FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2022/2023

Trabajo de Fin de Máster

TÍTULO: **Análisis predictivo de la popularidad de una canción en Spotify**

***Alumno*: Luis María Plaza Chicote**

***Tutor*: Javier Castro Cantalejo**

Septiembre de 2023

UNIVERSIDAD COMPLUTENSE
MADRID

Resumen

En los últimos años, con el auge de las plataformas de entretenimiento por streaming ha revolucionado la forma en que las personas acceden y consumen contenido multimedia. En este contexto, las empresas de contenido se esfuerzan por aprovechar los datos generados por los consumidores de estas plataformas, con el objetivo de brindarles un servicio personalizado y de calidad que se adapte a sus gustos y preferencias.

La presente investigación aborda el análisis predictivo del nivel de popularidad de una canción en la plataforma de streaming musical Spotify. El objetivo principal es utilizar diversas técnicas de Machine Learning a través de una serie de atributos que ofrece la aplicación en cuanto a la canción, artista y álbum, para desarrollar un modelo que pueda predecir la popularidad futura de una canción.

Palabras clave: Spotify, Machine Learning, popularidad, streaming, música

Abstract

In the last years, the rise of streaming entertainment platforms has revolutionized the way in which people access and consume multimedia content. In this context, media companies are working to take advantage of the data generated by the consumers of these platforms, with the aim of providing them with a personalized and quality service that adapts to their tastes and preferences.

The present research addresses the predictive analysis of the popularity level of a song on the music streaming platform Spotify. The main objective is to use various Machine Learning techniques through a set of attributes offered by the application in terms of song, artist and album, to develop a model that can predict the future popularity of a song.

Keywords: Spotify, Machine Learning, popularity, streaming, music

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Definición del caso a investigar

En los últimos años, hemos presenciado una revolución en la forma en la que consumimos contenido multimedia. Con el auge de los servicios de streaming, plataformas como Netflix, HBO, Spotify, YouTube o Twitch han transformado la forma en que vemos películas y series, escuchamos música y podcasts o vemos transmisiones en vivo de videojuegos, ampliando enormemente nuestras opciones de entretenimiento online.

El streaming de video ha experimentado un crecimiento masivo en los últimos años. Plataformas como Netflix, Amazon Prime Video, HBO Max y Disney+ han acumulado más de 160 millones de suscriptores en todo el mundo cada una, ofreciendo una amplia variedad de películas, programas de televisión y documentales al alcance de un solo clic. Estos servicios no solo han transformado la forma en que consumimos contenido, sino que también han influido en la cultura popular, con series originales como Stranger Things y Game of Thrones que han ganado numerosos premios y han captado la atención del público. Por otro lado, el tiempo dedicado a ver transmisiones en vivo de videojuegos también ha aumentado significativamente, sobre todo en las nuevas generaciones. Streamers populares como Ninja y Rubius atraen a millones de personas a sus transmisiones de juegos como Fortnite y Minecraft en plataformas como YouTube y Twitch. Además, los deportes electrónicos, conocidos como eSports, han ganado una gran base de espectadores jóvenes, con más de 530 millones de espectadores en todo el mundo en 2022.

En cuanto al streaming de música, este sigue en auge constante con más de 580 millones de usuarios de pago en todo el mundo. Esto ha llevado a un crecimiento significativo en los ingresos generados por esta industria, superando los 17,000 millones de dólares en 2022. Aunque existen varias plataformas de música en streaming, Spotify ha logrado consolidarse como el líder del mercado, con más del 30% de los suscriptores a nivel global, duplicando las cifras de sus competidores más cercanos, Amazon Music y Apple Music, con 14.6 y 7 millones de usuarios en el último año, respectivamente. Otro aspecto en crecimiento dentro de la industria del streaming de audio es el podcast. El número de oyentes de podcasts en línea ha alcanzado los 640 millones a nivel mundial en 2022, casi triplicando la cifra de hace cuatro años. Esto ha llevado a un aumento en el valor económico de este mercado, proyectando un tamaño cercano a los 156,100 millones de dólares para 2030 (Orús, 2023).

En resumen, el streaming de contenido multimedia ha transformado la forma en que accedemos y disfrutamos de música, películas, programas de televisión y videojuegos. Las plataformas de streaming han tenido un impacto significativo en la industria del entretenimiento, ofreciendo una amplia gama de opciones y generando un crecimiento económico considerable. Con el continuo avance de la tecnología y el aumento en la adopción de servicios de streaming, está más que claro que esta tendencia se mantendrá en el futuro, ofreciendo aún más opciones y experiencias de entretenimiento personalizadas.

Con el aumento de la oferta en cualquier tipo de contenido multimedia del que hablemos, los usuarios adquieren cada vez unas expectativas más altas y quieren que el contenido se ajuste a sus gustos y esté disponible en cualquier momento y en cualquier lugar. En este sentido, las compañías se ven obligadas a invertir en tecnología para tratar de no quedarse atrás respecto a la competencia. Avances como el 5G, el crecimiento en la capacidad de cómputo de los dispositivos y los impresionantes avances en Inteligencia Artificial y Machine Learning, han creado un escenario propicio para que las empresas desarrollen soluciones que no solo les permitan mantenerse en el mercado, sino tambien generar nuevas oportunidades de negocio (Minhondo, 2023). En este trabajo llevaremos a cabo un estudio práctico del Machine Learning, el cual se utiliza en una amplia variedad de aplicaciones tecnológicas donde se busca dar el mejor servicio al usuario. La capacidad del Machine Learning para aprender y mejorar automáticamente a partir de los datos generados en sus aplicaciones, lo convierte en una herramienta muy poderosa para procesar grandes cantidades de información y automatizar tareas complejas. Cada vez que un usuario reproduce una canción, ve una película o escucha un podcast, se generan datos valiosos que pueden ser utilizados para comprender los gustos, preferencias y comportamientos del cliente.

1.2. Estado del arte

Las empresas de contenido multimedia utilizan una variedad de modelos de Machine Learning para segmentar a los clientes y brindarles contenido con más posibilidades de ser consumido en el momento y lugar adecuados. Además, la calidad del servicio es fundamental. Las infraestructuras y arquitecturas para la distribución de contenido suelen ser complejas y dependen de varios componentes interrelacionados. Esto implica una serie de posibles puntos de fallo, pero gracias a los avances tecnológicos actuales, podemos ir más allá del monitoreo tradicional y utilizar técnicas de Machine Learning para anticiparnos a posibles errores.

En el caso de la plataforma en la que centramos el estudio en este trabajo, el Machine Learning juega un papel fundamental en todos los aspectos del negocio de Spotify, el cual se encuentra en constante evolución en las metodologías de aprendizaje automático que van apareciendo en los últimos tiempos. El uso que realiza del Machine Learning permite mejorar la experiencia del oyente, estableciendo una conexión personalizada, actuando como asesor personal de música proporcionando listas de sugerencias diarias, destacando las canciones favoritas semanales y ofreciendo un resumen anual de su actividad (Knees & Schedl, 2013). Spotify, al igual que Netflix, no se basa únicamente en un modelo de recomendación, sino que combina diversas técnicas exitosas utilizadas previamente por otras herramientas para crear su propio sistema de recomendación único y poderoso. Esto da lugar a lo que podríamos llamar un “potente motor de descubrimiento” en lugar de un simple “recomendador”. En este sistema se utilizan tres modelos de recomendación en conjunto (Ciocca, 2017):

- **Modelos de filtrado colaborativo**. Este modelo analiza tanto el comportamiento del usuario como el de usuarios relacionados. A diferencia de Netflix, que utiliza calificaciones de estrellas, Spotify utiliza datos implícitos como el número de escuchas y los comportamientos de los usuarios como guardar una canción o visitar el perfil del artista. Utilizando algoritmos matemáticos y técnicas de factorización de matrices, Spotify compara los gustos musicales de los usuarios y las características de las canciones para encontrar similitudes y sugerir pistar que puedan ser de interés.
- **Modelos de procesamiento de lenguaje natural (NLP)**. Este enfoque utiliza datos de texto como metadatos de canciones, artículos de noticias y blogs para comprender las opiniones y descripciones de artistas y canciones. Spotify rastrea constantemente la web en busca de información relevante y analiza el lenguaje utilizado. Utilizando estos datos, se crean "vectores culturales" o "términos principales" que representan las caracteristicas musicales y se utilizan para determinar la similitud entre canciones.
- **Modelos de audio**. El tercer modelo de recomendación se basa en el análisis de datos de audio sin procesar. A diferencia de los modelos anteriores, este enfoque permite tener en cuenta canciones nuevas que aún no tienen mucha popularidad o menciones en Internet. Para analizar el audio, se utilizan redes neuronales convolucionales que extraen caracteristicas claves de las canciones, como la clave musical, el modo, el tempo o el volumen (Van den Oord et al., 2013). Estas caracteristicas ayudan a comprender las similitudes entre las canciones y permiten recomendarlas a usuarios con preferencias similares.

Entre los enfoques de mayor interés para llevar a cabo estas funciones mencionadas, se encuentran el aprendizaje por refuerzo, la inferencia aproximada, los modelos gráficos, la inferencia causal, el aprendizaje profundo, el modelado de series temporales y el aprendizaje de metamodelos (Spotify Research, n.d.). Estas técnicas han sido estudiadas a lo largo de los años por diversos autores, permitiendo asi desgranar los mecanismos para entender mejor el funcionamiento de este tipo de plataformas. Anteriormente, la generación automática de listas de reproducción, como forma de recomendación musical, se estudió intensamente (Aucouturier & Pachet, 2002). En cuanto a predecir el éxito de una canción, tema a tratar en este trabajo, se ha estudiado previamente a través de diversos enfoques. Uno de los enfoques más relevantes fue utilizar las redes sociales para evaluar la percepción pública, por ejemplo, usando Twitter para analizar publicaciones (tweets) asociadas a las etiquetas *#nowplaying*, *#np* (versión abreviada) o *#itunes* (la plataforma musical de Apple). El objetivo era predecir si una canción tendría éxito como para aparecer en el ranking Billboard Hot 100, en función de lo comentada que fuese en la red social. Este estudio logró una precisión del 0.90 a través de un Random Forest (Kim et al., 2014). Tambien con Twitter, se trabajó en un análisis de sentimiento, recopilando tweets con menciones a álbumes populares de 2016 y 2017, para verificar si las menciones de los tweets, ya fuesen positivas o negativas, tenían correlación con la popularidad en Spotify. Resultó ser efectiva tan solo para canciones con reseñas positivas (Araujo et al., 2017).

Si nos acercamos aún más al enfoque que llevaremos en nuestro trabajo, buscando la popularidad de la canción a través de caracteristicas acústicas que nos proporciona la propia página musical, son varios los estudios que se han realizado al respecto, como predecir las semanas que una canción se mantenía en una playlist de éxitos. En este caso, se logró una precisión del 0.55, y posteriormente consiguieron aumentar dicha precisión al 0.59 añadiendo información relacionada con el artista, como apariciones anteriores del artista en la playlist (Lee & Lee, 2015). En otro estudio, se extrajeron de Spotify, Billboard y Last.fm, un conjunto de datos que contaba con más de 9000 canciones que habían aparecido en rankings de popularidad entre 2013 y 2014. Además, tambien se añadieron todas las canciones de los álbumes en los que se lanzaron las canciones populares, aumentando la cifra a 23300 canciones. El objetivo era predecir cuál sería la canción más exitosa de un álbum nuevo. Los autores emplearon dos modelos de datos temporales, una red clasificadora autorregresiva no lineal (NAR) y su variante con entradas exógenas (NARX), obteniendo una precisión de 0.46 y 0.52, respectivamente (Karydis et al., 2018).

2. Objetivos

El objetivo principal de este proyecto es profundizar en las técnicas de Machine Learning para encontrar el mejor algoritmo de predicción de la popularidad de una canción de Spotify. Para ello, se plantean una serie de preguntas que sirvan como motivación del planteamiento de los objetivos específicos. Sabemos que el carácter subjetivo es un factor muy importante en el usuario para que una canción alcance un cierto grado de popularidad. La primera pregunta es si podremos encontrar un patrón más objetivo a partir de ciertas características musicales, que en el caso de que se cumplan, una canción se convierta en popular. La segunda pregunta que surge es si será posible aplicar diversas técnicas de Machine Learning que den lugar a modelos que permitan acercarse lo máximo posible al valor real de popularidad que posee cierta canción.

Para responder a estas preguntas y por tanto, lograr ese objetivo principal, se deberán alcanzar una serie de objetivos específicos, que son los siguientes:

- Construir y depurar la base de datos para un correcto estudio, eliminando aquellas variables que supongan un aumento de dimensionalidad y creando nuevas a partir de otras.
- Identificar las variables que más afectan a la popularidad de una canción a través de un análisis exploratorio exhaustivo.
- Evaluar el poder predictivo que tienen las variables estudiadas.
- Aplicar técnicas de Machine Learning para determinar un modelo que logre predecir lo máximo posible la popularidad de una canción.
- Analizar y comparar los diferentes modelos obtenidos para evaluar su capacidad predictiva y aportar conocimiento fruto de los objetivos alcanzados.

Con este estudio se pretende conocer desde dentro los factores más importantes que determinan que una canción pueda llegar ser más o menos exitosa, algo que puede servir de ayuda a cantantes y productoras musicales para sus futuros proyectos.

3. Metodología empleada

3.1. Metodología SEMMA

La metodología utilizada para este proyecto será la metodología SEMMA. Esta metodología, propuesta por el SAS Institute, es definida como el proceso de selección, exploración y modelado de grandes volúmenes de datos para descubrir patrones de negocio desconocidos (Rodríguez Montequín et al., 2003). El nombre de esta terminología es el acrónimo correspondiente a las cinco fases del proceso, las cuales detallamos a continuación:

- ***Sample*** **(Muestreo)**

El proceso se inicia con la extracción de los datos con los que se realizará el estudio, y si es necesario, extraer una muestra representativa de la población sobre la que se va a aplicar el análisis con el fin de reducir la dimensión de los datos y optimizar el proceso de modelización. Esta muestra debe ser representativa para asegurar la validez del modelo.

- ***Explore*** **(Exploración)**

Una vez obtenido el conjunto de muestra representativo, se debe realizar una exploración de la información disponible para poder entender y simplificar el problema, para de esta manera, lograr una mejor eficiencia del modelo. Esto se logra mediante herramientas de visualización y técnicas estadísticas donde veremos la importancia de las variables y la relación entre ellas, para poder determinar posteriormente que variables serán las que se utilizarán en el proceso de modelado.

- ***Modify*** **(Trasformación)**

La tercera fase implica la manipulación de los datos, en base a la exploración realizada en el paso anterior, para definir y dar un formato adecuado a los datos que se introducirán en el modelo.

- ***Model*** **(Modelización)**

Tras dejar lista la base de datos se procede al modelado de los datos. Aquí se utilizan técnicas y herramientas estadísticas o de aprendizaje automáticas con la finalidad de establecer una relación entre las variables explicativas y la variable objetivo de estudio y de esta manera, lograr una predicción adecuada de la variable objetivo. Las técnicas de aprendizaje automático empleadas en el estudio serán desarrolladas en el siguiente apartado.

- ***Assess*** **(Evaluación)**

Finalmente, la última fase de este procedimiento implica evaluar los resultados mediante el análisis de la calidad de los modelos ejecutados, en comparación con otros métodos estadísticos o con nuevas muestras de validación o test, si fuese necesario. Más tarde expondremos los métodos y medidas de evaluación elegidos para este trabajo.

3.2. Modelos empleados en el estudio

En este capítulo estudiaremos teóricamente los modelos utilizados en la correspondiente fase de Modelización de la metodología SEMMA. Como nuestro objetivo está en la predicción de la popularidad de una canción de Spotify, la variable objetivo es una variable continua, por lo que el modelo será de regresión. Tambien, se trata de un aprendizaje supervisado, ya que se entrena el modelo con datos de entrada asociados a datos de salida para buscar patrones de comportamiento.

Por tanto, los siguientes modelos de regresión han sido utilizados para la predicción de la popularidad de una canción.

3.2.1. Selección de variables

Antes de aplicar todos los modelos que hemos elegido para este trabajo, se realizará una selección de variables previa por diversas razones que nombramos a continuación (Acuña Collazos et al., 2012):

- Predecir con mayor precisión al eliminar variables sin relevancia y de esta manera evitar el sobreajuste.
- Describir un conjunto de datos con parsimonia, es decir, ajustar bien los datos con la menor cantidad de variables posibles.
- Estimar los coeficientes de regresión con errores estándar pequeños, sobre todo cuando algunas variables predictoras están altamente correlacionadas.
- Emplear un menor conjunto de variables predictoras para reducir el esfuerzo computacional durante todo el trabajo.

Crearemos varios subconjuntos de variables en función de dos factores. Entre todos los subconjuntos creados, nos quedaremos con el mejor de ellos aplicándole el primer modelo que tenemos, la regresión lineal. El primer factor de selección de variables son los métodos heurísticos, que son los tres siguientes:

- **Forward (Selección hacia adelante)**. Este método se basa en partir de un modelo inicial con una sola variable e ir incluyendo el resto una a una, en cada iteración del algoritmo. Tiene la ventaja de requerir menos capacidad de cálculo al comenzar con pocas variables pero por otro lado, no es capaz de eliminar variables cuando al incluir otras, estas se vuelven innecesarias.
- **Backward (Selección hacia atrás)**. En este método se hace el proceso contrario. Se comienza con todas las variables explicativas y van eliminándose una a una. Es un buen método para evitar que una variable significativa quede excluida del modelo, pero requiere de una gran capacidad de cálculo al trabajar con todas las variables desde el principio. Además, suele dar problemas de colinealidad, ya que muchas de las variables estén fuertemente correlacionadas.
- **Stepwise (Selección paso a paso)**. El último método es una combinación de los dos anteriores, asi, evita los inconvenientes de la selección hacia adelante y no requiere de una capacidad de cálculo tan grande como la de la selección hacia atrás. En cada

paso se contrasta si entra una variable explicativa o sale una que ya esté en el modelo. Es el modelo más utilizado y que mejores resultados aporta.

El segundo factor que combinaremos con los métodos de selección mencionados son los criterios de información. Estos criterios nos ofrecen una medida que balancea la calidad del modelo frente al número de predictores empleados (Álvarez Liébana, 2022):

- **AIC (Criterio de Información de Akaike)**. Este criterio es un estimador muestral de la esperanza de la log-verosimilitud, es decir, como de probable es que, si el modelo fuese cierto, hayamos obtenido los resultados que hemos obtenido. El AIC se define mediante la siguiente función:

$$AIC(k) = -2\ ln\mathcal{L}\left[\hat{\theta}(k)\right] + 2k$$

  donde $\mathcal{L}\left[\hat{\theta}(k)\right]$ es la función de máxima verosimilitud de las observaciones, $\hat{\theta}(k)$ la estimación máximo verosímil del vector de parámetros θ y k el nº de parámetros independientes estimados dentro del modelo.

- **BIC (Criterio de Información Bayesiano)**. Este otro criterio, propuesto Schwarz (1978), argumentando que el AIC no es justificable asintóticamente, y es por ello por lo que presentó una modificación de AIC, donde en BIC se penaliza el número de parámetros con $ln\, n$, en lugar de 2, por lo que cuando crece n. BIC penaliza más el sobreajuste. Su función es la siguiente:

$$BIC(k, n) = -2\ ln\mathcal{L}\left[\hat{\theta}(k)\right] + k\ ln\, n$$

  donde $\mathcal{L}\left[\hat{\theta}(k)\right]$ es la función de máxima verosimilitud de las observaciones, $\hat{\theta}(k)$ la estimación máximo verosímil del vector de parámetros θ y k el nº de parámetros independientes estimados dentro del modelo, mientras n es el tamaño muestral.

3.2.2. Regresión lineal

El primer modelo que estudiaremos será la regresión lineal. La regresión lineal es un método estadístico que intenta modelizar la relación entre dos o más variables interpretativas (independientes) y una variable de respuesta (dependiente) ajustando una ecuación lineal a los datos observados (Nasteski, 2017). La función de la regresión lineal es la siguiente:

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

donde y es la variable dependiente, β_n son los parámetros que miden la influencia de las variables independientes sobre la variable dependiente, x_n son las variables independientes y ε representa el residuo o error cometido en la estimación.

Como se muestra en la figura siguiente, el modelo (línea roja) se calcula utilizando datos de entrenamiento (puntos azules) en los que cada punto tiene una etiqueta conocida (eje Y) para ajustarse a los puntos con la mayor precisión posible minimizando lo máximo posible el error.

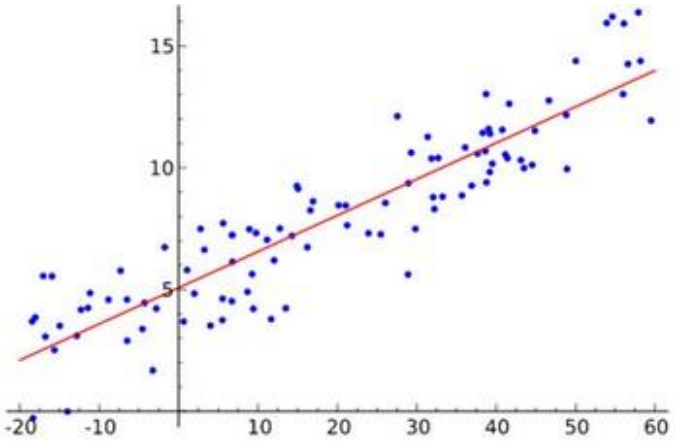


Figura 1. Representación visual de una regresión lineal (Nasteski, 2017)

El objetivo de la regresión lineal es encontrar la relación y dependencia entre las variables del conjunto con respecto a la variable objetivo, buscando ajustarse a los puntos de observación minimizando el error. Es un algoritmo robusto, rápido y útil cuando la relación entre las variables objetivo y las independientes no es demasiado compleja. Además, es menos propenso a caer en sobreajuste.

