die With A Smile	single	6	0	0.521	-7.777	0.0304	0.308
2	3sK8wGT43QFpWrvN…	DtMF	Bad Bunny	4q3ewBCX7sLwd24e…	QM4MF2447070	98	1	DeBÍ TiRAR MáS FOTOS	album	7	0	0.625	-27.405	0.0717	0.177
3	6dOtVTDdiauQNBQE…	BIRDS OF A FEATHER	Billie Eilish	0qqNVTkY8uBg9cP3…	USUM72401994	98	0	HIT ME HARD AND SOFT	album	2	1	0.747	-10.171	0.0358	0.2
4	2LTm559tuIvatlT1…	BAILE INoLVIDABLE	Bad Bunny	4q3ewBCX7sLwd24e…	QM4MF2447057	96	1	DeBÍ TiRAR MáS FOTOS	album	10	1	0.168	-46.113	0.0615	0.192
5	6AI3ezQ4oJHUoP6D…	Not Like Us	Kendrick Lamar	2YZyLoL8N0Wb9xBt…	USUG12400910	96	1	Not Like Us	single	1	1	0.898	-7.001	0.0776	0.0107
6	7ne4VBA60CxGM75v…	That's So True	Gracie Abrams	4tuJ0bMpJh08umKk…	USUG12406903	96	1	The Secret of Us (D…	album	1	1	0.554	-4.169	0.0368	0.214
7	36CdLUSnKSMJhs4T…	All The Stars (with …	Kendrick Lamar, SZA	2YZyLoL8N0Wb9xBt…	USUM71713947	95	1	Black Panther The A…	album	8	1	0.695	-4.946	0.0599	0.0612
8	2262bWmqomIaJXwC…	Sailor Song	Gigi Perez	1iCnM8foFssWlPRL…	USHM92638095	95	0	Sailor Song	single	11	1	0.494	-10.432	0.0254	0.682
9	3QaPy1kgI7nu9FJE…	WILDFLOWER	Billie Eilish	0qqNVTkY8uBg9cP3…	USUM72401993	95	0	HIT ME HARD AND SOFT	album	6	0	0.467	-12.002	0.0431	0.612
10	2C6NAOSuO1MEFCbB…	luther (with sza)	Kendrick Lamar, SZA	2YZyLoL8N0Wb9xBt…	USUG12408496	94	0	GNX	album	2	1	0.707	-7.546	0.125	0.251
11	0aB0v4027wkVz1U6…	tv off (feat. lefty …	Kendrick Lamar, Lefty Gunplay	2YZyLoL8N0Wb9xBt…	USUG12408493	94	1	GNX	album	6	0	0.855	-6.679	0.263	0.0837
12	5TFD2bmFK6hoCRbX…	NUEVAYoL	Bad Bunny	4q3ewBCX7sLwd24e…	QM4MF2447055	94	0	DeBÍ TiRAR MáS FOTOS	album	6	1	0.142	-20.024	0.139	0.265
13	6qqrTXSdwiJaq8SO…	Ordinary	Alex Warren	0fTSzq9jAh4c3oUV…	USAT22500463	94	0	Ordinary	single	2	1	0.446	-6.229	0.0497	0.739
14	0WbMK4wrZ1WF5ty9…	Good Luck, Babe!	Chappell Roan	76lB0eepoPq7fFi5…	USUG12401967	94	0	Good Luck, Babe!	single	11	0	0.7	-5.96	0.0356	0.0502

Figura 4. Resultados del segundo código SQL, realizado en DataGrip.