3.2.3. Redes neuronales

Las redes neuronales es un modelo computacional biológico inspirado en las funciones del cerebro humano. Suelen constar de tres capas: una capa de entrada, una o varias capas ocultas (neuronas) y una capa de salida. Durante el entrenamiento de la red, se extraen las relaciones entre las capas de entrada y salida a partir de los datos. Se realiza un mapeo de entrada-salida utilizando una serie de procesamientos interconectados en la capa oculta. Cada neurona en la capa oculta recibe señales externas u otras neuronas y las procesa a través de una función de activación transferible. Los datos se procesan desde la entrada hacia la salida a través de la capa oculta en lo que se conoce como alimentación directa (Al-Mukhtar, 2019).

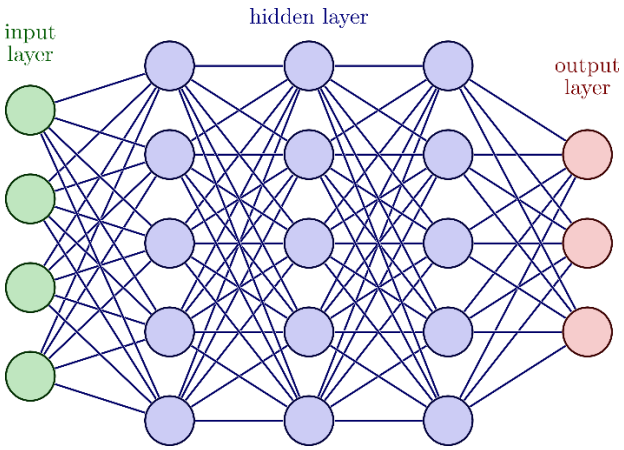


Figura 2. Estructura de una red neuronal (Neutelings, 2021)

Cada neurona calcula una suma ponderada de sus entradas para aplicarle una función de activación. Esto permite resolver problemas complejos que surgen de la no linealidad entre las variables. La información procesada por una neurona se transmite a la siguiente neurona, repitiendo el mismo proceso. Los pesos de las conexiones entre las neuronas, que actúan como parámetros a estimas, determinan la conectividad de la red, de manera similar a como ocurre en la regresión lineal. Una ventaja con respecto a la regresión es que las redes son muy útiles para encontrar relaciones no lineales entre las variables. Además, cuanto más variables independientes haya, mejor funcionará. Con las redes neuronales podremos modificar diversos parámetros estudiando qué opción es la mejor para encontrar el modelo óptimo con el que minimizar el error. Dichos parámetros son los siguientes:

- ***size***: número de nodos, es decir, las unidades que se encuentran en la capa oculta.
- ***decay***: parámetro que ajusta el algoritmo de optimización controlando la velocidad a la que se mueve dicho algoritmo hasta llegar al mínimo de la función de pérdida de error.

3.2.4. Bagging

El primero de los cuatro modelos que vamos a probar basados en árboles es el Bagging. Por dar una pequeña introducción, los árboles son un tipo de modelización no paramétrica utilizada tanto para la clasificación como para la regresión de datos, que son muy intuitivos y que se pueden representar gráficamente, lo que facilita su comprensión. La construcción de un árbol consiste en dividir los datos en subconjuntos más pequeños, donde se ajusta un valor predicho diferente en cada uno. Estos subconjuntos se dividen nuevamente en otros dos nodos en función del valor de una variable independiente, y asi hasta que se cumple un criterio de parada, como el número máximo de nodos terminales o el mínimo número de observaciones en cada nodo terminal (Breiman, 1984). Aunque tienen ventajas como su capacidad para tratar variables categóricas y su robustez, también presentan limitaciones en cuanto a su poder predictivo y su tendencia al sobreajuste. Los nodos terminales se conocen como hojas, mientras que los nodos que no son la raíz ni las hojas se llaman nodos internos.

Dada este pequeña introducción y funcionamiento de los árboles de decisión, pasamos a analizar más en detalle el Bagging. El Bagging consiste en ajustar varios árboles de decisión utilizando diferentes muestras del conjunto de entrenamiento para después tomar la clase más frecuente como predicción final para un nuevo individuo. Los árboles en el Bagging no se podan, ya que el objetivo está en reducir el sesgo en cada árbol y no se considera un problema si la variabilidad aumenta (Breiman, 1996). Sin embargo, el Bagging tiene tambien algunas desventajas. Los árboles generados pueden ser similares entre sí, especialmente cuando existen variables cuya importancia a la hora de clasificar son muy superiores a otras. Y esto ace que las predicciones estén muy correlacionadas y hacer que el modelo se ralentice a la hora de reducir la variabilidad de las predicciones.

3.2.5. Random Forest

El Bagging, aunque introduce aleatoriedad al seleccionar observaciones para cada árbol, no logra que los árboles sean independientes debido a la similitud en las variables utilizadas y la estructura. Para abordar esto, se desarrolló el método Random Forest (Breiman, 2001). En este método, se toman N submuestras aleatorias con reemplazamiento del conjunto de datos original, lo que aumenta la variabilidad en el modelo, garantiza la independencia de los árboles y reduce el riesgo de sobreajuste. Esto se debe a que al promediar las predicciones de los diferentes árboles, el modelo se mantiene estable al enfrentar nuevas observaciones. Para reducir la correlación entre los árboles, en cada división se introduce aleatoriedad en la selección de variables predictoras. Todo esto tambien tiene sus desventajas, y es que el Random Forest resulta más difícil de interpretar que el Bagging y más complejo, ya que requiere estimar un parámetro adicional, el tamaño de la muestra. Los parámetros que se ajustan tanto para los modelos de Bagging como de Random Forest son los siguientes (Portela García-Miguel, 2023):

- ***mtry***: número de variables que se sortean en cada división del árbol. En el caso del Bagging ponemos el número total de variables independientes ya que en este modelo no se podan. En cambio, para el Random Forest se probarán distintos números de variables buscando el más adecuado.
- ***ntrees***: número de árboles que se van a promediar, es decir, número de iteraciones del modelo.
- ***nodesize***: número de observaciones que habrá como mínimo en una rama-nodo, es decir, el tamaño mínimo de la hoja. Este parámetro mide la complejidad. Se puede ampliar para evitar sobreajuste (reducir varianza) o reducir para ajustar mejor (reducir sesgo).
- ***sampsize***: tamaño de cada muestra. Este parámetro solamente está para el Random Forest.

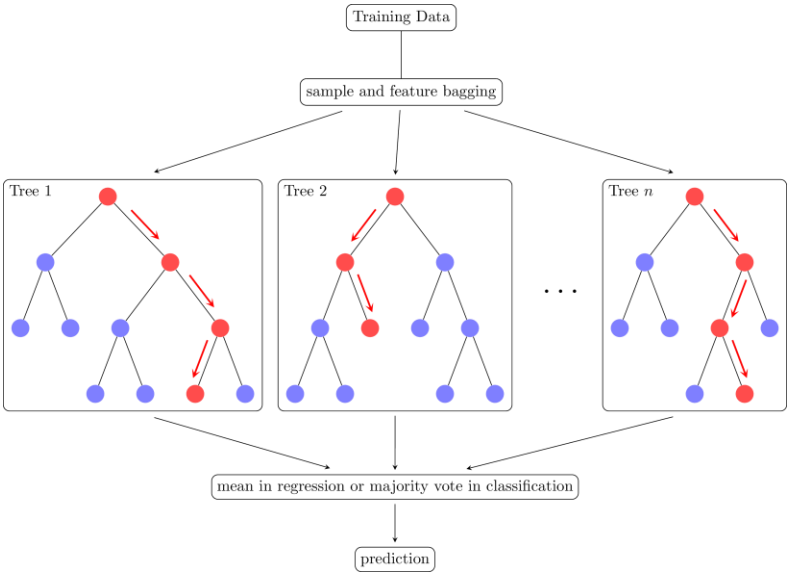


Figura 3. Representación del Random Forest (Riebesell et al., 2023)

3.2.6. Gradient Boosting

El Gradient Boosting o potenciación del gradiente surge a partir de la idea de que un modelo débil puede modificarse de forma que se convierta en un modelo mejor (Friedman, 1999). El objetivo de este algoritmo está en convertir hipótesis relativamente débiles en hipótesis realmente buenas. Suele ser popularmente utilizado para problemas de clasificación binaria. No obstante, será estudiado y tuneado con nuestro problema de la misma manera que los modelos expuestos hasta ahora, con la esperanza de obtener un resultado susceptible de utilización o por el contrario, confirmar su pérdida de efectividad en problemas de predicción con variable continua.

Este modelo trata de construir un modelo paso a paso que minimiza los valores residuales generados en las iteraciones previas, obteniendo asi de manera gradiente modelos que convergen en un modelo final donde los errores son mínimos. Como ventajas de este modelo está su buen tratamiento de valores ausentes y variables categóricas, su gran poder de predicción y su robustez con variables poco importantes, detectando interacciones ocultas. Sin embargo, en el caso de tratar con bases de datos más simples con pocas variables y linealidad, no aporta mejorías respecto a modelos más simples (Portela García-Miguel, 2023). El Gradient Boosting es conocido por tener una gran cantidad de hiperparámetros que optimizar, pero los que nosotros tunearemos son los siguientes:

- ***shrinkage***: parámetro de regularización para corregir posibles estimaciones erróneas. Mide la velocidad de ajuste o tasa de aprendizaje. A menor parámetro, más lento y necesita más iteraciones, pero es más fino en el ajuste.
- ***n.minobsinnode***: tamaño mínimo de nodos finales, es decir, número mínimo de observaciones que debe tener cada hoja. Es el parámetro que mide la complejidad.
- ***n.trees***: número de árboles (iteraciones).
- ***interaction.depth***: profundidad máxima de cada árbol.

3.2.7. Extreme Gradient Boosting

El algoritmo Extreme Gradient Boosting, más conocido como XGBoost, es una variante más actual del Gradient Boosting surgida de la plataforma Kaggle (Brownlee, 2016). En este caso, se construyen los árboles de decisión de forma secuencial para corregir los errores de los árboles existentes utilizando el descenso de gradiente para minimizar la pérdida en la agregación de los nuevos árboles. Es una alternativa más eficiente y de implementación más escalable por su mayor rango de trabajo con la regularización del modelo a la hora de controlar el sobreajuste, logrando así un mejor rendimiento. XGBoost está altamente enfocado en lograr una mayor velocidad computacional y en un mejor rendimiento del modelo, sin embargo, no siempre tiene por qué ser mejor que el Gradient Boosting. Para el tuneo del XGBoost contamos con 7 parámetros. Comentar que algunos de ellos se dejarán predeterminados con un valor por defecto, cuyos motivos explicamos a continuación:

- ***min_child_weight***: número de observaciones mínimas en el nodo final.
- ***eta***: velocidad de ajuste (*shrinkage*).
- ***nrounds***: número de iteraciones.
- ***max_depth***: profundidad máxima de los árboles. A partir de este parámetro trabajaremos con valores predeterminados. En este caso, se suele establecer por defecto un valor de 6 ya que un valor menor puede causar modelos subajustados
- ***gamma***: parámetro que especifica la reducción mínima requerida en la función de pérdida para que se realice una partición adicional en un nodo del árbol. Dejando *gamma* en 0, se permite cualquier división que reduzca la pérdida, lo que puede conducir a un modelo más complejo y propenso al sobreajuste. Sin embargo, como en nuestro caso de estudio tendremos suficientes datos de entrenamiento, lo mejor es dejar *gamma* en 0 y permitir que el modelo aprenda automáticamente la importancia de realizar divisiones adicionales.
- ***colsample_bytree***: porcentaje de sorteo de variables antes de cada árbol, al estilo del Random Forest pero antes del árbol, no en cada nodo.
- ***subsample***: porcentaje de sorteo de observaciones antes de cada árbol, al estilo del Random Forest pero antes del árbol, no en cada nodo.

En estos dos últimos parámetros vistos, el valor predeterminado es de 1 que indica que se utiliza el conjunto completo de datos para cada árbol. Reducir estos valores introduciría aleatoriedad en la selección de variables y observaciones, lo que puede ayudar a reducir el sobreajuste y mejorar la generalización del modelo, pero con un conjunto tan amplio como el nuestro, no es necesario reducir estos parámetros.

En el siguiente gráfico, para cerrar el estudio de los algoritmos basados en árboles, se muestra la evolución de ellos a lo largo de los años, desde los árboles de decisión como tal, hasta el XGBoost:

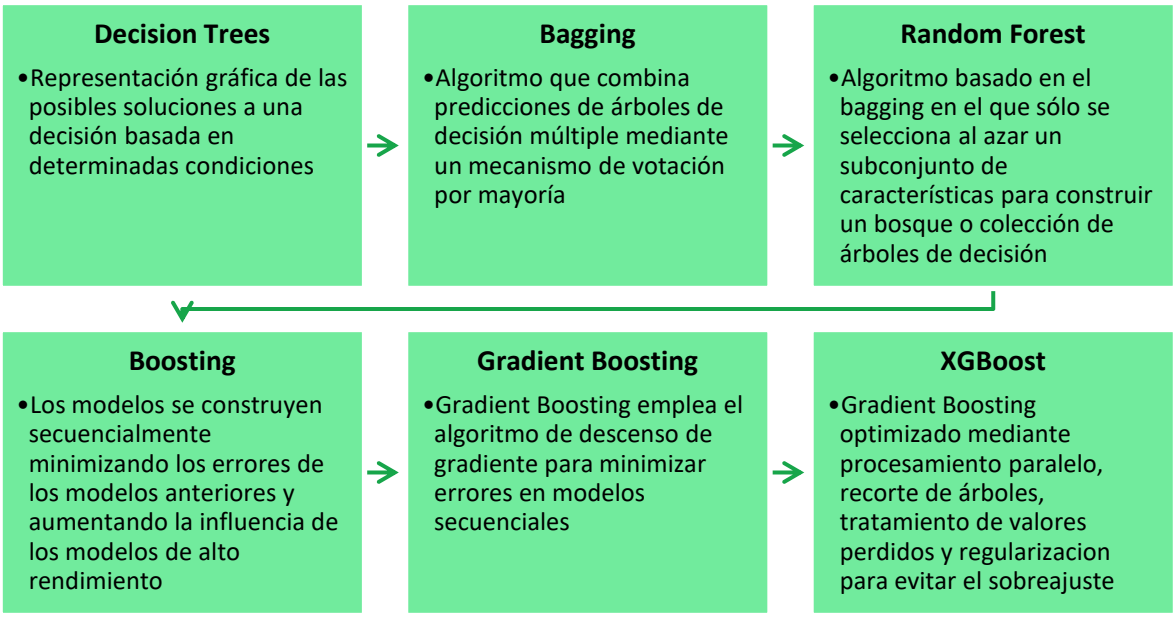


Figura 4. Evolución de los algoritmos de árboles (Morde, 2019)

3.2.8. Support Vector Machine

Support Vector Machine (SVM) es un algoritmo de aprendizaje supervisado cuyo objetivo es encontrar un hiperplano con el margen más amplio posible que separe de la mejor forma dos clases diferentes de puntos de datos (Gandhi, 2018). Los vectores de soporte son un subconjunto de las observaciones de entrenamiento que identifican la ubicación del hiperplano de separación. El margen se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores. Como en la práctica esto tiende a ser muy complicado, el algoritmo maximiza el margen flexible permitiendo un pequeño número de clasificaciones erróneas.

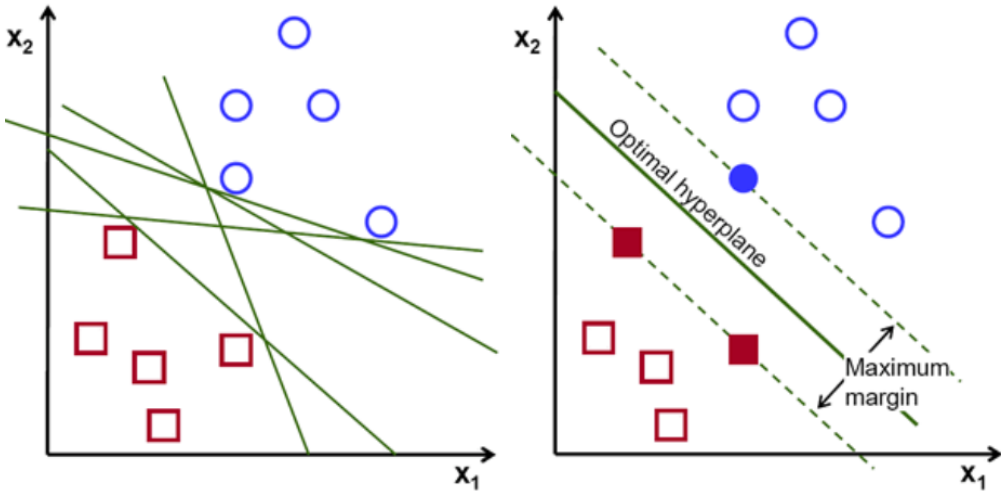


Figura 5. Definición del margen entre clases en el SVM (Gandhi, 2018)

El SVM utiliza funciones de kernel para transformar las características de los datos a un especio multidimensional diferente, con la expectativa de que resulte más fácil separar las clases. Esta transformación simplifica los límites de decisión complejos no lineales y los hace lineales en el espacio dimensional de características superior asignado. Esto se conoce como el truco de kernel y evita la necesidad de transformar explícitamente los datos (Boser et al., 1992). También, hace que los SVM sean más flexibles y capaces de gestionar problemas no lineales. Para trabajar aplicando esta idea, se puede hacer uso de distintas funciones kernel, que harán referencia a un tipo de SVM distinto:

Tipo de SVM	Función kernel	Descripción
Lineal	$K(x_i \cdot x_j) = x_i^T \cdot x_j$	Aprendizaje de dos clases.
Polinomial	$K(x_i \cdot x_j) = (x_i^T x_j + 1)^{\rho}$	ρ representa el orden del polinomio.
Radial (RBF) o Gaussiana	$K(x_i \cdot x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$	Aprendizaje de una clase. σ representa la anchura del kernel.

Tabla 1. Funciones kernel según el tipo de SVM (Boser et al., 1992)

3.2.9. Técnicas de ensamblado

Las técnicas de ensamblado o ensemble consisten en la predicción de la variable dependiente partiendo de la unión de varios modelos. La idea consiste en combinar las predicciones de los mejores modelos obtenidos durante el estudio para conseguir así un resultado más favorable y sobre todo más robusto. Las técnicas básicas de ensamblado son Bagging, Boosting, Stacking, entre otros. Para este proyecto, utilizaremos el método Stacking, que permite obtener la predicción a través de tres formas:

- **Promedio**. Se calcula el promedio de las predicciones de los modelos que se van a combinar. Utilizaremos esta opción en nuestro estudio.
- **Voto**. Predice el resultado con mayoría sobre el total. Esta opción se utiliza en problemas de clasificación.
- **Combinación de un algoritmo diferente**. Las predicciones que se han obtenido de diferentes algoritmos, utilizarlas como variables independientes en otro algoritmo.

Las ventajas de realizar ensamblado de modelos es que los modelos resultantes son bastante robustos y reducen generalmente la varianza, casi nunca empeoran a los modelos originales. En cambio, se trata de modelos complejos que resultan más difíciles de interpretar que los diferentes modelos por separado.

3.3. Métodos de evaluación de modelos

3.3.1. Validación cruzada

Como método de evaluación de los distintos modelos anteriormente estudiados, con el propósito de que el azar no influya en los resultados obtenido por probarlos una sola vez, se utilizan técnicas de remuestreo donde se usen varios conjuntos de datos. En nuestro caso, como el conjunto de datos con el que trabajaremos es bastante grande, podríamos haber trabajado con una partición train-test, recomendado para conjuntos de datos de gran tamaño (Álvarez Liébana, 2022). En cambio, con la finalidad de poder obtener los mejores resultados posibles, se opta por utilizar las técnicas de validación cruzada simple y repetida.

Usaremos la técnica conocida como validación cruzada para tunear los parámetros de cada algoritmo y quedarnos con el mejor. Esta técnica consiste en dividir el conjunto de datos en submuestras y construir el modelo con todas las observaciones menos las de una submuestra y evaluarlo a continuación con las observaciones de dicha submuestra excluida. El problema de esta técnica es que habrá algunas muestras que se queden sin evaluar y otras que se evalúen más de una vez, es decir, las submuestras se pueden solapar (Laura-Ochoa, 2019). Como regla general en este trabajo, se realizarán 4 particiones de los datos en todos los modelos.

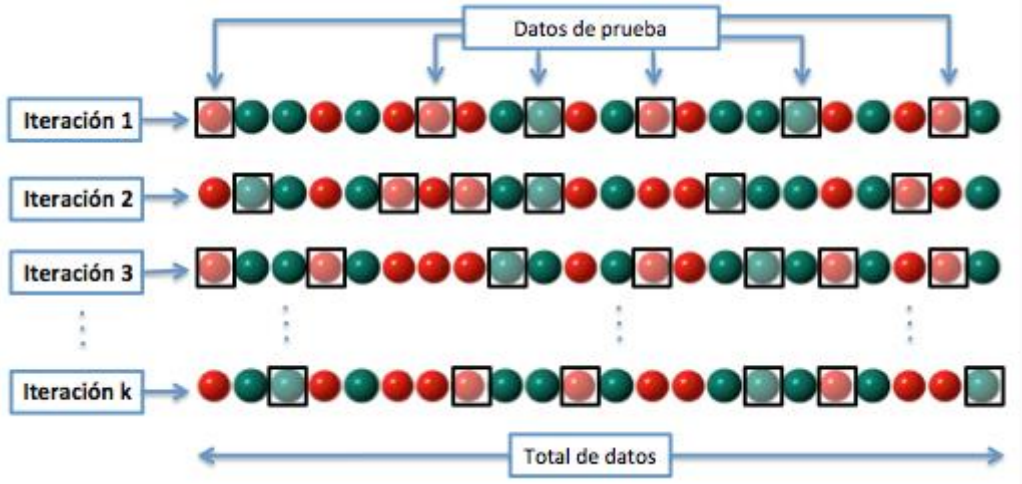


Figura 6. Validación cruzada simple (Álvarez Liébana, 2022)

Tambien se usará la validación cruzada repetida, en este caso para comparar a través de los diagramas de cajas un algoritmo final con el resto de algoritmos ejecutados. Esta técnica consiste en repetir este mismo proceso aleatoriamente las veces que se indique. En este caso, también se realizarán 4 particiones de datos y se repetirá 10 veces cada uno. Con estas dos técnicas, a pesar de que requiera una gran carga computacional al repetir el mismo proceso varias veces, lograremos un modelo más estable de cada algoritmo.

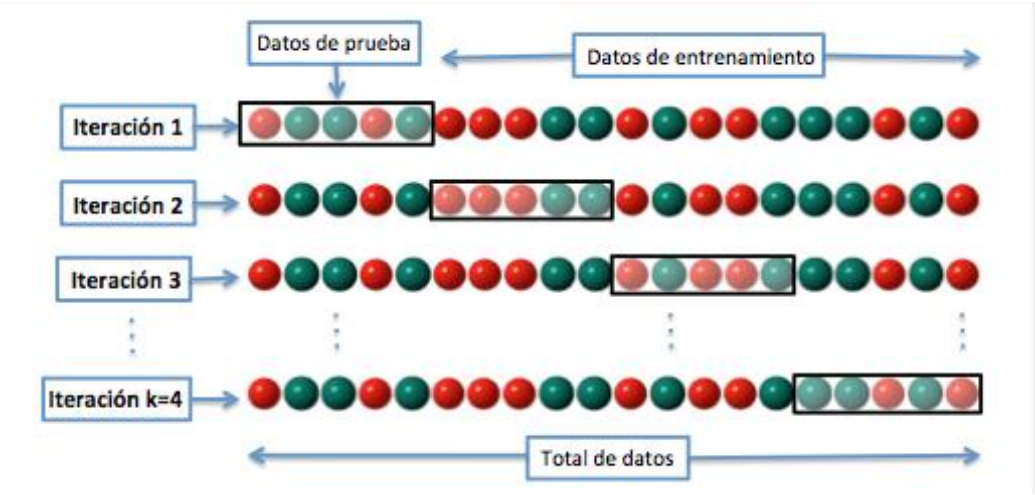


Figura 7. Validación cruzada repetida (Álvarez Liébana, 2022)

3.3.2. Medidas de evaluación

Como finalización de nuestro estudio, deberemos analizar la precisión de cada modelo ejecutado a través de unas medidas de evaluación. Las medidas de evaluación que se usarán en este proyecto serán las siguientes:

- **Error cuadrático medio (MSE)**. Representa el promedio de la diferencia cuadrática entre los valores reales y predichos en el conjunto de datos. Mide la varianza de los residuos. Cuanto menor sea el valor del MSE, mejor será el modelo. La fórmula es la siguiente:

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

  y_i e $\hat{y}_i$ son los valores reales y predichos de la variable objetivo, respectivamente.

- **Raíz del error cuadrático medio (RMSE)**. Raíz cuadrada del MSE. Mide la desviación estándar de los residuos. Se usará esta medida en lugar del MSE debido a su mayor facilidad de interpretación, ya que se encuentra expresado en las mismas unidades que la variable objetivo. Al igual que con el MSE, cuanto menor sea el valor, mejor será el modelo.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

- **Coeficiente de determinación (R^2)**. Representa la proporción de la varianza en la variable dependiente que es explicada por el modelo de regresión lineal. Es una puntuación sin escala, es decir, independientemente de que los valores sean pequeños o grandes, el valor de R^2 será menor que uno.

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

  SSE es el MSE multiplicado por el número de observaciones y SST es la suma de cuadrados de los errores que se cometen cuando se usa la media como valor de predicción para todas las observaciones. Cuando el valor es muy preciso, este coeficiente toma valores cercanos a 1.

- **Error absoluto medio (MAE)**. Representa el promedio de la diferencia absoluta entre los valores reales y predichos en el conjunto de datos. Mide el promedio de los residuos.

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

En el desarrollo de este trabajo se calcularán todas las medidas contempladas para cada modelo pero se valorarán tan solo el RMSE y el R^2, al ser las medidas más determinantes para la elección de un modelo. A partir de estas dos métricas se elegirán los mejores conjuntos de hiperparámetros para los modelos. No obstante, tambien se tendrá en cuenta otros factores en caso de semejanza entre algunos algoritmos con la intención de que la elección del mejor modelo no dependa de factores meramente objetivos. Dichos factores son los siguientes:

- **Simplicidad del modelo**. Cuanto más simple sea el modelo, mejor opción será para su elección. En el caso de hacer una selección de variables, aquel conjunto que cuente con menos variables será favorito frente a otro con un mayor número, en el caso de que sus medidas sean muy parecidas. Lo mismo ocurrirá a la hora de decantarnos por un modelo con varios parámetros a ajustar, como puede ser el caso de las redes, árboles o SVM. Cuanto más simple sean los parámetros de cada algoritmo, mejor será el modelo.

- **Variabilidad del modelo y capacidad predictiva**. Para comparar todos los modelos haremos uso de los diagramas de cajas y bigotes o boxplot, ya que son una herramienta útil para visualizar la variabilidad de los modelos, al analizar el sesgo y varianza de cada modelo.

    - El sesgo se refiere a la tendencia sistemática del modelo a predecir valores que se desvían de los valores reales. Un modelo sesgado puede deberse cuando el modelo es demasiado simple y no captura todas las complejidades de los datos, o cuando se seleccionan características irrelevantes o se omiten características importantes.
    - Respecto a la varianza, se refiere a la variabilidad de las predicciones cuando se entrena con diferentes conjuntos de datos. Un modelo con alta varianza puede ser sensible a las variaciones en los datos y tener dificultades en la predicción. Esto puede darse cuando el modelo es demasiado complejo o se sobreajusta a los datos de entrenamiento.

    En resumen, debemos encontrar el equilibrio entre ambos, seleccionando el modelo que tenga un sesgo bajo y una varianza moderada. Es por ello por lo que utilizaremos los diagramas de cajas para tomar las decisiones sobre qué modelo es más adecuado en términos de estabilidad y capacidad predictiva.

- **Interpretabilidad del modelo**. Cuando ya hayamos analizado todos los factores anteriores y sigamos teniendo modelos muy similares con el que quedarnos, recurriremos a la interpretabilidad de ellos, es decir, aquel modelo que tenga una mayor capacidad explicativa. Algoritmos como los ensamblados o las redes neuronales son denominados de caja negra, debido a que sus procesos y resultados son muy complejos de explicar a pesar de ser muy poderosos en la predicción. En cambio, la regresión lineal o los árboles son algoritmos de caja negra ya que son muy fáciles de entender, permiten definir los pesos de cada variable utilizada en el proceso y facilita la interpretación de los resultados. Dicho esto, optaremos siempre por un modelo de caja blanca en caso de encontrarnos con modelos muy parejos.

4. Construcción, exploración y depuración de la base de datos

A lo largo de este capítulo se va a explicar la obtención y construcción de la base de datos que hemos utilizado para este trabajo. A su vez, realizaremos un análisis descriptivo de las variables y una depuración para poder trabajar mejor con nuestros datos. Posteriormente, haremos una exploración más detallada de cada variable para analizar su distribución, y de esta manera, proceder a una correcta depuración de datos para tener una base de datos limpia y eficiente para empezar con la modelización de este trabajo.

4.1. Construcción de la base de datos

La base de datos utilizada para este trabajo ha sido extraída de la plataforma Kaggle (Shahane, 2022). Trata de pistas de audio extraídas de la plataforma musical de streaming Spotify con una gran cantidad de características musicales sacadas tanto de dicha plataforma, como tambien de la página web de música Genius, conocida por sus bases de datos de letras de canciones. Este conjunto de datos no solo cuenta con canciones, también puede ser radio hablada, podcasts o programas de noticias, todo sacado de la plataforma musical de streaming Spotify. La base de datos principal está construida a partir de cuatro dataset, aportándonos cuatro conjuntos de datos, los cuales son:

- **spotify_tracks**: pistas de reproducción de Spotify seleccionadas junto con sus características musicales (101.939 observaciones). Este será el dataset base.
- **spotify_artists**: artistas a los que pertenece las distintas pistas junto con sus características (56.129 observaciones).
- **spotify_albums**: álbumes a los que pertenecen las distintas pistas junto con sus características (75.511 observaciones).
- **lyrics_features**: letras de las pistas junto con sus características (94.954 observaciones). Este dataset es extraído desde Genius.

Antes de realizar el análisis descriptivo de las variables, vamos a descartar variables que vemos irrelevantes ya a simple vista, como pueden ser enlaces que llevan a la plataforma Spotify, columnas que vienen repetidas en varios dataset o enlaces a imágenes de portadas de discos o artistas.

También, renombraremos los nombres de las columnas para que sepamos su dataset de origen con el prefijo correspondiente: *track_*, *album_*, *artist_* o *lyric_*. Así, no crearemos columnas duplicadas como podría ser la variable *name* del artista y la variable *name* del track. Ya por último, uniremos los cuatro dataset en uno llamándolo *spotify_fusion*.

Nuestro conjunto de datos recoge 101.939 observaciones con 38 atributos (variables) de cada una de ellas. Las variables están agrupadas en cuatro categorías principales en base a la procedencia de su dataset de origen. A continuación realizamos un análisis descriptivo de cada variable:

Dataset de origen	Nombre de la variable	Descripción de la variable
spotify_tracks	track_id	ID de la pista proporcionado por Spotify
	track_acousticness	Medida de confianza sobre si la pista es acústica
	track_available_markets	Países en los que se ha escuchado la pista
	track_country	País de origen de la pista
	track_danceability	Combinación de elementos musicales para determinar la bailabilidad de una pista
	track_disc_number	Número del disco del artista en la que aparece la pista
	track_duration_ms	Duración de la pista en milisegundos
	track_energy	Medida perceptiva de intensidad y actividad
	track_instrumentalness	Predice si una pista no contiene voces y es instrumental
	track_key	Tonalidad global estimada de la pista
	track_liveness	Vitalidad de la pista
	track_loudness	Sonoridad general de una pista en decibelios (db)
	track_mode	Modalidad (mayor o menor) de una pista, el tipo de escala del que se deriva su contenido melódico
	track_lyrics	Letra de la pista
	track_name	Nombre de la pista
	track_playlist	Lista de reproducción de la que ha sido extraída la pista
	track_popularity	Popularidad de la pista
	track_speechiness	Nivel de locuacidad que tiene la pista
	track_tempo	Tempo global estimado de una pista en pulsaciones por minuto (BPM)
	track_time_signature	Compás musical de la pista
	track_number	Número de la pista en su álbum correspondiente
	track_valence	Medida que describe la positividad musical de una pista
spotify_artists	artist_popularity	Popularidad del artista
	artist_followers	Número se seguidores del artista
	artist_genres	Géneros de música en los que se cataloga al artista
	artist_id	ID del artista proporcionado por Spotify
	artist_name	Nombre del artista
spotify_albums	album_type	Tipo del álbum
	album_id	ID del álbum proporcionado por Spotify
	album_name	Nombre del álbum
	album_release_date	Fecha de lanzamiento del álbum
	album_tracks	Número de pistas del álbum
lyrics_features	lyric_mean_syllables_word	Media de sílabas por palabra en la pista
	lyric_mean_words_sentence	Media de palabras por frase en la pista
	lyric_n_sentences	Número de frases en la pista
	lyric_n_words	Número de palabras en la pista
	lyric_sentence_similarity	Similitud entre frases de la pista
	lyric_vocabulary_wealth	Nivel de riqueza del vocabulario

Tabla 2. Análisis descriptivo de las variables

4.2. Análisis exploratorio

A lo largo de este apartado se va a hacer una exploración minuciosa de cada variable de nuestro conjunto de datos para ver la distribución de la variable objetivo en cada una, descartar variables que no nos servirán para la modelización, al igual que crear variables nuevas a partir de otras, reagrupar categorías, pasar de variables categóricas a numéricas, y viceversa, y detectar valores atípicos que puedan perjudicar el resultado de modelos sensibles a estos. Toda esta exploración ha sido apoyada a través de gráficos realizados en R, los cuales se pueden consultar en el Anexo I.

4.2.1. Variables cualitativas

- **track_available_markets**

Esta variable es de tipo lista, donde en cada observación aparecen los países en los que se escucha la pista analizada. Si hacemos un *count*, aparecen las listas más repetidas que suelen ser los mismos países siempre. Esta variable tal cual viene representada no nos aporta información ya que aparecen más de 3000 listas distintas de países. Por ello, debemos transformarla en una nueva variable llamada **track_num_countries**, que nos dirá el número de países donde se escucha la pista, sumando de cada lista el número de países que contiene. Con esta nueva columna, vemos que el 66% de las pistas se escuchan en 79 países, y que más del 85% de las pistas se escuchan en más de 70 países. Si analizamos que media de popularidad existe en las pistas que son escuchadas en el mismo número de países, no hay mucha diferencia entre pistas que son escuchadas en pocos países y pistas que son escuchadas en los 79 países.

Para seguir sacándole partido a esta variable, podemos crear cinco variables nuevas que hagan referencia a los cinco continentes, y si una pista es escuchada en un país de dicho continente, en la nueva columna saldrá como 1, y si no, como 0. Creamos así 5 variables binarias: *market_Africa*, *market_America*, *market_Asia*, *market_Europa* y *market_Oceania*. Como era de esperar haciendo el primer análisis de esta variable, al menos el 90% de los registros con los que cuenta la base de datos se escuchan en cada continente. El continente con más pistas escuchadas es Europa con casi un 96% de ellas. En cuanto a popularidad, la media de esta en cada continente es idéntica, 40.

- **track_country**

En teoría esta variable tendría que ser el país de origen de la pista, pero observando las variables que arroja esta variable, solo nos da tres países que no corresponden con el origen de la pista correspondiente. Los países son Argentina (*AR*), Finlandia (*FI*) y Bélgica (*BE*). Quizás sea el país de donde fueron descargadas las pistas. Dejaremos la variable por si nos sirve para futuros estudios. Hay que destacar que el país con una mayor proporción de tracks es Argentina (49%), y también es donde sus tracks tienen una media de popularidad mayor (43).

- **track_playlist**

Listas de reproducción de las que provienen las pistas extraídas. Contamos con 3.754 playlists distintas donde ninguna alcanza un 1% de representatividad. Para poder trabajar con esta variable, deberemos recategorizarla. Crearemos una nueva variable llamada **playlist_mean** donde agrupamos las playlists en función de la media de popularidad que tienen todas sus pistas. De esta forma, pasamos de tener 3.754 categorías a tener solo 8. Cada categoría recoge un intervalo de 10, por ejemplo, *playlist_20* recoge las playlists que tienen una media de popularidad de 20 a 30. Añadir que playlists con más de 70 de media de popularidad es poco frecuente, por lo que las hemos agrupado en *playlist_70*. Si hacemos un count de la variable, vemos que las listas de reproducción más repetidas son *playlist_40*, *playlist_30* y *playlist_50*, con un 73.6% de los datos.

- **artist_genres**

Esta columna muestra los géneros en los que está catalogado el artista de la pista. Con esta variable nos ocurre lo mismo que con **track_available_markets**, son listas con las que es complicado trabajar y analizar, ya que aquí tenemos más de 12000 listas distintas, y una lista puede contener hasta 6 géneros distintos asociados al artista. Para poder sacarle provecho a esta variable, haremos lo mismo que hicimos en **track_available_markets**. En este caso, crearemos variables binarias con los géneros de música más escuchados en todos los tiempos y si una pista tiene ese género en su lista, se le pondrá un 1 a su variable correspondiente. Por ejemplo, si tenemos una pista que en la variable *artist_genres* presenta la siguiente lista: “ *['classic swedish indie', 'swedish jazz', 'swedish pop'] ”*, las nuevas variables binarias la catalogarán como una pista de indie, jazz y pop, siendo dichas variables marcadas con 1 y el resto con 0. Tras crear las distintas variables con los géneros más populares, analizamos ahora la proporción de ellos en nuestra base de datos. Un 31% de las pistas son catalogadas como pop, un 16% como rock y un 13% como indie. Por otro lado, el género con más popularidad es el rap con un 53 de media, siguiéndole géneros como hip hop, dance o electro.

- **album_type**

Tipo de álbum de la pista de reproducción que puede ser de tres tipos: *album*, *single* o *compilation*. El más usual es álbum con un 52%. Según la media de la popularidad, tiende a tener mayor popularidad una pista cuando es sacada como single, en lugar de álbum o compilación.

4.2.2. Variables cuantitativas

- **track_popularity**

La variable objetivo con la que basaremos nuestros modelos predictivos es *track_popularity*, es decir, la popularidad de una pista en Spotify. La popularidad de una pista es un valor entre 0 y 100, siendo 100 la más popular. La popularidad se calcula por algoritmo y se basa, en su mayor parte, en el número total de reproducciones que ha tenido la pista y lo recientes que son esas reproducciones. La media de popularidad de las pistas se encuentra en torno a 40. El 25% de las pistas tienen una popularidad menor o igual a 29, mientras que el 75% de las canciones tienen una popularidad menor o igual a 52.

- **track_acousticness**

Medida de confianza entre 0 y 1 sobre si la pista es acústica. 1 representa una confianza alta en que la pista es acústica. Podemos representar esta variable con un gráfico de densidad, donde veremos que la mayor parte del conjunto de datos tiene una medida acústica cercana a 0. Alrededor del 27% de las canciones tienen una puntuación de 0, lo que significa que son completamente no acústicas.

- **track_danceability**

La bailabilidad describe lo adecuada que es una pista para bailar basándose en una combinación de elementos musicales que incluyen el tempo, la estabilidad del ritmo, la fuerza del compás y la regularidad general. Un valor de 0 es el menos bailable y 1 el más bailable. La mayoría de las pistas tienen una puntuación de bailabilidad de alrededor 0.5, siendo la puntuación más frecuente 0.7 con un 23%. Podemos decir que en su gran mayoría destaca una bailabilidad relativamente alta.

- **track_disc_number**

Número de disco en el que se encuentra la canción, en el caso de que el álbum esté compuesto de más de un disco, algo que no suele ser lo habitual. Los valores que se toman en esta base de datos es de 1 a 81, pero como hemos comentado, el 98% de registros pertenecen a un solo disco. Esta variable podríamos reagruparla en dos categorías que fuese unique o various.

- **track_duration_ms**

Duración de la pista en milisegundos. Para una mejor interpretación, transformamos la variable a minutos, pasando a llamarse **track_duration_min**. Una vez hecho, vemos que hay mucha variedad en cuanto a la duración de las pistas, desde 0 hasta 91 minutos, desde pistas que quizás son simples sonidos hasta lo que podrían ser programas de radio o cuentos narrados. La media de duración de una pista es 4.12 minutos.

- **track_energy**

La energía es una medida de 0 a 1 y representa una medida perceptual de intensidad y actividad. Normalmente, las pistas energéticas son rápidas, fuertes y ruidosas. Las características perceptivas que contribuyen a este atributo incluyen el rango dinámico, el volumen percibido, el timbre, la velocidad de inicio y la entropía general. La mayoría de las pistas tienen una energía entre 0.4 y 0.8, siendo la media 0.58.

- **track_instrumentalness**

Esta variable predice si una pista no contiene voces de 0 a 1. Cuanto más se acerque el valor a 1, mayor será la probabilidad de que la pista no contenga voces, mientras que un valor cercano a 0 sería una pista "vocal" sin instrumentalidad, como podría ser un podcast. Casi el 75% de los datos tienen un valor de 0, lo que significa que son canciones vocales, pero también hay un pico de canciones en torno a un 11% con presencia instrumental entre 0.8 y 0.9.

- **track_key**

Clave general estimada de la pista. Cada número se asigna a una clave utilizando la notación estándar Pitch Class. Por ejemplo, 0 = Do, 1 = Do#/Sib, 2 = Re… La tonalidad más frecuente es Do (key = 0) con un 11.83%, siguiéndole Sol (key = 7) con un 11.34% y Re (key = 2) con un 10.04%. En general, se puede ver una distribución relativamente uniforme de las tonalidades.