6.2. Anexo 2

Con la base de datos limpia, podemos empezar a escribir nuestro código Python, para el análisis de los datos he decidido usar el IDE Visual Studio Code ya que es uno de los IDE mas simples y funcionales actualmente. En este primer archivo python (spotify_lightgbm.py) realizaremos la regresión linear múltiple, donde definiremos nuestras 12 variables independientes, y nuestra variable dependiente, *popularity*, y estimamos con el modelo OLS. Añadimos una función para que nos saque las estrellas significativas, ayudándonos a ver los datos de una forma más rápida y visual, y construimos y creamos la tabla de resultados, junto con con el R^2, el R^2 ajustado y el número de observaciones. Con este código y tabla de salida en el terminal podemos argumentar nuestro análisis simple del punto 3.1.

```
# REGRESIÓN LINEAL MÚLTIPLE SPOTIFY
# Antes de ejecutar el código, instalar "pip install statsmodels" y "pip install
jinja2" en el terminal
import pandas as pd
import statsmodels.api as sm
# Cargar los datos
df = pd.read_csv("DataBases\spotify_with_features.csv", decimal=",")
# Definir las variables
y = df["popularity"]
X = df[
    [
        "explicit",
        "key",
        "mode",
        "danceability",
        "loudness",
        "speechiness",
        "energy",
        "valence",
        "acousticness",
        "tempo",
        "instrumentalness",
        "liveness",
    ]
]
```

```python
# Añadimos la constante
X = sm.add_constant(X)
# Estimar modelo OLS
model = sm.OLS(y, X).fit()
# Función para estrellas
def significance_stars(p):
    if p < 0.01:
        return "***"
    elif p < 0.05:
        return "**"
    elif p < 0.1:
        return "*"
    else:
        return ""
# Construir tabla de resultados
rows = []
for var in model.params.index:
    coef = model.params[var]
    se = model.bse[var]
    pval = model.pvalues[var]

    coef_str = f"{coef:.3f}{significance_stars(pval)}"
    se_str = f"({se:.3f})"

    rows.append([coef_str, se_str])
table = pd.DataFrame(
    rows,
    index=model.params.index,
    columns=["Coeficiente", "Error estándar"],
)
# Mostrar tabla de resultados
print("\nTABLA DE REGRESIÓN (DEPENDIENTE: POPULARITY)\n")
print(table)
# Información adicional de R²,R² ajustado y Obervaciones
print("\nR²:", round(model.rsquared, 3))
print("R² ajustado:", round(model.rsquared_adj, 3))
print("Observaciones:", int(model.nobs))
# Exportar resultados a CSV
table.to_excel("tabla_regresion_spotify.xlsx")
latex_table = table.to_latex(
    caption="Regresión lineal múltiple (variable dependiente: Popularity)",
    label="tab:spotify_regression",
    escape=False
)
with open("tabla_regresion_spotify.tex", "w") as f:
```

```
    f.write(latex_table)
```

Script 3. Primer script de Python en Visual Studio Code, spotify_regression.py.

```
Training until validation scores don't improve for 50 rounds
[100]   train's rmse: 5.82195   test's rmse: 5.88623
Early stopping, best iteration is:
[140]   train's rmse: 5.78832   test's rmse: 5.88444
R²: 0.023
RMSE: 5.884
            Variable  Importance_relativa
4           loudness             0.159707
10  instrumentalness             0.119225
5        speechiness             0.118078
6             energy             0.113266
3       danceability             0.095956
8       acousticness             0.091756
9              tempo             0.080589
11          liveness             0.078872
7            valence             0.071723
0           explicit             0.036735
1                key             0.025474
2               mode             0.008618
```

Figura 5. Resultados del primer script Python, realizado en Visual Studio Code.

6.3. Anexo 3

El siguiente script entrena un modelo LightGBM Regressor para explicar la popularidad de canciones de Spotify a partir de características acústicas, evaluando su rendimiento y analizando la importancia de las variables. Para ello importamos las librerías necesarias, cargamos y limpiamos los datos, eliminando los valores nulos. Tenemos que definir las variables, tanto la dependiente como el conjunto de variables explicativas acústicas, en este caso no es necesario que escalemos las variables porque LightGBM se basa en árboles. Dividimos el conjunto de datos en Train/Test, siendo un 80% de entrenamiento y un 20% de test, permitiendo evaluar el rendimiento fuera de muestra. Creamos los datasets de LightGBM, utilizando el formato propio de este para optimizar el entrenamiento, definimos los parámetros y entrenamos el modelo con Early Stopping. El modelo se entrena hasta un máximo de 1.000 árboles, pero se detiene automáticamente si no mejora en 50 iteraciones. Evaluamos el modelo utilizando la capacidad explicativa, R^2, y el error medio de predicción, RMSE. Por último en la parte de LightGBM, se calcula la importancia de cada variable según la ganancia promedio en los árboles, creando la *Figura 1* y su tabla (*Figura 6*).

Yendo un paso más allá, hacemos una interpretación avanzada con SHAP, ya que nos permite interpretar cómo cada variable contribuye a la predicción del modelo. Primero mostramos el conjunto de entrenamiento y calculamos los valores SHAP, añadimos la importancia global y creamos el gráfico de la *Figura 2*. Con esto, tenemos valores entre 0 y 1, un gráfico limpio y visualmente interpretable.

```
# ANÁLISIS DE POPULARIDAD CON LIGHTGBM
# Antes de ejecutar el código, instalar "pip install lightgbm shap" en el terminal
# Importar librerías
import pandas as pd
```

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import lightgbm as lgb
import shap
import matplotlib.pyplot as plt
# Cargar los datos
df = pd.read_csv("DataBases\spotify_with_features.csv", decimal=",")
df = df.dropna()
# Definir las variables (dependiente y explicativas)
y = df["popularity"]
X = df[
    [
        "explicit",
        "key",
        "mode",
        "danceability",
        "loudness",
        "speechiness",
        "energy",
        "valence",
        "acousticness",
        "tempo",
        "instrumentalness",
        "liveness",
    ]
]
# Hacer el test split
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)
# Crear data sets LightGBM
train_data = lgb.Dataset(X_train, label=y_train)
test_data = lgb.Dataset(X_test, label=y_test, reference=train_data)
# Definir parámetros del modelo, siendo estos estables, estándar y defendibles
params = {
    "objective": "regression",
    "metric": "rmse",
    "boosting_type": "gbdt",
    "learning_rate": 0.05,
    "num_leaves": 31,
    "max_depth": -1,
```

```
    "feature_fraction": 0.9,
    "bagging_fraction": 0.8,
    "bagging_freq": 5,
    "verbosity": -1,
    "seed": 42,
}
# Entrenamos el modelo con early stopping
model = lgb.train(
    params,
    train_data,
    num_boost_round=1000,
    valid_sets=[train_data, test_data],
    valid_names=["train", "test"],
    callbacks=[
        lgb.early_stopping(stopping_rounds=50),
        lgb.log_evaluation(100),
    ],
)
# Evaluación del modelo
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("R²:", round(r2, 3))
print("RMSE:", round(rmse, 3))
# Vemos la importancia de variables
importance = pd.DataFrame({
    "Variable": X.columns,
    "Importance": model.feature_importance(importance_type="gain"),
})
importance["Importance_relativa"] = (
    importance["Importance"] / importance["Importance"].sum()
)
importance = importance.sort_values("Importance_relativa", ascending=False)
print(importance[["Variable", "Importance_relativa"]])
# Creamos el gráfico
plt.figure(figsize=(8, 6))
plt.barh(
    importance["Variable"],
    importance["Importance_relativa"]
)
plt.xlabel("Gain")
plt.ylabel("Variable")
plt.title("Feature Importance")
plt.gca().invert_yaxis()  # variable más importante arriba
plt.tight_layout()
```

```
plt.show()
# 8) SHAP
sample_X = X_train.sample(5000, random_state=42)
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(sample_X)
plt.figure(figsize=(9, 6))
shap.summary_plot(
    shap_values,
    sample_X,
    plot_type="bar",
    show=False
)
plt.title("SHAP Value Summary Plot", fontsize=14)
plt.xlabel("SHAP")
plt.ylabel("Absolute average impact on popularity")
plt.tight_layout()
plt.show()
```

Script 4. Segundo script de Python en Visual Studio Code, spotify_lightgbm.py.

```
Training until validation scores don't improve for 50 rounds
[100]   train's rmse: 5.82195   test's rmse: 5.88623
Early stopping, best iteration is:
[140]   train's rmse: 5.78832   test's rmse: 5.88444
R²: 0.023
RMSE: 5.884
           Variable  Importance_relativa
4          loudness             0.159707
10  instrumentalness             0.119225
5       speechiness             0.118078
6            energy             0.113266
3      danceability             0.095956
8      acousticness             0.091756
9             tempo             0.080589
11         liveness             0.078872
7          valence             0.071723
0          explicit             0.036735
1               key             0.025474
2              mode             0.008618
```

Figura 6. Resultados del segundo script Python en VSCode, LightGBM.

6.4. Anexo 4

Para hacer el modelo predictivo de machine learning, necesitamos el modelo LightGBM anterior ya entrenado, así que las primeras líneas de código serán bastante parecidas a las del *Script 4*. La idea clave de este modelo es no generar números aleatorios, sino dentro de los rangos reales de Spotify, así que tenemos que incluir esos parámetros en nuestra generación de las canciones ficticias, pudiendo tener así canciones ficticias pero realistas. Para predecir la popularidad con LightGBM es muy importante que las columnas estén en el mismo orden que X_train, y añadimos la popularidad al Data Frame. Crearemos un documento CSV con los datos ficticios y su popularidad debido a esos datos, llamándolo

“canciones_simuladas_prediccion_popularidad.csv”, y ordenamos las canciones simuladas por orden de popularidad predicha.