- **track_liveness**

Liveness detecta la presencia de público en la grabación de 0 a 1. Los valores de liveness más cercanos a 1 representan una mayor probabilidad de que la pista se haya interpretado en directo. La mayoría de las pistas (58.26%) tienen una probabilidad entre 0.1 y 0.2 de haber sido grabadas en vivo, por lo que habrán sido grabadas en estudio.

- **track_mode**

Variable binaria. El modo indica la modalidad (mayor o menor) de una pista, el tipo de escala del que se deriva su contenido melódico. Mayor se representa con 1 y menor con 0. El 62.2% de los datos tiene una modalidad mayor.

- **track_loudness**

La sonoridad muestra el volumen global de una pista en decibelios (dB). Los valores de sonoridad se promedian en toda la pista y son útiles para comparar la sonoridad relativa de las pistas. La sonoridad es la cualidad de un sonido que es el principal correlato psicológico de la fuerza física (amplitud). Los valores suelen oscilar entre -60 y 2.7 db.

El valor medio es de -9.59 dB, lo que indica que la mayoría de las canciones tienen un nivel de sonoridad por debajo de 0 dB, que es el nivel máximo que se puede alcanzar sin distorsión en la mayoría de los dispositivos de reproducción. Además, el valor mediano es de -7.67 dB, lo que sugiere que la mayoría de las canciones tienen un nivel de sonoridad por debajo de este valor.

- **track_speechiness**

Esta variable estudia la locuacidad de una pista. La locuacidad detecta la presencia de palabras habladas en una pista. Cuanto más hablada sea la grabación (programa de entrevistas, audiolibro, poesía), más se acercará a 1. Los valores entre 0,33 y 0,66 describen pistas que pueden contener tanto música como voz (música rap). Los valores inferiores a 0,33 representan probablemente música y otras pistas no habladas. Según los gráficos realizados, nos aporta la información de que la mayoría de datos tiene un valor muy cercano a 0, lo que sugiere que no contiene palabras habladas significativas. Esto nos crea cierta confusión ya que la variable estudiada anteriormente, **track_instrumentalness**, que medía si la pista no contiene voces y es instrumental, resulta que nos aportaba totalmente lo contrario, que la mayoría de las pistas contenían voces y no eran instrumentales.

- **track_tempo**

Tempo global estimado de una pista en pulsaciones por minuto (BPM). En terminología musical, el tempo es la velocidad o el ritmo de una pieza determinada y se deriva directamente de la duración media de los tiempos. Presenta valores de 0 a 244 BPM. La media de BPM es de 118.30 BPM.

- **track_time_signature**

Esta variable mide el compás musical de la pista, que está compuesto por varias medidas de tiempo. Un valor de 0 significa que el tiempo de la canción no está definido o es desconocido. Un valor de 1 representa un compás de una sola nota, un valor de 2 representa un compás de dos notas… y así hasta 5. El valor más representado es 4 con un 86.06%, es decir, la mayoría de las pistas tienen un compás de cuatro notas.

- **track_number**

Número de la pista en su álbum correspondiente. A diferencia de la variable **track_disc_number**, que nos decía el número de disco en el que se encuentra la canción dentro de su álbum, esta variable nos indica el número de la canción dentro del álbum. Esta variable puede aportarnos más información. El 44% de las pistas se encuentran en la primera posición del álbum. Hay que tener en cuenta el detalle de que no estamos diferenciando entre álbum o single, por lo que en un single es lógico que sea la primera pista y única. Más información sobre esta variable. Hay una representatividad de al menos un 1% cuando hay hasta 14 pistas en un álbum. A partir de 14, la proporción es casi nula, llegando hasta 655 como máximo.

- **track_valence**

Medida de 0 a 1 que describe la positividad musical transmitida por una pista. Las pistas con valencia alta suenan más positivas, mientras que las pistas con valencia baja suenan más negativas. Proporción muy similar a la variable **track_popularity**, tendremos que ver la relación entre ellas.

- **artist_popularity**

Popularidad del artista, que es calculada a través de la popularidad de sus canciones. Al igual que para las canciones, la popularidad del artista se mide de 0 a 100. Distribución también muy similar a la variable **track_popularity**, tendremos que ver la relación entre ellas.

- **artist_followers**

Número de seguidores del artista. Presenta una distribución muy amplia desde 0 hasta 41 millones, lo mejor será crear una nueva columna donde reagrupemos los valores para poder estudiarla. Pero antes de eso, debemos tener en cuenta que su mediana es 24.034 seguidores, por lo que probablemente la mayoría de artistas de nuestra base de datos contaran con menos de 500.000 seguidores. Observamos que casi el 90% de los artistas tienen menos de un millón de seguidores y que apenas un 3% superan los 5 millones.

- **album_tracks**

Número de pistas del álbum al que pertenece la pista. Como hay una gran parte de las observaciones que son single, en esta variable aparece como 1. Es por ello por lo que representa casi el 30% esta categoría. No obstante, la media de pistas por álbum suele ser de 11 y también contamos con casos anómalos como un álbum con 977 pistas.

- **lyric_mean_syllables_word**

Esta variable extrae de la variable **track_lyrics** la media de sílabas por palabra en la letra de la pista. La media de sílabas por palabra es de 1.05 y el máximo de 4. Destacar que hay 15.387 pistas, el 16% del total, que tienen un valor de -1. Estas pistas son las que no tienen letra de la canción ya sea porque son pistas totalmente acústicas o porque no están registradas en la plataforma Genius.

- **lyric_mean_words_sentence**

Media de palabras por frase en la pista. La media de palabras por frase es de 3.44 y puede comprender en una frase desde 0 palabras hasta 252.

- **lyric_n_sentences**

Número de frases en la pista. La media de frases por pista es de 43.57 y puede comprender desde 2 frases hasta 2519.

- **lyric_n_words**

Número de palabras en la pista. La media de palabras por pista es de 313.8 y puede comprender desde 1 palabra hasta 39111.

- **lyric_sentence_similarity**

Esta variable mide la similitud entre frases de la pista. Un valor cercano a 0 nos diría que no hay apenas relación entre frases, como puede ser una historia o un podcast, donde usualmente no se repiten frases, mientras que un valor cercano a 1 supondría una fuerte relación entre frases, como puede ser una canción de reggaeton, donde se suelen repetir muchas veces la misma frase. La media de esta variable se encuentra en -0.11, ya que los 15.000 registros que no cuentan con letra tienen una fuerte importancia. No obstante, el valor más repetido es el 0 con un 46% de representatividad, por lo que no hay pistas con una gran similitud.

- **lyric_vocabulary_wealth**

Nivel de riqueza del vocabulario. Un valor cercano a 0 se cataloga como una canción con pobre vocabulario y un valor cercano a 1 se cataloga como una canción rica en vocabulario. La media es de 0.30 (teniendo en cuenta las canciones sin letra) y el valor más repetido es 0.6, por lo que se puede considerar que la gran parte de las pistas cuentan con un vocabulario rico.

4.2.3. Variables fecha

- **album_release_date**

Fecha de lanzamiento de la pista, ya sea como álbum o como single. Comprende fechas desde 1926 hasta 2019, pero si es verdad que pistas del siglo XX solo hay un 6.33%. El año con más pistas registradas es 2018 con casi 25.000 pistas (28.7%), seguidos de 2017 y 2019, con un 13% aproximadamente cada uno.

4.3. Depuración de los datos

4.3.1. Descarte de variables

Una vez realizada la exploración de las variables, debemos mencionar tambien las variables que hemos descartado del dataset y que no hemos mencionado en la exploración, para que quede constancia de ello. Dichas variables son eliminadas porque tienen una equivalencia con otra variable, o porque no cuenta con representatividad como puede ser por ejemplo el nombre de la pista. Las nombramos a continuación:

- **album_id**: Número identificativo del álbum, equivale a album_name. No usaremos esta variable.
- **artist_id**: Número identificativo del artista, equivale a artist_name. No usaremos esta variable.
- **track_id**: Número identificativo de la pista, equivale a track_name. No usaremos esta variable.
- **track_lyrics**: Esta variable que aporta la letra de la canción ya la tenemos transformada en las variables en el conjunto de variables de lyrics_features extraída de Genius, ofreciéndonos el número de palabras y frases en la letra. Por tanto, no es necesario usar esta variable.
- **track_name**: Esta variable nos muestra el nombre de la pista. Lógicamente, cada pista tiene un nombre, pero debemos tener en cuenta que hay canciones que se llaman igual, con nombres típicos como Home, You, Time, Breathe… Por otro lado, también nos encontramos con las mismas canciones y mismo nombre que ocurre cuando una misma canción fue sacada por ejemplo como single, en álbum, con una colaboración posteriormente… No obstante, no usaremos esta variable ya que sus registros no tienen apenas representatividad.
- **artist_name**: Artistas de las respectivas pistas de reproducción. Son más de 40000 artistas distintos y no hay ninguno con una representatividad mayor que el resto. No usaremos esta variable.
- **album_name**: Álbum del que procede la pista de reproducción. Son más de 66000 registros y al igual que en artist_name, ninguno tiene una proporción destacada. No usaremos esta variable.

4.3.2. Presencia de valores Missing

Tras observar el fichero y haber creado nuevas variables, consideramos que es importante realizar una “limpieza” de los datos disponibles, como la transformación de alguna variable tipo fecha o la transformación a variables de tipo numérico o factor.

Por otra parte, hemos podido observar que en el dataset proveniente de *lyrics_features*, tenemos 6964 registros que no cuentan con la información correspondiente en la plataforma Genius y vienen marcados como *NA*. También, existe en la variable de

fecha *album_release_date* 8.367 observaciones que no cuentan con la fecha de lanzamiento. Para tener una base de datos con información completa, decidimos eliminar estos registros, ya que son variables donde no es posible imputarles otro valor.

4.3.3. Grado de importancia de las variables

A través del explorador de estadísticos de SAS Miner, podemos detectar las variables que más nos aportarán a nuestros modelos. Al ser nuestra variable objetivo continua, los estadísticos consultados han sido el gráfico de correlación de Pearson y el valor de la variable respecto a la variable objetivo. Podemos ver con el gráfico de correlación de Pearson que las variables de intervalo no están altamente correlacionadas con la variable objetivo, salvo la variable *artist_popularity* que sería una excepción alcanzando un 0.71, la mayor correlación positiva sería *artist_followers* con un 0.29, mientras que la mayor correlación negativa sería *track_speechiness* con un -0.29.

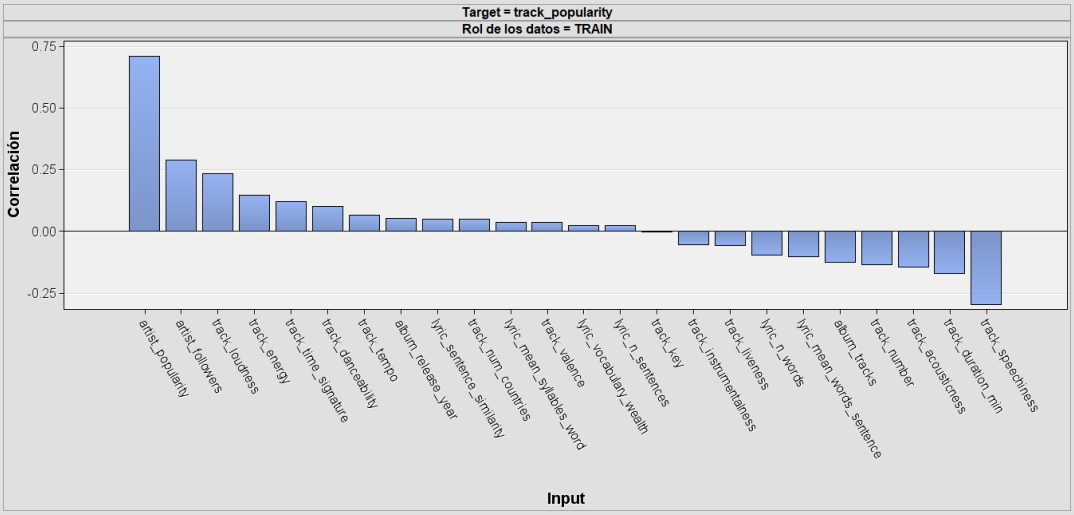


Figura 22. Gráfico de correlación de Pearson

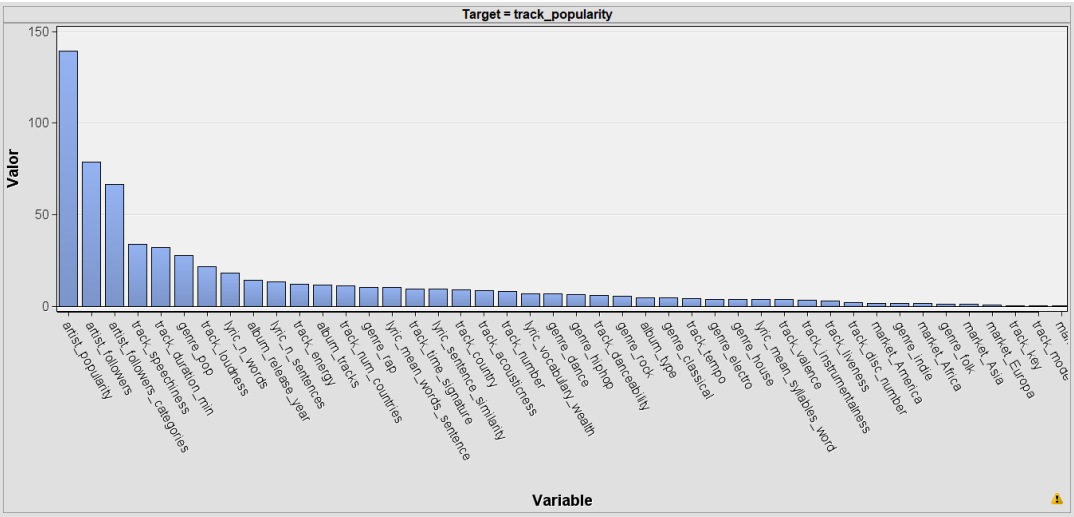


Figura 23. Gráfico de valor de la variable

Para finalizar este apartado de exploración y depuración de nuestra base de datos, vemos las características de las variables en función de su categoría, donde podremos observar la distribución de cada variable. Contaremos con 86.608 observaciones y 46 variables, de las cuales 6 son categóricas, 15 de ellas son binarias y 25 son numéricas.

Variables categóricas	**n_unique**	**top_counts**
track_country	3	*AR*: 42470, *FI*: 26113, *BE*: 18025
track_disc_number	2	*unique*: 85213, *various*: 1395
track_mode	2	*mayor*: 53421, *minor*: 33187
album_type	3	*album*: 43095, *single*: 38298, *compilation*: 5215
artist_followers_categories	5	*0 - 100K*: 57904, *100K - 500K*: 13953, *500K - 1M*: 5302, *1M - 5M*: 7149, *> 5M*: 2300
album_release_centurie	2	*S. XX*: 5484, *S. XXI*: 81124

Tabla 3. Variables de tipo categórica

Variables tipo binaria	**n_unique**	**top_counts**
market_Africa	2	1: 80073, 0: 6535
market_America	2	1: 82582, 0: 4026
market_Asia	2	1: 81277, 0: 5331
market_Europa	2	1: 82987, 0: 3621
market_Oceania	2	1: 78599, 0: 8009
genre_rap	2	0: 80899, 1: 5709
genre_pop	2	0: 59567, 1: 27041
genre_rock	2	0: 72601, 1: 14007
genre_house	2	0: 81390, 1: 5218
genre_indie	2	0: 75097, 1: 11511
genre_dance	2	0: 80412, 1: 6196
genre_folk	2	0: 82415, 1: 4193
genre_classical	2	0: 83713, 1: 2895
genre_hiphop	2	0: 79446, 1: 7162
genre_electro	2	0: 82046, 1: 4562

Tabla 4. Variables de tipo binaria

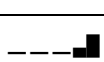
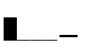
Variables tipo numérica	mean	sd	p0	p25	p50	p75	p100	hist
track_popularity	40.42	16.83	0.00	30.00	41.00	52.00	97.00	
track_acousticness	0.35	0.34	0.00	0.04	0.23	0.64	1.00	
track_danceability	0.59	0.18	0.00	0.48	0.61	0.72	0.99	
track_energy	0.58	0.26	0.00	0.41	0.63	0.80	1.00	
track_instrumentalness	0.16	0.31	0.00	0.00	0.00	0.06	1.00	
track_key	5.27	3.58	0.00	2.00	5.00	8.00	11.00	
track_liveness	0.20	0.17	0.00	0.10	0.12	0.24	1.00	
track_loudness	-9.46	6.29	-60.00	-11.06	-7.54	-5.48	2.72	
track_speechiness	0.13	0.20	0.00	0.04	0.05	0.10	0.97	
track_tempo	118.27	30.35	0.00	95.89	118.13	136.07	242.32	
track_time_signature	3.88	0.52	0.00	4.00	4.00	4.00	5.00	
track_number	4.49	7.16	1.00	1.00	2.00	6.00	655.00	
track_valence	0.47	0.26	0.00	0.26	0.46	0.68	0.99	
artist_popularity	47.68	20.28	0.00	33.00	49.00	63.00	100.00	
artist_followers	556444.34	2035948.80	0.00	1846.00	21908.00	217834.00	41561693.00	
album_tracks	10.75	16.35	1.00	1.00	9.00	14.00	977.00	
lyric_mean_syllables_word	1.04	0.97	-1.00	1.14	1.26	1.51	4.00	
lyric_mean_words_sentence	3.45	8.25	-1.00	2.55	3.28	3.97	252.50	
lyric_n_sentences	43.89	45.90	-1.00	22.00	42.00	61.00	2519.00	
lyric_n_words	316.91	824.55	-1.00	127.00	244.00	363.00	39111.00	
lyric_sentence_similarity	-0.12	0.40	-1.00	0.01	0.03	0.07	0.96	
lyric_vocabulary_wealth	0.29	0.59	-1.00	0.40	0.53	0.63	0.78	
track_num_countries	72.37	19.43	1.00	78.00	79.00	79.00	79.00	
track_duration_min	4.08	3.19	0.02	3.06	3.60	4.31	91.76	
album_release_year	2013.63	8.24	1926.00	2012.00	2017.00	2018.00	2019.00	

Tabla 5. Variables de tipo numérica

5. Modelización

5.1. Selección de variables

La selección de variables es fundamental para realizar la modelización de cualquier conjunto de datos. Antes de empezar con los distintos modelos que llevaremos a cabo en este trabajo, vamos a realizar una selección de variables para poder obtener unos resultados sin sobreajuste y optimizar el trabajo computacional.

Primero de todo, trabajaremos con un muestreo del 20% del conjunto de los datos, lo cual supone 17.281 observaciones, ya que trabajar con el 100% de los datos resultaba computacionalmente inabordable. El siguiente paso es estandarizar las variables numéricas de modo que todas tengan los mismos pesos a la hora de ejecutar los modelos. Por otra parte, las variables de clases es necesario convertirlas a variables *dummies*, donde se crearán nuevas variables ficticias que tomarán el valor 1 si la observación correspondiente toma el valor del nivel de la propia variable, y 0 en caso contrario. De esta manera, pasaremos a tener 72 variables input en nuestro dataset.

Trabajaremos con los criterios de información *AIC* (Akaike Information Criterion) y *BIC* (Bayesian Information Criterion). Además, por cada criterio de información probaremos los métodos de selección de variables *Stepwise* y *Backward* (decidimos descartar el método *Forward* ya que sus resultados eran idénticos al método *Stepwise*). Esto nos generaría 4 selecciones de variables posibles.

A través de validación cruzada repetida, sortearemos cada vez un 70% de datos train con 50 semillas distintas. Durante el proceso, se realizará una cuenta de las veces que cada criterio de información escoge una selección de variables. Una vez ejecutado, se escogerán las dos selecciones de variables que más veces se hayan contado, por lo que tendríamos un total de 8 selecciones de variables posibles. Para poder distinguirlas, las denominaremos como *menor* o *mayor*, según el número de variables escogido. En la siguiente tabla se muestran los distintos modelos que crearemos:

Tipo de modelo	Nombre del modelo
Regresión Stepwise con AIC (menor)	Modelo 1
Regresión Stepwise con AIC (mayor)	Modelo 2
Regresión Backward con AIC (menor)	Modelo 3
Regresión Backward con AIC (mayor)	Modelo 4
Regresión Stepwise con BIC (menor)	Modelo 5
Regresión Stepwise con BIC (mayor)	Modelo 6
Regresión Backward con BIC (menor)	Modelo 7
Regresión Backward con BIC (mayor)	Modelo 8

Tabla 6. Modelos de selección de variables

Una vez realizados todos los modelos de selección de variables, se muestran en la siguiente tabla los conjuntos de variables que cada modelo ha escogido, para que podamos valorarlos de forma conjunta y elegir el más apropiado (comentar que las variables que no han entrado en ningún modelo no han sido incluidas en esta tabla para una mejor lectura).