```
# MODELO PREDICTIVO
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import lightgbm as lgb
import shap
import matplotlib.pyplot as plt
# Antes de generar canciones aleatorias, copiamos el modelo entrenado
# Cargar los datos
df = pd.read_csv("DataBases\spotify_with_features.csv", decimal=",")
df = df.dropna()
# Definir las variables (dependiente y explicativas)
y = df["popularity"]
X = df[
    [
        "explicit",
        "key",
        "mode",
        "danceability",
        "loudness",
        "speechiness",
        "energy",
        "valence",
        "acousticness",
        "tempo",
        "instrumentalness",
        "liveness",
    ]
]
# Hacer el test split
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)
# Crear data sets LightGBM
train_data = lgb.Dataset(X_train, label=y_train)
test_data = lgb.Dataset(X_test, label=y_test, reference=train_data)
# Definir parámetros del modelo, siendo estos estabñes, estándar y defendibles
params = {
```

```
    "objective": "regression",
    "metric": "rmse",
    "boosting_type": "gbdt",
    "learning_rate": 0.05,
    "num_leaves": 31,
    "max_depth": -1,
    "feature_fraction": 0.9,
    "bagging_fraction": 0.8,
    "bagging_freq": 5,
    "verbosity": -1,
    "seed": 42,
}
# Entrenamos el modelo con early stopping
model = lgb.train(
    params,
    train_data,
    num_boost_round=1000,
    valid_sets=[train_data, test_data],
    valid_names=["train", "test"],
    callbacks=[
        lgb.early_stopping(stopping_rounds=50),
        lgb.log_evaluation(100),
    ],
)
# Generamos canciones aleatorias pero realistas
def generar_canciones_ficticias(n_canciones=10, random_state=42):
    np.random.seed(random_state)
    data = {
        "explicit": np.random.randint(0, 2, n_canciones),
        "key": np.random.randint(0, 12, n_canciones),
        "mode": np.random.randint(0, 2, n_canciones),
        "danceability": np.random.uniform(0, 1, n_canciones),
        "loudness": np.random.uniform(-60, 0, n_canciones),
        "speechiness": np.random.uniform(0, 1, n_canciones),
        "energy": np.random.uniform(0, 1, n_canciones),
        "valence": np.random.uniform(0, 1, n_canciones),
        "acousticness": np.random.uniform(0, 1, n_canciones),
        "tempo": np.random.uniform(60, 200, n_canciones),
        "instrumentalness": np.random.uniform(0, 1, n_canciones),
        "liveness": np.random.uniform(0, 1, n_canciones),
    }
    canciones = pd.DataFrame(data)
    canciones.index = [f"Song_{i+1}" for i in range(n_canciones)]
    return canciones
# Generar 100 canciones simuladas
```

```python
canciones_simuladas = generar_canciones_ficticias(n_canciones=10000)
# Asegurar mismo orden de columnas
canciones_simuladas = canciones_simuladas[X.columns]
# Predicción
canciones_simuladas["popularidad_predicha"] = model.predict(canciones_simuladas)
# Guardar resultados en CSV
canciones_simuladas.to_csv(
    "canciones_simuladas_prediccion_popularidad.csv",
    index=True,
    float_format="%.2f",
    decimal="."
)
# Ordenar por canciones más prometedores
canciones_simuladas.sort_values(
    "popularidad_predicha",
    ascending=False
).head(10)
```

Script 5. Tercer script de Python en Visual Studio Code, model_predictor.py.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		explicit	key	mode	danceability	loudness	speechiness	energy	valence	acousticness	tempo	instrumentalness	liveness	popularidad_predicha
2	Song_1	0	2	0	0.58	-34.00	0.46	0.29	0.23	0.78	67.27	0.31	0.12	60.36
3	Song_2	1	7	1	0.13	-49.79	0.18	0.47	0.00	0.26	168.14	0.81	0.99	64.68
4	Song_3	0	6	0	0.23	-29.11	0.16	0.83	0.51	0.44	153.66	0.74	0.74	60.13
5	Song_4	0	2	1	0.34	-54.92	0.36	0.27	0.77	0.63	89.91	0.77	0.41	60.81
6	Song_5	0	5	1	0.20	-23.01	0.57	0.34	0.13	0.97	121.78	0.11	0.99	59.80
7	Song_6	1	3	1	0.15	-45.50	0.42	0.38	0.53	0.03	151.84	0.13	0.56	62.47
8	Song_7	0	10	1	0.06	-16.51	0.60	0.38	0.31	0.52	62.14	0.30	0.81	59.87
9	Song_8	0	4	0	0.87	-23.56	0.71	0.20	0.84	0.94	154.80	0.10	0.08	60.16
10	Song_9	0	10	0	0.29	-15.22	0.75	0.92	0.10	0.26	116.22	0.52	0.94	60.46
11	Song_10	1	4	0	0.29	-15.02	0.40	0.79	0.85	0.11	73.28	0.53	0.95	59.80
12	Song_11	0	6	0	0.21	-48.20	0.72	0.98	0.61	0.60	82.90	0.35	0.86	60.49
13	Song_12	0	10	0	0.48	-59.96	0.14	0.37	0.92	0.15	119.73	0.41	0.16	61.47
14	Song_13	0	4	0	0.51	-38.05	0.51	0.05	0.94	0.88	147.69	0.10	0.00	60.08
15	Song_14	0	10	1	0.21	-4.86	0.07	0.12	0.81	0.72	108.21	0.68	0.90	59.68
16	Song_15	1	2	0	0.61	-25.33	0.24	0.36	0.53	0.12	75.41	0.26	0.31	62.85
17	Song_16	0	9	0	0.72	-16.35	0.58	0.91	0.14	0.58	175.68	0.78	0.36	60.01
18	Song_17	1	4	0	0.37	-7.23	0.74	0.98	0.23	0.03	130.04	0.77	0.47	59.88
19	Song_18	1	6	1	0.59	-5.64	0.10	0.82	0.42	0.85	99.61	0.59	0.32	61.29
20	Song_19	1	4	0	0.50	-28.26	0.21	0.24	0.73	0.21	155.66	0.44	0.59	63.75
21	Song_20	0	4	1	0.20	-27.54	0.99	0.66	0.54	0.73	107.08	0.21	0.40	58.75
22	Song_21	1	8	0	0.73	-8.94	0.34	0.60	0.44	0.11	110.64	0.94	0.73	61.35
23	Song_22	0	2	0	0.80	-11.55	0.28	0.82	0.48	0.47	98.87	0.90	0.03	60.27
24	Song_23	1	5	0	0.65	-18.07	0.09	0.51	0.74	0.93	110.53	0.75	0.21	61.15
25	Song_24	1	0	1	0.32	-23.73	0.69	0.32	0.03	0.07	96.73	0.92	0.91	63.27
26	Song_25	1	9	1	0.93	-22.97	0.51	0.02	0.44	0.30	197.54	0.90	0.05	63.02
27	Song_26	1	4	0	0.46	-38.23	0.16	0.80	0.57	0.78	96.05	0.52	0.48	62.25
28	Song_27	1	7	0	0.27	-9.54	0.63	0.53	0.59	0.46	150.59	0.21	0.18	60.59
29	Song_28	1	4	0	0.41	-22.93	0.53	0.53	0.62	0.65	86.90	0.20	0.78	61.53
30	Song_29	1	8	1	0.69	-46.59	0.56	0.96	0.05	0.81	137.95	0.50	0.52	62.61
31	Song_30	1	10	1	0.85	-45.61	0.99	0.49	0.56	0.78	131.25	0.72	0.72	62.10
32	Song_31	0	5	0	0.36	-8.30	0.03	0.32	0.55	0.40	137.52	0.40	0.83	62.17
33	Song_32	0	7	1	0.23	-22.46	0.91	0.34	0.85	0.43	109.90	0.40	0.75	58.53
34	Song_33	1	9	1	0.63	-36.08	0.84	0.01	0.76	0.57	181.91	0.07	0.41	61.85
35	Song_34	1	7	1	0.92	-48.77	0.42	0.12	0.03	0.12	101.70	0.78	0.46	65.59
36	Song_35	1	2	1	0.18	-23.39	0.88	0.04	0.56	0.87	181.99	0.17	0.66	61.97
37	Song_36	0	6	0	0.80	-48.94	0.65	0.24	0.69	0.09	116.89	0.90	0.75	60.74
38	Song_37	1	6	0	0.48	-31.79	0.89	0.06	0.58	0.18	151.32	0.17	0.37	62.50
39	Song_38	0	9	1	0.80	-19.35	0.59	0.33	0.68	0.94	61.41	0.74	0.78	59.05

Figura 7. Resultados del tercer script en VSCode, canciones_simuladas_prediccion_popularidad.csv.