	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6	Modelo 7	Modelo 8
album_release_centurie.S_XX			✓	✓				✓
album_release_year			✓	✓				✓
album_tracks	✓	✓	✓	✓	✓	✓	✓	✓
album_type.album	✓	✓	✓	✓				
album_type.single					✓	✓	✓	✓
artist_followers_categories.0_100K	✓	✓	✓	✓	✓	✓	✓	✓
artist_followers_categories.100K_500K		✓		✓				
artist_popularity	✓	✓	✓	✓	✓	✓	✓	✓
genre_classical.0	✓	✓	✓	✓	✓	✓	✓	✓
genre_dance.0	✓	✓	✓	✓	✓	✓	✓	✓
genre_electro.0	✓	✓	✓	✓				
genre_hiphop.0	✓							
genre_house.0	✓	✓	✓	✓	✓	✓	✓	✓
genre_pop.0	✓	✓	✓	✓				
genre_rap.0	✓	✓	✓	✓				
genre_rap.1					✓	✓	✓	✓
lyric_mean_words_sentence		✓		✓				
lyric_n_sentences		✓		✓				
lyric_n_words	✓	✓	✓	✓				
lyric_sentence_similarity	✓	✓	✓	✓	✓	✓	✓	✓
lyric_vocabulary_wealth	✓	✓	✓	✓	✓	✓	✓	✓
market_America.0		✓		✓				
market_America.1	✓		✓		✓	✓	✓	✓
market_Asia.0	✓	✓	✓	✓	✓	✓	✓	✓
market_Europa.0	✓	✓	✓	✓		✓		✓
market_Oceania.0		✓		✓				
track_country.AR	✓	✓	✓	✓	✓	✓	✓	✓
track_country.BE	✓	✓	✓	✓		✓		✓
track_danceability	✓	✓	✓	✓	✓	✓	✓	✓
track_disc_number.unique	✓	✓	✓	✓				
track_duration_min	✓	✓	✓	✓	✓	✓	✓	✓
track_instrumentalness		✓		✓				
track_liveness	✓	✓	✓	✓				
track_num_countries	✓	✓	✓	✓	✓	✓	✓	✓
track_number	✓	✓	✓	✓		✓		✓
track_speechiness	✓	✓	✓	✓	✓	✓	✓	✓
track_tempo		✓		✓				
track_time_signature	✓	✓	✓	✓	✓	✓	✓	✓
track_valence	✓	✓	✓	✓				
N.º TOTAL DE VARIABLES	**28**	**33**	**29**	**35**	**18**	**21**	**18**	**23**

Tabla 7. Variables seleccionadas en función de cada modelo

Podemos observar que los conjuntos de variables más sencillos son el 5 y el 7 con 18 variables, los cuales son las selecciones menores del criterio BIC, donde han sido seleccionadas las mismas variables, por lo que están duplicados estos dos métodos.

5.2. Regresión lineal

Una vez realizada la selección de variables con los 8 modelos con los que contamos, procedemos a aplicar la regresión lineal a cada modelo para estudiar su RMSE y su R^2. A continuación se muestra la distribución del error de cada modelo de regresión lineal a través de un diagrama de cajas para estudiar su sesgo y varianza. Estos modelos y todos los que haremos a lo largo de este trabajo se ejecutarán mediante validación cruzada repetida con los siguientes parámetros: 4 grupos de validación cruzada y 10 repeticiones, comenzando la semilla en 1234.

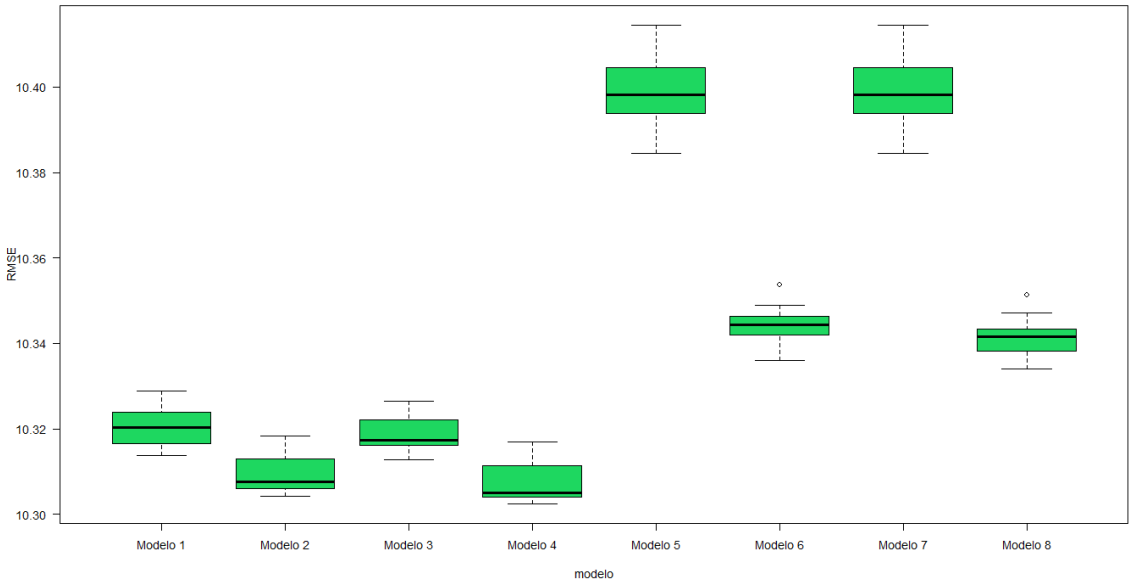


Figura 8. Diagrama de cajas de los distintos modelos de variables

Con el diagrama de cajas, podemos apreciar a simple vista que los modelos con criterio BIC (modelos 5, 6, 7 y 8) presentan un mayor error a pesar de ser los modelos más sencillos porque cuentan con menos variables. Los modelos con criterio AIC cuentan con un menor error, aunque si valoramos el R^2, veremos que las diferencias entre un criterio y otro se miden en milésimas. Por tanto, nos compensa decantarnos por un modelo con menos parámetros, ya que para futuros modelos nos facilitará el trabajo computacional.

Modelo	RMSE	Rsquared	MAE
Modelo 1	10.32036	0.6223576	8.183203
Modelo 2	10.30871	0.6232081	8.172318
Modelo 3	10.31815	0.6225183	8.180252
Modelo 4	10.30697	0.6233349	8.170477
Modelo 5	10.39828	0.6166754	8.237650
Modelo 6	10.34403	0.6206296	8.200076
Modelo 7	10.39828	0.6166754	8.237650
Modelo 8	10.34118	0.6208373	8.197797

Tabla 8. Métricas de rendimiento de la regresión lineal

Decidimos quedarnos con el modelo 6 (Stepwise con BIC), que cuenta con 22 parámetros y el cual nos permitiría explicar el 62% de la variación en la variable de respuesta. La selección de variables con la que trabajaremos en el resto de modelos es la siguiente:

Selección de variables		
album_tracks	genre_rap.1	track_country.BE
album_type.single	lyric_sentence_similarity	track_danceability
artist_followers_categories.0_100K	lyric_vocabulary_wealth	track_duration_min
artist_popularity	market_America.1	track_num_countries
genre_classical.0	market_Asia.0	track_number
genre_dance.0	market_Europa.0	track_speechiness
genre_house.0	track_country.AR	track_time_signature

Tabla 9. Selección de variables con regresión lineal

5.3. Redes neuronales

Tras haber logrado con la regresión lineal un conjunto de variables idóneo para la ejecución de diversos modelos que impida un sobreajuste debido al exceso de variables, o por el contrario, un ajuste insuficiente si se han escogido menos variables de las que se debía, vamos a aplicarlo en las redes neuronales.

Para la construcción de las redes, debemos configurar el número de nodos de la capa oculta, que dependen tanto del número de datos como de la complejidad de estos. Para no tener un modelo predictivo sobreajustado, se debe aspirar como mínimo a 20 observaciones por parámetro (Portela García-Miguel, 2023). Esto implica 864 parámetros máximo (17281/20=864).

$$h(k+1)+h+1 = nº\ observaciones/20$$

Siendo h el número de nodos ocultos y k el número de nodos input. Por tanto:

$$21h+2h+1 = 17281/20 \rightarrow h\sim 38\ nodos$$

Obtenemos que el número máximo de nodos que deberíamos probar es 38, por lo que haremos una selección partiendo desde 3 nodos, que es el número mínimo de nodos que se recomienda en redes, hasta 38. Además del número de nodos, debemos ajustar también el decay. Los valores con los que trabajarán las redes serán los siguientes:

- *size* = 3, 10, 17, 24, 31, 38.
- *decay* = 0.1, 0.01, 0.001.

Tras la ejecución de las redes, mostramos los resultados en el siguiente gráfico de puntos, donde el eje Y representa el RMSE, el eje X representa el número de nodos que hemos seleccionado y el color de los puntos representa el decaimiento. Este gráfico nos es de gran ayuda para identificar la mejor combinación de parámetros para ajustar la red y evaluar el rendimiento del modelo, realizado a través de un *expand.grid*.

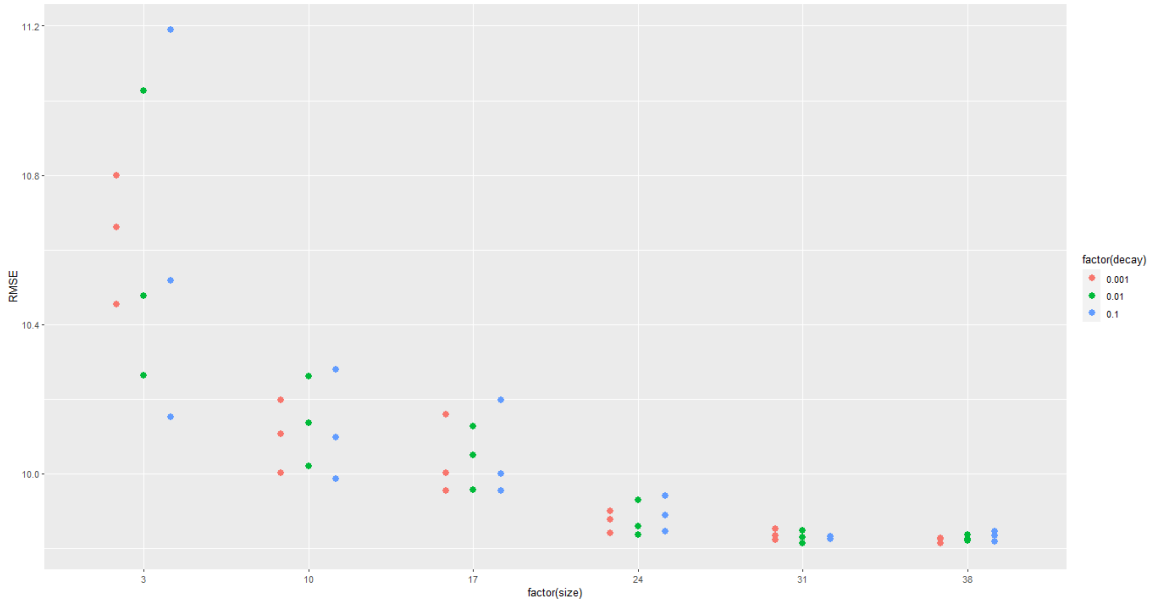


Figura 9. RMSE en función de los hiperparámetros de las redes

Si plasmamos la gráfica anterior en una tabla para analizar en números tanto el RMSE como de Rsquared, vemos como el RMAE es muy similar, al igual que el R^2, que nos indica que eligiendo 31 nodos y un decaimiento de 0.01, podríamos explicar un 65.6% de la variación en la variable de respuesta, mejorando ligeramente a la regresión lineal.

size	decay	RMSE	Rsquared	MAE
3	0.001	10.399114	0.6195687	8.27666
3	0.01	10.450324	0.6195475	8.318552
3	0.1	10.352325	0.6222679	8.242185
10	0.001	10.226292	0.6307441	8.098591
10	0.01	10.116202	0.6379177	7.983359
10	0.1	10.189847	0.6333096	8.069348
17	0.001	10.175684	0.6343399	8.075776
17	0.01	10.07096	0.6413926	7.95908
17	0.1	10.08018	0.6407633	7.979844
24	0.001	9.903983	0.6525966	7.787869
24	0.01	9.90228	0.6524256	7.786554
24	0.1	9.92348	0.6509492	7.808022
31	0.001	9.856981	0.6555623	7.756085
31	0.01	9.847532	0.6562289	7.731919
31	0.1	9.872237	0.6546053	7.767628
38	0.001	9.877663	0.6541438	7.747346
38	0.01	9.872544	0.6545645	7.751209
38	0.1	9.90371	0.65234	7.77952

Tabla 10. Métricas de rendimiento de las redes neuronales

Si echamos un vistazo al diagrama de cajas de las redes respecto a los diversos modelos ejecutados con la regresión lineal, a través de la validación cruzada repetida, vemos que las redes mejoran su RMSE con bastante diferencia a los modelos de regresión lineal.

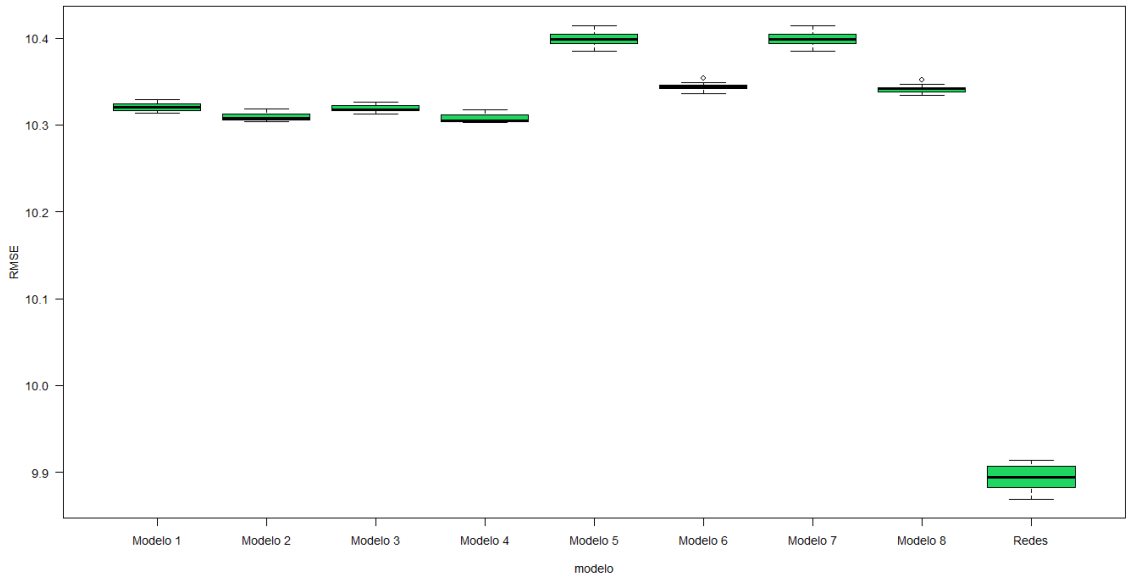


Figura 10. Diagrama de cajas de los modelos de regresión con las redes

5.4. Modelos basados en árboles

5.4.1. Bagging

El primer modelo basado en árboles que vamos a probar es el Bagging, el cual utiliza muestras con reemplazamiento. Para decidir con que parámetros quedarnos, tanto en este modelo como en el Random Forest, donde la metodología es la misma, al haber pocos parámetros que ajustar y además, ser más simples, resulta más cómodo explorarlos de forma jerárquica, es decir, realizando iteraciones y ajustes sucesivos para encontrar la configuración óptima. Es por ello por lo que aquí no utilizamos un *expand.grid* para ver todos los parámetros en un solo gráfico.

En este caso del Bagging, para el primer parámetro, no tenemos que decidir un *mtry* como tendremos que hacer con el Random Forest. Aquí el *mtry* será 21, el número de variables input que tenemos. El siguiente parámetro que debemos analizar es el *nodesize*, el número óptimo de observaciones mínimas por hoja. A menor número de observaciones por hoja se consigue un menor sesgo, pero más posible caer en sobreajuste. Ampliándolo demasiado, por el contrario, se reduce más la varianza, pero aumenta el sesgo. Hemos probado con un número mínimo de 10, 20, 30 y 40 observaciones, dejando claro que tanto en RMSE como en R^2, la mejor opción es trabajar con 10 observaciones mínimas por hoja.

nodesize	RMSE	Rsquared	MAE
10	9.819366	0.658199	7.658014
20	9.849301	0.656095	7.692697
30	9.844404	0.656484	7.69858
40	9.837309	0.656923	7.696615

Tabla 11. Métricas de rendimiento del Bagging

Una vez nos hemos decantado por el número óptimo del *nodesize*, estudiamos el número de árboles donde el error empieza a estabilizarse, para ello hacemos uso del Early Stopping. Estudiarlo es fundamental para aligerar la ejecución de los modelos y evitar la complejidad de estos. Tras probar hasta con 200 árboles para poder decidir con suficiente margen el criterio de parada, vemos que con 50 son más que suficientes ya que es cuando se normaliza el error.

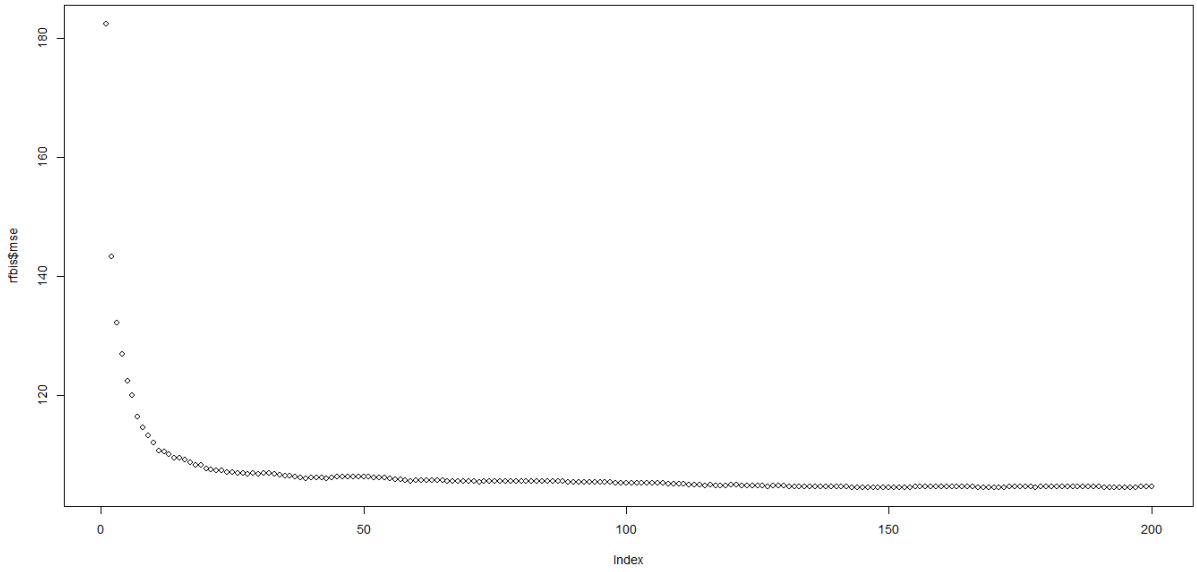


Figura 11. Estudio de Early Stopping para Bagging

Los parámetros que conformarán este modelo son un número máximo de 50 árboles, sorteo de 21 variables a introducir en cada reemplazamiento y 10 observaciones por nodo.

5.4.2. Random Forest

Tras el Bagging, pasamos ahora a ejecutar el Random Forest. Para su estudio, probamos primero con el número de variables que podría ser idóneo para cada árbol, incorporando un mtry de 3, 5, 7, 10, 13, 15, 17, 19 y 21 variables. El Random Forest es bastante robusto a variables malas, por lo que es recomendable poner el máximo de variables inicialmente. Se prueba con 300 árboles y con un número mínimo de 10 observaciones por nodo. En la tabla se observa que el mejor modelo obtenido es con un mtry de 7, obteniendo un R^2 del 66.62%, aproximadamente.

mtry	RMSE	Rsquared	MAE
3	9.919076	0.6585785	7.848812
5	9.717956	0.6664444	7.62072
7	9.710212	0.6661937	7.594118
10	9.736782	0.6641558	7.59471
13	9.762324	0.6623572	7.613049
15	9.791759	0.660323	7.629055
17	9.804272	0.6594634	7.642046
19	9.82214	0.6582384	7.649051
21	9.848964	0.6563937	7.672074

Tabla 12. Métricas de rendimiento del Random Forest

Al igual que hemos hecho con el Bagging, probamos hasta 200 árboles para estudiar el Early Stopping, donde volvemos a ver que con 50 árboles, el error se estabiliza.

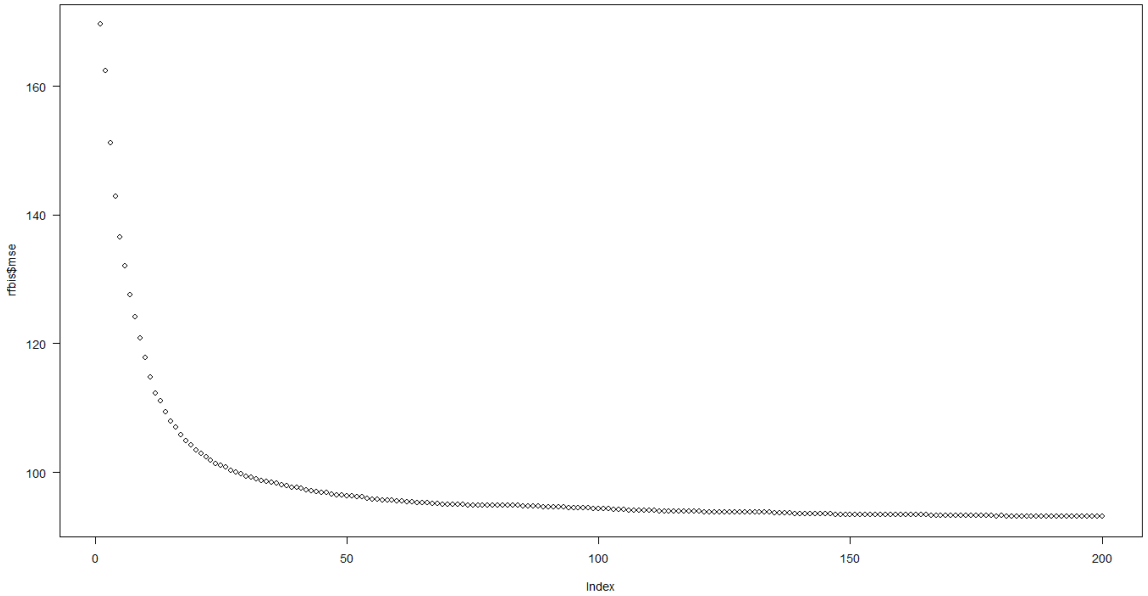


Figura 12. Estudio de Early Stopping para Random Forest

Tras ejecutar el Random Forest, podemos ver la importancia de las variables a partir del Incremento de Pureza de los Nodos, que mide la pureza de los nodos de árbol construidos durante el entrenamiento del modelo. Cuanto mayor sea el incremento, más útil es la variable para separar los datos en subgrupos homogéneos. En resumen, las variables con valores altos en ambas medidas son las más importantes para el modelo y tienen un impacto significativo en la precisión del modelo. Observaremos que la variables más importante con amplia diferencia respecto al resto es *artist_popularity*. Le siguen *artist_followers_categories.0_100K*, *track_speechiness* y *track_duration_min*.

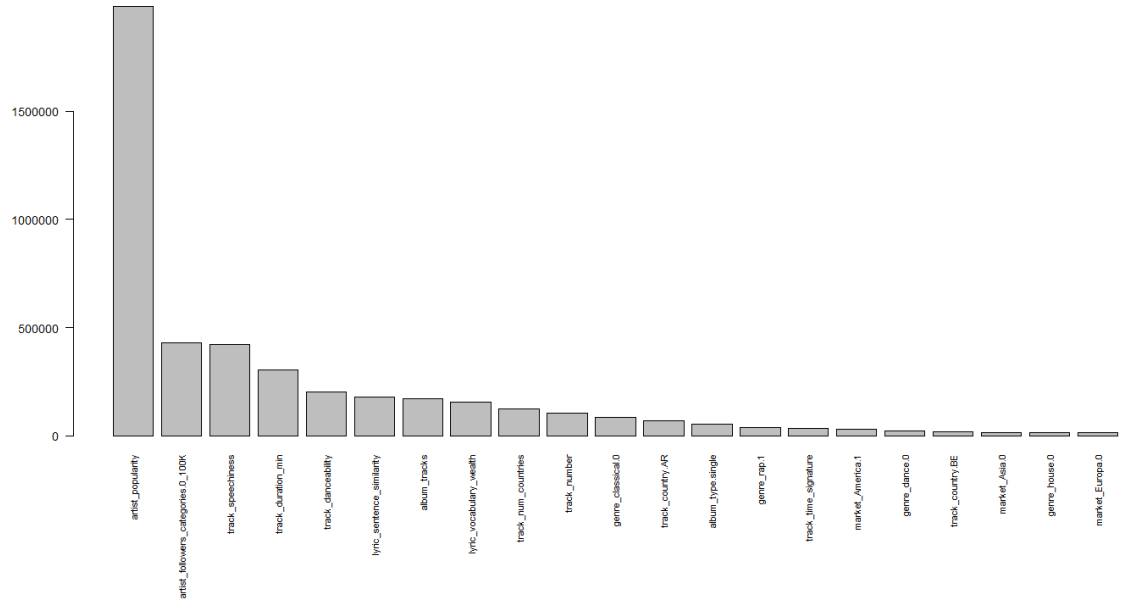


Figura 13. Importancia de las variables en Random Forest

Los parámetros que conformarán este modelo son un número máximo de 50 árboles, sorteo de 7 variables a introducir en cada reemplazamiento y 10 observaciones por nodo.

5.4.3. Gradient Boosting

Pasamos ahora a aplicar el modelo Gradient Boosting. Tanto para la configuración del Gradient Boosting como para posteriores modelos, con la finalidad de tratar de lograr el RMSE más óptimo, cambiaremos el método de evaluación que habíamos realizado con el Bagging y el Random Forest. En este caso, los parámetros a utilizar interactúan entre ellos de manera más compleja y requieren una exploración exhaustiva de todas las combinaciones posibles para encontrar la configuración óptima. El uso del *expand.grid* nos permite evaluar y comparar más fácilmente el rendimiento del modelo. Por tanto, para el Gradient Boosting se ha probado con varios valores que asignamos a los siguientes parámetros, los cuales graficaremos para conocer mejor en que niveles el error es menor y asi, buscar la mejor combinación con 4 grupos de validación cruzada:

- *shrinkage* = 0.001, 0.01, 0.03, 0.05, 0.1, 0.2.
- *n.minobsinnode* = 10, 20, 30, 50.
- *n.trees* = 500, 1000, 2000, 5000.
- *interaction.depth* = 2.

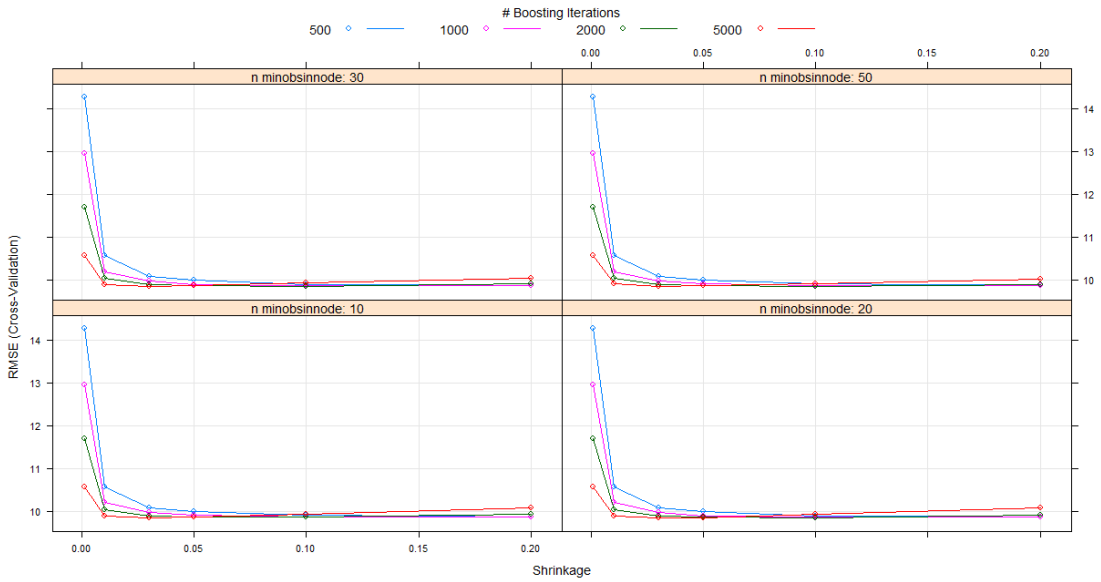


Figura 14. RMSE en función de los hiperparámetros del Gradient Boosting

Los resultados que obtenemos es que a un mayor *shrinkage*, menor RMSE habrá, pero se estabiliza en un número moderado, en torno al 0.01. En cuanto al número de iteraciones, las curvas de error van bastante parejas ya sea con 500 iteraciones o con 5000 como tenemos representados en el gráfico. No obstante, se observa claramente que a más iteraciones menor error. Para decidir el número de iteraciones idóneo, es decir, el número de árboles que nos quedaremos para este modelo, estudiamos el Early Stopping.

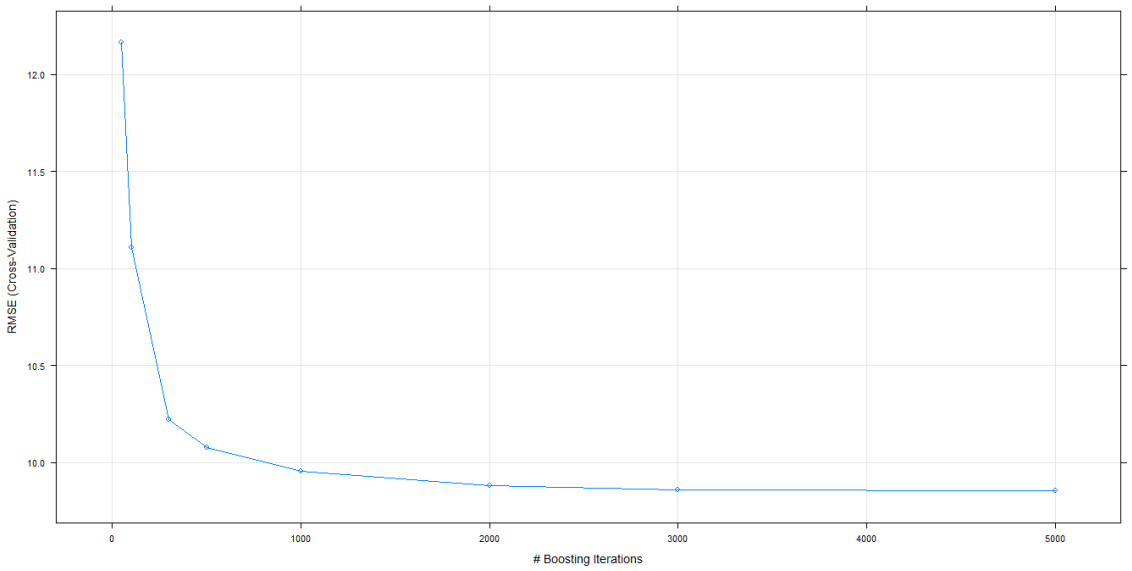


Figura 15. Estudio de Early Stopping para Gradient Boosting

Como mejor modelo nos sale el de 2000 árboles pero podemos ver en el gráfico que con 1000 árboles ya se estabiliza el error, por lo que nos quedamos con este modelo. Para finalizar en este apartado, observamos la importancia de variables según este modelo. Podremos ver que aquí la variables más determinante vuelve a ser *artist_popularity*, siguiéndole con una importancia mucho menor, *track_speechiness, genre_classical.0 y track_num_countries.*

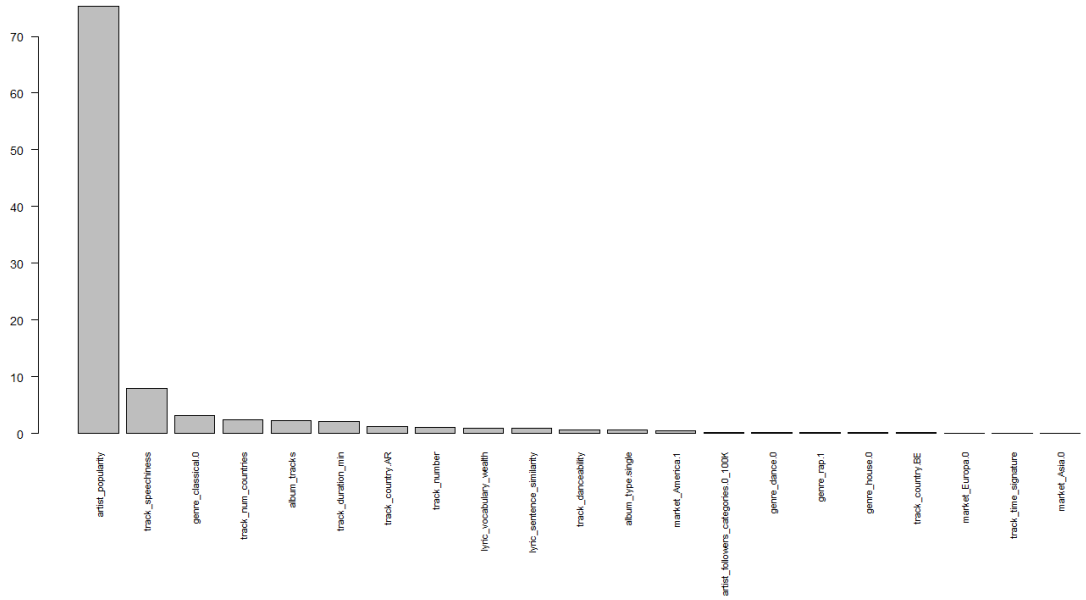


Figura 16. Importancia de las variables en Gradient Boosting

Si comparamos este gráfico con el gráfico de importancia de variables del Random Forest, veremos que no sigue el mismo orden de importancia entre un gráfico y otro. Esto se debe a que Random Forest y Gradient Boosting utilizan enfoques diferentes para evaluar la importancia de las variables. El Random Forest se basa en la medida de "Incremento de Pureza de los Nodos", mientras que el Gradient Boosting utiliza la medida de "Importancia Relativa". Dichas medidas capturan diferentes aspectos de la importancia de las variables y, por lo tanto, generan diferentes órdenes.

Los parámetros escogidos para el Gradient Boosting en la validación cruzada repetida con el resto de modelos son los siguientes: n.trees=1000, interaction.depth=2, shrinkage=0.03 y n.minobsinnode=20.

5.4.4. XGBoost

Este último modelo basado en árboles, el cual es una modificación reciente del Gradient Boosting, pero con un tratamiento del error más agresivo, cuenta con más parámetros a tunear que nombramos a continuación. Tal y como mencionábamos en el estudio teórico de este modelo, algunos de los parámetros a tunear se mantienen con un valor por defecto sin ser modificados:

- *min_child_weight* = 5, 10, 20, 30.
- *eta* = 0.001, 0.01, 0.03, 0.05, 0.1, 0.2.
- *nrounds* = 100, 500, 1000, 5000.
- *max_depth* = 6.
- *gamma* = se deja a 0 por defecto.
- *colsample_bytree* = se deja a 1 por defecto.
- *subsample* = se deja a 1 por defecto.

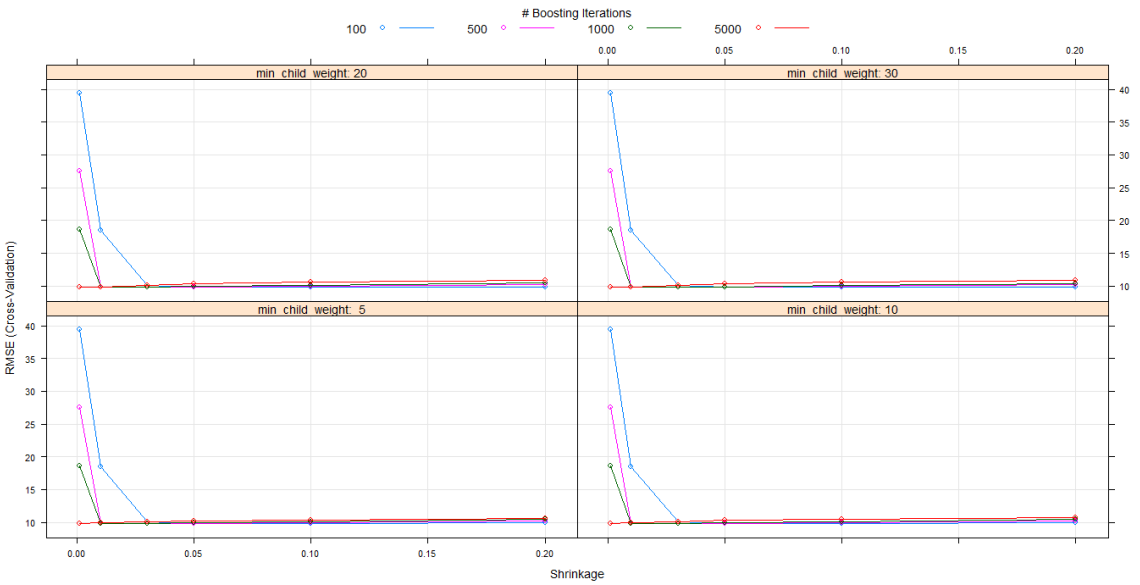


Figura 17. RMSE en función de los hiperparámetros del XGBoost

Tras su ejecución y observando los gráficos obtenidos, vemos que al igual que en el Gradient Boosting, a un mayor *eta*, menor RMSE habrá, pero se estabiliza en torno al 0.03. Decidimos que el mejor modelo para XGBoost es con los siguientes parámetros: nrounds=1000, max_depth=6, eta=0.01, gamma=0, colsample_bytree=1, min_child_weight=5 y subsample=1. Al igual que con Gradient Boosting, para estudiar el Early Stopping volvemos a ejecutar un XGBoost fijando los parámetros con los que nos hemos quedado, y probando varias iteraciones para analizar cómo evoluciona y quedarnos con el número donde el error se iguale.

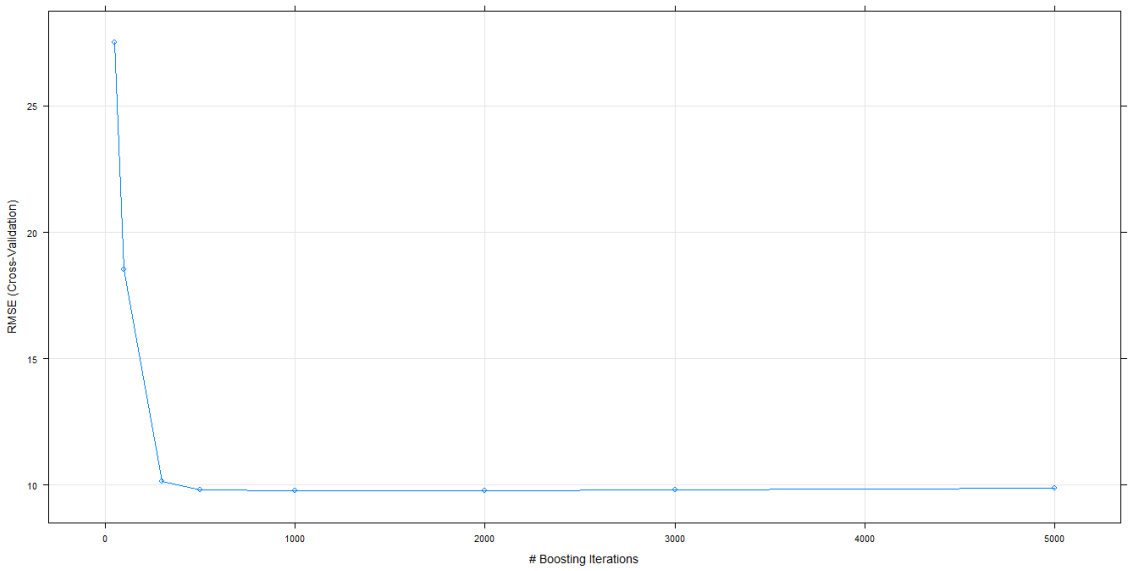


Figura 18. Estudio de Early Stopping para XGBoost

El gráfico de Early Stopping nos indica que el mejor valor son las 500 iteraciones, por lo que será el valor que utilizaremos para el modelo de XGBoost final. Para finalizar, observamos la importancia de variables según este modelo, que presenta el mismo orden que en el Gradient Boosting, al ser el XGBoost una extensión de este.

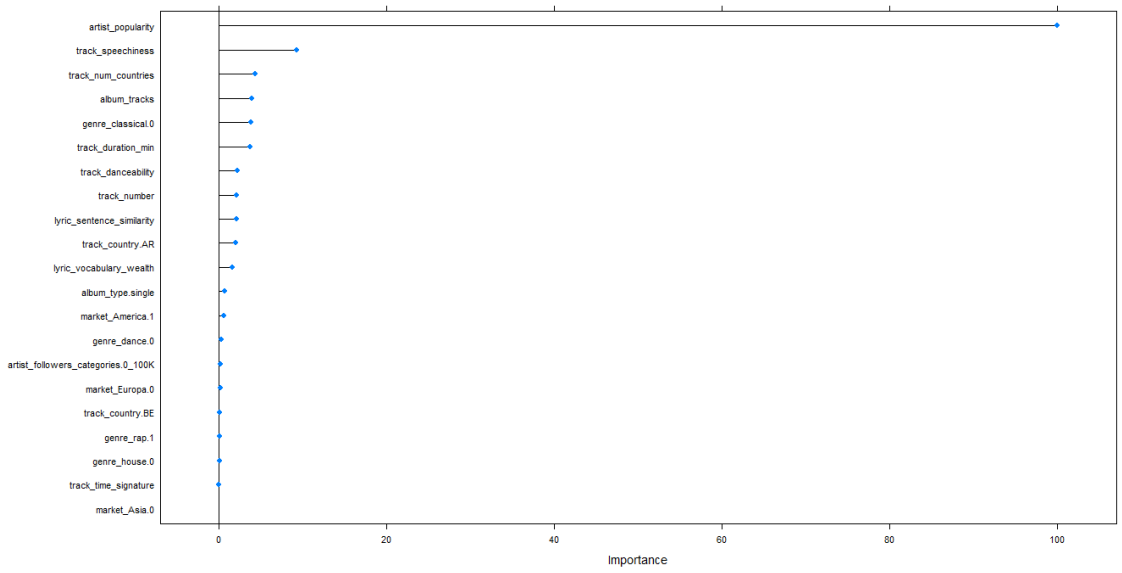


Figura 19. Importancia de las variables en XGBoost

Los parámetros escogidos para el XGBoost son los siguientes: nrounds=500, max_depth=6, eta=0.01, gamma=0, colsample_bytree=1, min_child_weight=5 and subsample=1.

Tras el estudio de los cuatro modelos basados en árboles, con todos los parámetros ya definidos, los unimos todos en un diagrama de cajas y los analizamos junto con la tabla de métricas de rendimiento. Se procede aplicando validación cruzada repetida exactamente igual que como se viene haciendo con los modelos anteriores, con 4 grupos de validación cruzada y 10 repeticiones, comenzando la semilla en 1234. Vemos que el Random Forest y el XGBoost son las mejores opciones, siendo el Random Forest el mejor con un RMSE del 9.72. El XGBoost se encuentra por detrás del Random Forest, pero ofreciendo tambien un RMSE muy ajustado, 9.79. Respecto al Bagging y al Gradient Boosting, quedan un poco peor posicionados, con una tasa de fallos de 9.81 y 9.84, respectivamente. En cuanto al R^2, el mejor modelo de árboles extraído, el Random Forest, logra un 66.49%, mejorando cualquier modelo ejecutado hasta el momento.

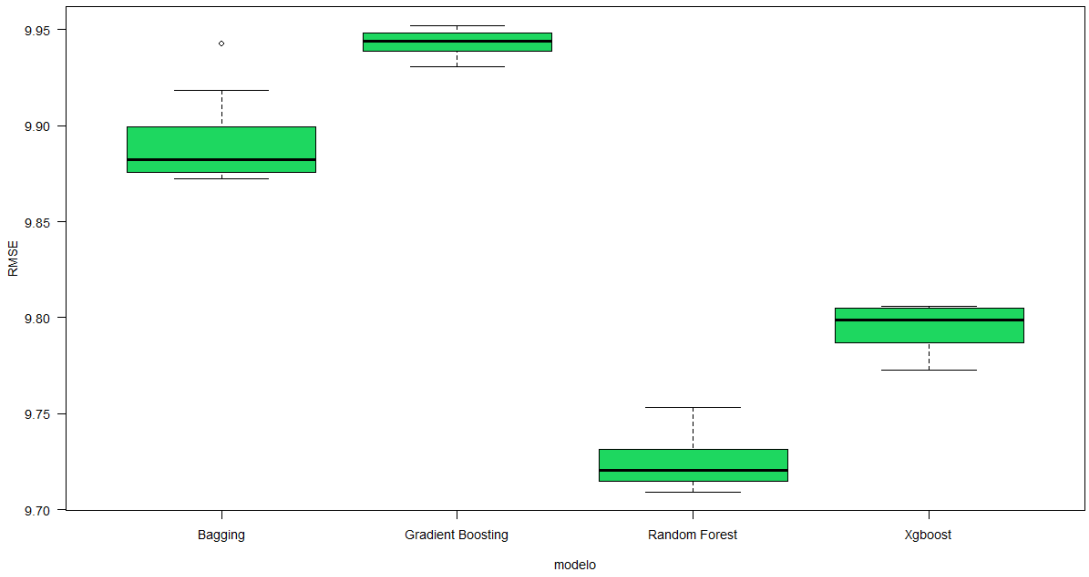


Figura 20. Diagrama de cajas de los distintos árboles

Modelo	RMSE	Rsquared	MAE
Random Forest	9.724603	0.6649371	7.609550
XGBoost	9.794844	0.6603887	7.691505
Bagging	9.819366	0.6581990	7.658014
Gradient Boosting	9.942684	0.6495384	7.841119

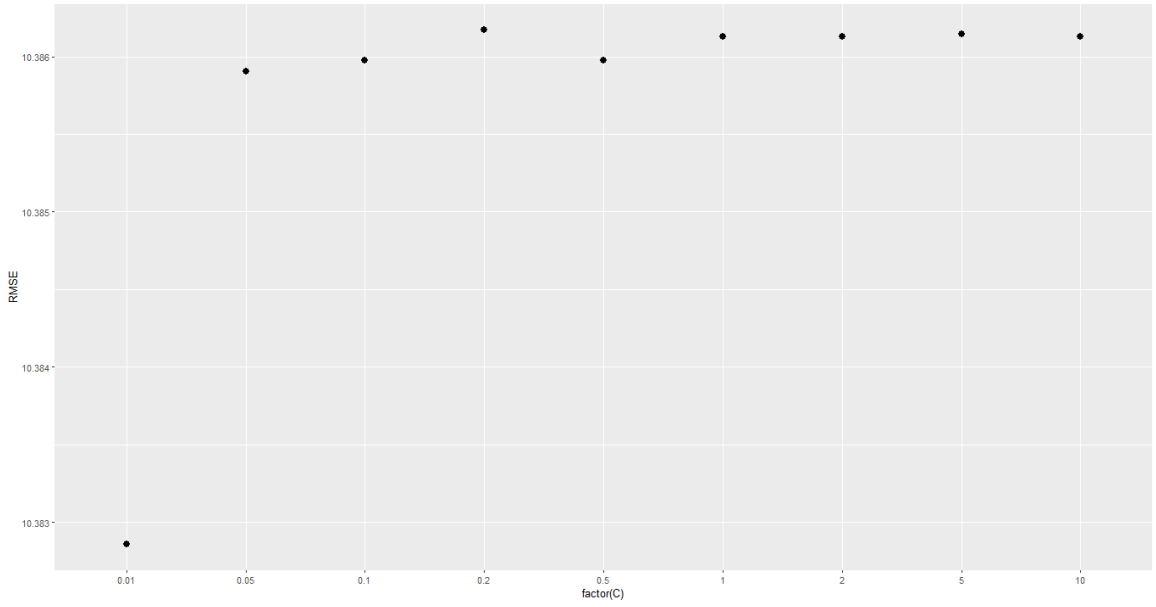
Tabla 13. Métricas de rendimiento de los modelos de árboles

5.5. SVM

Último algoritmo por analizar antes de dar paso a las técnicas de ensamblado. El Support Vector Machine es un algoritmo basado en encontrar la separación lineal perfecta de los datos. Este modelo tiene tres variantes, de los cuales solo aplicaremos dos de ellos a nuestros datos, los más determinantes: SVM Lineal y SVM Radial. El SVM Polinomial requiere de una gran carga computacional y además, muy pocas veces es la mejor opción ante las otras dos opciones que vamos a ver, por lo que decidimos no ejecutarlo.

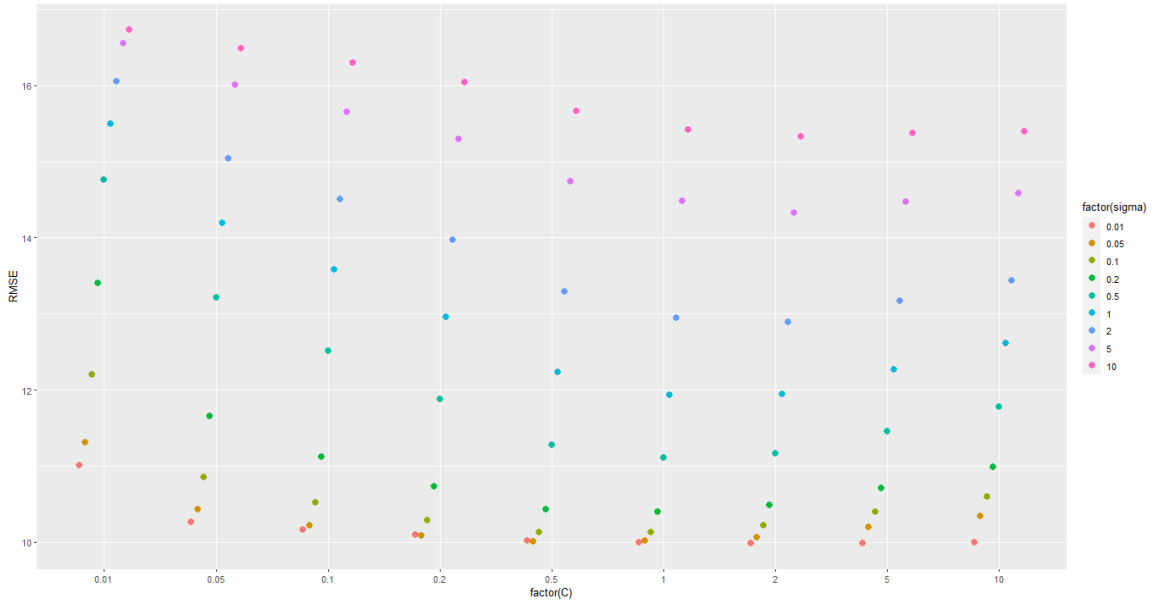
5.5.1. SVM Lineal

Para el SVM Lineal, solo debemos tener en cuenta el parámetro *C*, que es la constante de penalización, presente tambien en los siguientes modelos. Aumentar C implica un menor sesgo y un mayor sobreajuste. Tras probar con varias medidas desde 0.01 hasta 100, decidimos quedarnos entre una franja de números de 0.01 a 10, que es donde mejores resultados se obtienen, habiendo diferencias prácticamente inexistentes entre el menor y el mayor parámetro. El parámetro que escogemos con 4 grupos de validación cruzada es el 0.1.



5.5.2. SVM Radial

En el SVM Radial, no solo hay que tener en cuenta el parámetro *C*, el cual mantendremos los mismos valores que en el SVM Lineal. Aquí tambien debemos tunear el valor *sigma*, encargado de aumentar la dimensión latente de las variables input, que estará entre un conjunto de números de 0.01 y 10. Graficamos variables dos a dos para conocer en qué dirección se mueven los parámetros, donde vemos que la mejor opción con 4 grupos de validación cruzada pasa por aplicar un valor de la constante de 2 y un valor de *sigma* de 0.01.



Estudiamos a través de la validación cruzada repetida con el diagrama de cajas el RMSE de cada modelo de SVM, donde vemos que el SVM Radial mejora al SVM Lineal. No obstante, usaremos ambos modelos para las técnicas de ensamblado.

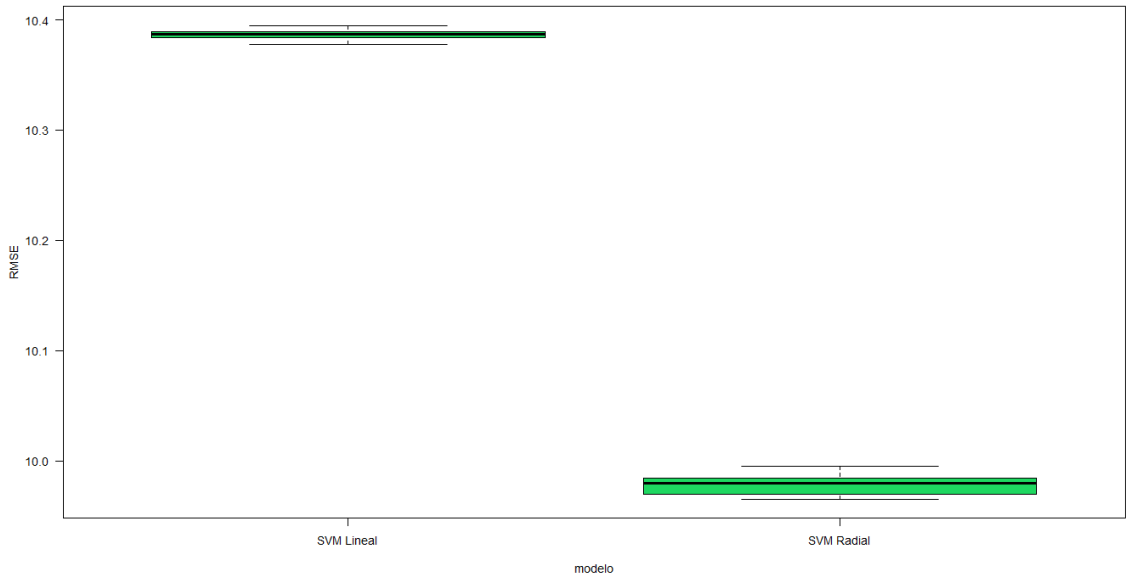


Figura 21. Diagrama de cajas de los distintos SVM

Una vez más, volvemos a mostrar un diagrama de cajas para comparar todos los modelos ejecutados hasta ahora a través de la validación cruzada repetida de 4 grupos y 10 repeticiones, donde el SVM no consigue mejorar ni a las redes ni a los modelos de árboles. De hecho, el SVM Lineal nos aporta un mayor error que la regresión.

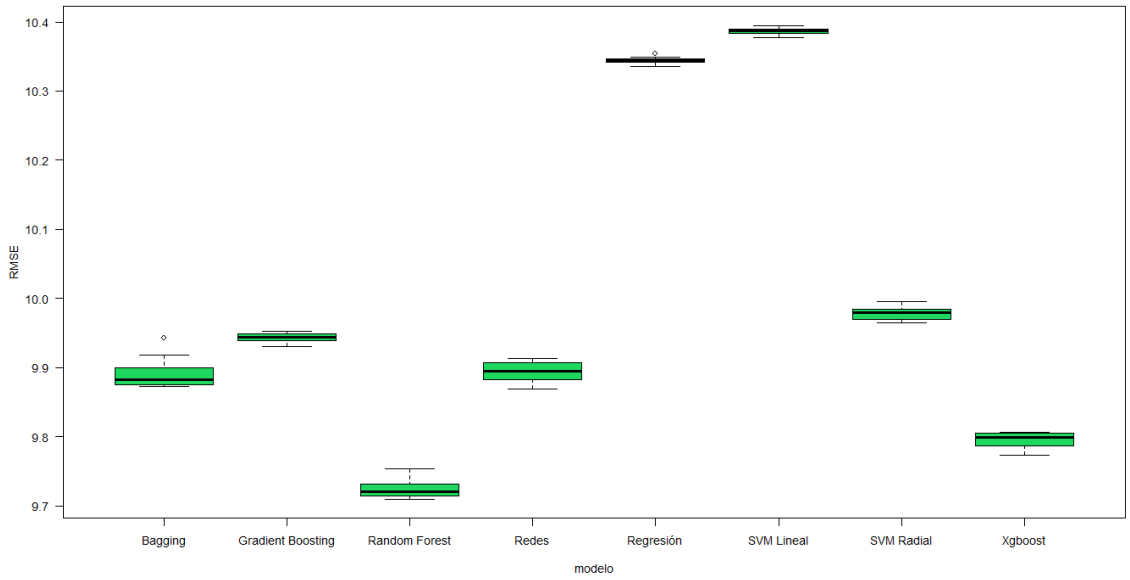


Figura 22. Diagrama de cajas de todos los modelos

5.6. Técnicas de ensamblado

Realizados y evaluados todos y cada uno de los algoritmos planteados, pasamos ahora a estudiar las técnicas de ensamblado. Como la metodología propuesta para este procedimiento consiste de unificar las predicciones de los modelos, se construirán a partir de las mejores opciones ya evaluadas de cada modelo. Por ello, a continuación mostramos una tabla resumen con los algoritmos que hemos analizado a lo largo de este trabajo, junto con las características internas del mejor modelo.

Modelo	Algoritmo	Mejor modelo
Modelo 1	Regresión Lineal	Regresión Stepwise con BIC (mayor)
Modelo 2	Redes Neuronales	size=31, decay=0.01, repeticiones=5, itera=100
Modelo 3	Bagging	nodesize=10, mtry=21, ntree=50
Modelo 4	Random Forest	nodesize=10, mtry=7, ntree=50
Modelo 5	Gradient Boosting	n.minobsinnode=20, shrinkage=0.03, n.trees=1000, interaction.depth=2
Modelo 6	XGBoost	min_child_weight=5, eta=0.01, nrounds=500, max_depth=6
Modelo 7	SVM Lineal	C=0.1
Modelo 8	SVM Radial	C=2, sigma=0.01

Hasta este punto, se habían evaluado directamente los resultados del error para la evaluación, sin tener en cuenta los valores de predicción. Para iniciar el ensamblado, es crucial considerar los resultados de las predicciones del modelo. Hemos realizado el ensamblado de 46 modelos nuevos, conformados por las predicciones entre 2, 3, 4 y 5 modelos juntos, y dividiendo por el número de técnicas usadas en cada caso, para obtener la predicción media de cada modelo de ensamblado. En el Anexo II podremos ver todas las combinaciones realizadas. En el siguiente gráfico se muestran todos ordenados por su RMSE de menor a mayor.

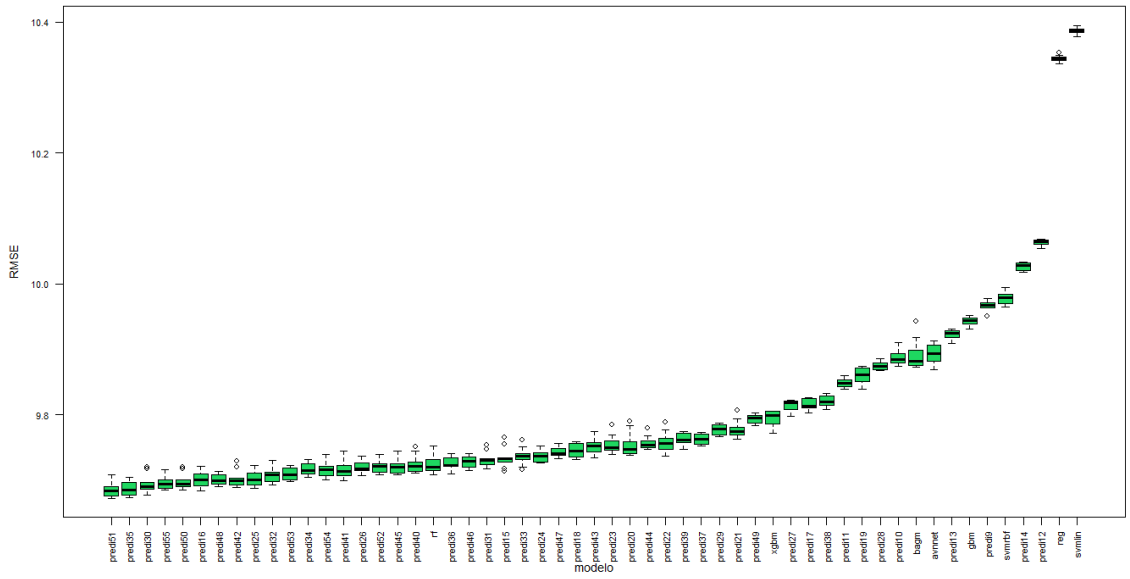


Figura 23. Diagrama de cajas de todos los ensamblados ejecutados

De entre los conseguidos, se escogen las 5 mejores combinaciones para la evaluación final con el resto de modelos, que son los que a continuación detallamos la unión de los distintos algoritmos de los que se componen.

Ensamblados	Unión de algoritmos	RMSE
predi51	Redes + Bagging + Random Forest + XGBoost	9.686079
predi35	Redes + Random Forest + XGBoost	9.687199
predi30	Redes + Bagging + Random Forest	9.694137
predi55	Redes + Bagging + Random Forest + Gradient Boosting + XGBoost	9.696877
predi50	Redes + Bagging + Random Forest + Gradient Boosting	9.698302

Tabla 14. Mejores modelos de ensamblado escogidos

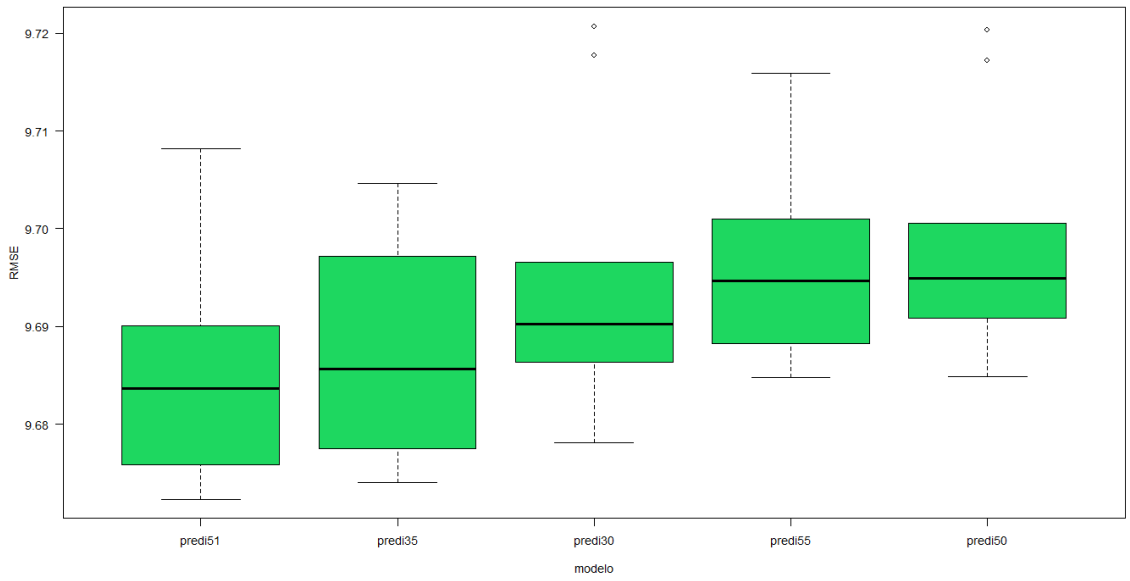


Figura 24. Diagrama de cajas de los 5 mejores ensamblados

En el siguiente gráfico, comparamos estos 5 modelos de ensamblado con el resto de algoritmos, donde vemos que los ensamblados mejoran en cuanto a error a cualquier modelo realizado individualmente.

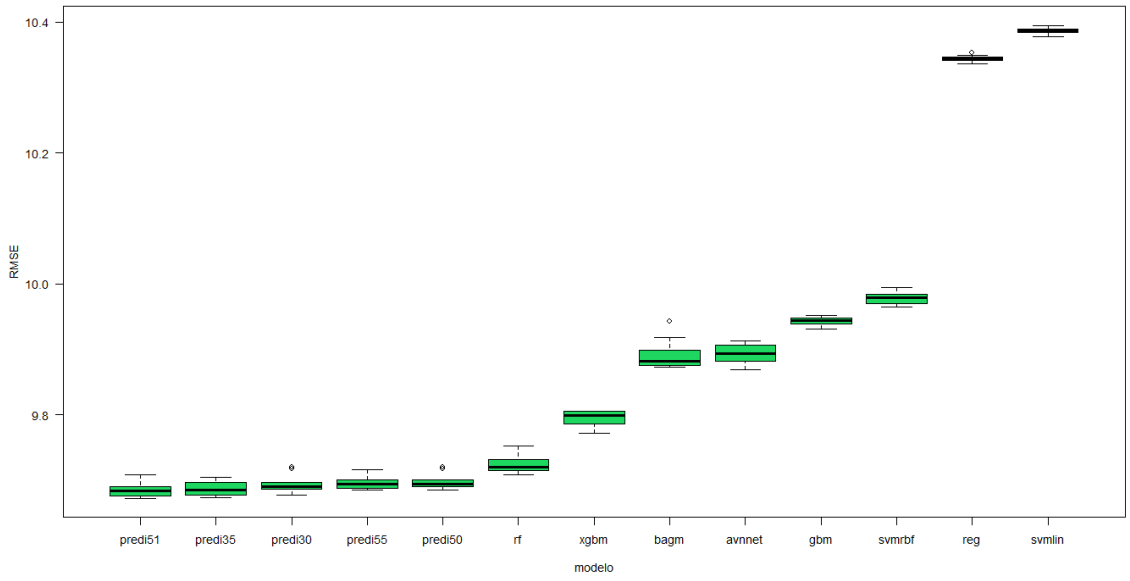


Figura 25. Diagrama de cajas de los 5 ensamblados elegidos con el resto de modelos

6. Comparación de modelos

Tras haber realizado las técnicas de ensamblado, añadimos los nuevos modelos al diagrama de cajas con el resto a través de la validación cruzada repetida, donde vemos que los ensamblados mejoran a cualquier otro modelo tanto en error como en R^2. Los ensamblados consiguen modelos más robustos, con poco sesgo y menos varianza en comparación con los modelos originales. No obstante, como no existe una diferencia notable de error entre los métodos de ensamblado y el resto de algoritmos, y puesto que las técnicas de ensamblado se caracterizan por su complejidad y difícil explicación, no los consideraremos como los mejores.

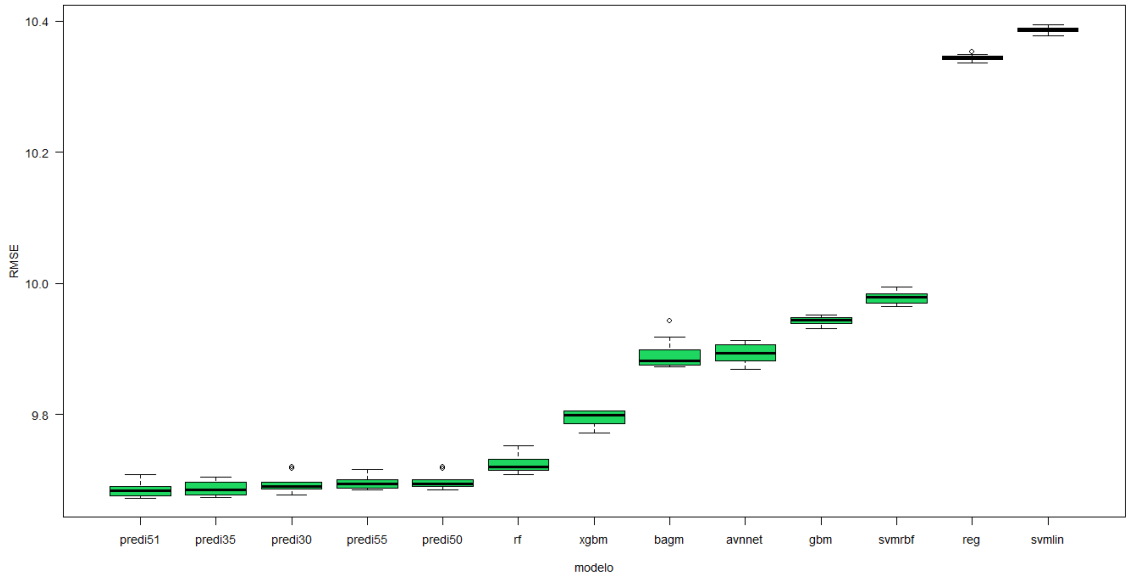


Figura 26. Diagrama de cajas de los 5 ensamblados elegidos con el resto de modelos

Modelo	RMSE	Rsquared	MAE
Random Forest	9.724603	0.6649371	7.609550
XGBoost	9.794844	0.6603887	7.691505
Bagging	9.819366	0.6581990	7.658014
Redes	9.847532	0.6562289	7.731919
Gradient Boosting	9.942684	0.6495384	7.841119
SVM Radial	9.978358	0.6495994	7.766335
Regresión	10.344030	0.6206296	8.200076
SVM Lineal	10.386010	0.6200478	8.167485

Tabla 15. Métricas de rendimiento de todos los modelos

Si descartamos los ensamblados y analizamos el error de los modelos originales, se observará que el Random Forest es el modelo con mejor rendimiento con un RMSE de 9.72 y un R^2 de 66.49%, seguido del XGBoost con un R^2 de 66.03%. En tercera posición tendríamos muy emparejados a las Redes con el Bagging, en torno al 65.70% de R^2. Por el contrario, el peor modelo obtenido sería el SVM Lineal, con un RMSE de 10.38 y un R^2 del 62%. Como ultima observación, comentar que entre el mejor y el peor modelo tan solo hay una diferencia de un 4% de R^2, por lo que todos están bastante parejos. Tanto el Random Forest como los modelos que le siguen, XGBoost, Bagging y Redes, son buenos modelos predictivos pero presentan una mayor varianza. En cambio, Gradient Boosting, Regresión y los modelos de SVM tanto lineal como radial, su varianza es mucho menor pero el error es mayor. Valorando estos dos puntos de vista, decidimos quedarnos con el modelo con menor error, es decir, Random Forest, ya que la varianza no es extremadamente notable. Como ya vimos en el estudio de este modelo, el Random Forest elegido estará conformado con un número máximo de 50 árboles, sorteo de 7 variables a introducir en cada reemplazamiento y 10 observaciones por nodo.

Con la elección del Random Forest como modelo ganador, tambien podemos interpretar más a fondo la importancia de las variables con los resultados que vimos al ejecutar este modelo. La importancia de las variables se miden a través de las siguientes dos medidas:

- **%IncMSE (Incremento en el Error Cuadrático Medio)**: medida de la disminución del rendimiento del modelo (en términos de Error Cuadrático Medio) cuando se quita una variable del modelo. Cuanto mayor sea el porcentaje, mayor será la disminución del rendimiento del modelo cuando se quita esa variable.
- **IncNodePurity (Incremento de Pureza de los Nodos)**: medida de la pureza de los nodos de árbol construidos durante el entrenamiento del modelo. Cuanto mayor sea el incremento, más útil es la variable para separar los datos en subgrupos homogéneos.

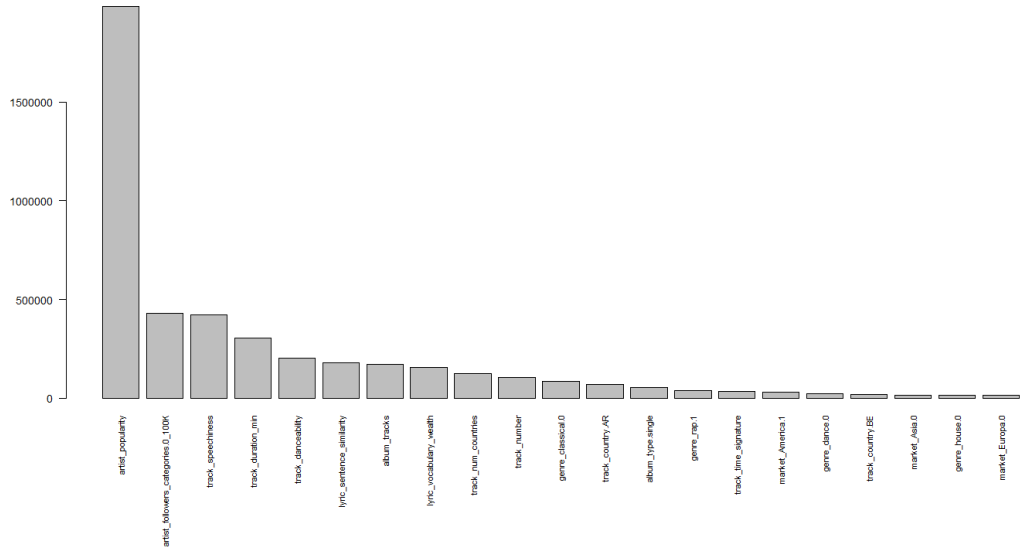


Figura 27. Importancia de las variables en Random Forest

Variables del modelo	Incremento en el Error Cuadrático Medio (%)	Incremento en la Pureza del Nodo
artist_popularity	119.72064	1985433.56
artist_followers_categories.0_100K	23.64582	428373.74
track_speechiness	56.10735	423596.36
track_duration_min	41.90412	303914.61
track_danceability	32.17292	203285.81
lyric_sentence_similarity	40.10689	178705.83
album_tracks	48.43255	173359.44
lyric_vocabulary_wealth	41.26891	155023.82
track_num_countries	47.94523	125608.67
track_number	38.56157	105096.79
genre_classical.0	45.72563	85640.39
track_country.AR	42.65843	68507.75
album_type.single	21.66068	53540.23
genre_rap.1	10.91816	37759.09
track_time_signature	10.41803	32855.75
market_America.1	22.31914	30007.64
genre_dance.0	19.54085	21409.83
track_country.BE	12.86115	18461.54
market_Asia.0	19.77278	16761.39
genre_house.0	11.60705	13632.56
market_Europa.0	16.03155	13234.41

Tabla 16. Importancia de las variables en Random Forest

Las 21 variables independientes con las que contamos en nuestro estudio tras la selección de variables que hicimos previa a la ejecución de todos los modelos, cuentan con una importancia más o menos significativa. Todas ellas tienen un valor positivo, es decir, su inclusión en el modelo contribuye positivamente a la capacidad de predicción. En el caso de haber una variable que tuviese un impacto negativo en la capacidad de predicción, su valor sería negativo, pero en nuestro conjunto no tenemos ningún caso así. Si hacemos un análisis más individualizado de las variables, hay una que destaca sobre el resto y que es la que más impacto significativo aporta en la predicción. Esta es *artist_popularity*, la cual muestra el valor de popularidad que posee el artista en su perfil de Spotify de 0 a 100, mismo criterio de puntuación que ocurre con *track_popularity*. Dicha variable aporta un incremento en el Error Cuadrático Medio de 119.72 %. La variable que le sigue, *track_speechiness*, tiene un 56.10 % de incremento, dejando claro que la diferencia entre *artist_popularity* y el resto de variables es inmensa.

7. Conclusiones y líneas abiertas

Con este trabajo hemos tratado de predecir una variable continua a través de varios modelos de Machine Learning, ayudándonos previamente de una selección de variables. El objetivo se focalizaba en predecir la popularidad de una canción de Spotify a través de varias características que aporta la plataforma musical. Las conclusiones que se extraen y el trasfondo de la investigación pueden ser aplicables a otras plataformas de entretenimiento donde se trate de entender el éxito de ciertas canciones, películas o series, y por ende, localizar los factores más importantes que justifiquen dichos éxitos.

Empezando previamente con una selección de variables a través de los criterios de información de Akaike y Bayesiano y aplicándoles la regresión lineal, pasamos a probar diversos algoritmos como redes, modelos de árboles o SVM, además de probar las técnicas de ensamblado, se ha conseguido lograr un modelo predictivo con buen rendimiento para predecir nuestra variable objetivo.

De los datos que se disponen, hemos podido extraer varias conclusiones con respecto a las variables que presentaba nuestra base de datos, logrando cumplir varios de los objetivos que nos habíamos planteado al principio del trabajo a través de la exploración exhaustiva de las variables. De las 72 variables predictoras que formaban parte de la base de datos tras la dummificacion y estandarización de los datos, con la selección de variables realizada a través de regresión lineal, observamos que el número óptimo son 22 variables. Con esto vemos que solo un 30% de las variables resultan valiosas para nuestro estudio, lo que demuestra que tener más información no siempre se traduce en mejores resultados.

La variable que más influye en la predicción de la popularidad de una canción es sin duda la popularidad del artista (*artist_popularity*), tambien valorada de 0 a 100. Otras variables importantes son los seguidores que cuenta el artista en la aplicación de Spotify (*artist_followers_categories*), muy relacionada con la variable anterior, y el nivel de locuacidad que tiene la pista (*track_speechiness*). Podemos afirmar pues que todas las características que presenta una canción de Spotify en nuestra base de datos como pueden ser las características musicales, el género musical, la duración de la pista, la fecha de lanzamiento… no tienen una relación determinante en la justificación de la popularidad de una canción. Esto sugiere que el reconocimiento y la fama del artista son factores determinantes en la aceptación y popularidad de sus canciones en Spotify.

Con estas características de nuestra extensa base de datos, trabajando con un muestreo estratificado del 20% para una actividad más eficiente computacionalmente hablando, son 17.281 observaciones que conforman el conjunto de datos con los que modelizamos los diversos algoritmos. Todos ellos logran predecir entre un 62% y un 66% de los datos, por lo que no hay grandes diferencias entre ellos. El algoritmo que mejor se adecúa a nuestro estudio y que daremos como ganador ha sido el Random Forest, con un RMSE de 9.72 y un R^2 del 66.49%. A pesar de que un 66% de predicción es un buen resultado, no podemos tomarlo de forma tan positiva, ya que debemos tener en cuenta que el modelo predictivo se apoya fundamentalmente en una variable independiente muy ligada a la variable objetivo como es *artist_popularity*. Por tanto, se puede confirmar que aunque la base de

datos proporcionó una gran cantidad de información, no fue del todo óptima, presentando ciertas limitaciones en cuanto a su capacidad para predecir la variable objetivo. Como se intuía a principio del trabajo, la popularidad de una pista musical resulta muy complicada de predecir, incluso con modelos complejos de Machine Learning. Quizás esta base de datos podría ser de mayor utilidad con otros objetivos de investigación, como por ejemplo, la clasificación del género musical a través del etiquetado automático, pasando la variable objetivo de ser numérica a ser multiclase (jazz, rap, rock…).

Es importante recordar que estas conclusiones se basan en los resultados del modelo predictivo escogido como ganador y en el conjunto de datos utilizado. Es posible que con otro conjunto de datos, otras variables hubiesen tenido un impacto más significativo. Como líneas abiertas, sería interesante la posibilidad de eliminar o mantener ciertas variables para futuros estudios. Por ejemplo, realizar el estudio descartando la variable más influyente y la cual está muy ligada a la variable objetivo, *artist_popularity*, para ver que otras variables cogerían más peso en la predicción y en qué grado mejoraría o empeoraría esta. Por otro lado, tambien podría ser de gran ayuda incluir otras variables que nos pudiesen resultar útiles. Al igual que este conjunto de datos contaba con información extraída de la plataforma musical Genius, enfocada en las letras de las canciones, se podría extraer información de plataformas como YouTube o Apple Music, para compararlas con Spotify y conseguir mejorar los resultados y obtener una visión más completa de los factores que influyen en la popularidad de las canciones.

Ya para finalizar, al evaluar el proyecto en su totalidad y considerar los objetivos establecidos, queda clara la gran importancia y aplicabilidad de la ciencia de datos en el contexto de las plataformas actuales de streaming. Este proyecto ha demostrado como las tecnologías actuales permiten extraer una amplia variedad de métricas relacionadas con las características de las canciones, lo cual facilita a los usuarios y a la industria musical a evaluar y definir el perfil de una canción. Las conclusiones de este proyecto demuestran las enormes posibilidades que la ciencia de datos puede aportar al campo de la industria musical y al análisis de datos en general. Al igual que en otros sectores, adoptar y aplicar estas técnicas en la industria permitirá a los actores involucrados obtener una ventaja competitiva significativa.

El código utilizado a lo largo de este trabajo está disponible en un repositorio de GitHub en el siguiente enlace:

https://github.com/luismpla/Prediccion-Popularidad-Spotify-TFM

8. Bibliografía

Acuña Collazos, J. A., Domínguez Castaño, A. H., & Toro Ocampo, E. M. (2012). Una comparación entre métodos estadísticos clásicos y técnicas metaheurísticas en el modelamiento estadístico. *Scientia et Technica*, 67–76.

Al-Mukhtar, M. (2019). Random forest, support vector machine, and neural networks to modelling suspended sediment in Tigris River-Baghdad. *Environmental Monitoring and Assessment*, *191*, 673.

Álvarez Liébana, J. (2022). *Apuntes de la asignatura Técnicas y Metodología de Minería de Datos (SEMMA)*.

Araujo, C. V., Neto, R. M., Nakamura, F. G., & Nakamura, E. F. (2017). Predicting music success based on users' comments on online social networks. *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, 149–156.

Aucouturier, J.-J., & Pachet, F. (2002). Music similarity measures: What's the use? *Ismir*, 339–340.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–152.

Breiman, L. (1984). *Classification and regression trees*. Routledge.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Brownlee, J. (2016). *XGBoost With python: Gradient boosted trees with XGBoost and scikit-learn*. Machine Learning Mastery.

Ciocca, S. (2017). How does spotify know you so well? *Medium*, *10*.

Friedman, J. H. (1999). *Greedy Function Approximation: A gradient Boosting Machine*.

Gandhi, R. (2018, June 7). *Support Vector Machine — Introduction to Machine Learning Algorithms*. Towards Data Science. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Karydis, I., Gkiokas, A., Katsouros, V., & Iliadis, L. (2018). Musical track popularity mining dataset: Extension & experimentation. *Neurocomputing*, *280*, 76–85.

Kim, Y., Suh, B., & Lee, K. (2014). # nowplaying the future billboard: Mining music listening behaviors of twitter users for hit song prediction. *Proceedings of the First International Workshop on Social Media Retrieval and Analysis*, 51–56.

Knees, P., & Schedl, M. (2013). A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *10*, 1–21.

Laura-Ochoa, L. (2019). Evaluación de algoritmos de clasificación utilizando validación cruzada. *17 Th LACCEI International Multi-Conference for Engineering, Education, and Technology*.

Lee, J., & Lee, J.-S. (2015). Predicting music popularity patterns based on musical complexity and early stage popularity. *Proceedings of the Third Edition Workshop on Speech, Language & Audio in Multimedia*, 3–6.

Minhondo, J. (2023). ¿Por qué las plataformas de streaming tienen cada vez más funciones? *IProUP*. https://www.iproup.com/economia-digital/37986-las-plataformas-de-streaming-tienen-cada-vez-mas-funciones

Morde, V. (2019). XGBoost Algorithm: Long May She Reign! *Towards Data Science*. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*, *4*, 51–62.

Neutelings, I. (2021, September 12). *Neural networks*. TikZ.Net. https://tikz.net/neural_networks/

Orús, A. (2023). Streaming en el mundo – Datos estadísticos. *Statista*. https://es.statista.com/temas/9059/streaming-en-en-el-mundo/#topicOverview

Portela García-Miguel, J. (2023). *Apuntes de la asignatura Técnicas de Machine Learning*.

Riebesell, J., Goodall, R., Jain, A., & Persson, K. (2023, March). Matbench Discovery-Can machine learning identify stable crystals? *Workshop on "'Machine Learning for Materials'" ICLR 2023*.

Rodríguez Montequín, M. T., Álvarez Cabal, J. V., Mesa Fernández, J. M., & González Valdés, A. (2003). Metodologías para la realización de proyectos de data mining. *VII Congreso Internacional de Ingeniería de Proyectos*.

Shahane, S. (2022). *Spotify and Genius Track Dataset*. Kaggle. https://www.kaggle.com/datasets/saurabhshahane/spotgen-music-dataset

Spotify Research. (n.d.). *Research Areas: Machine Learning*. Retrieved June 5, 2023, from https://research.atspotify.com/machine-learning/

Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. *Advances in Neural Information Processing Systems*, *26*.

ANEXOS

I. Tablas y figuras Capítulo 4.2: Análisis exploratorio

track_num_countries	n	porc
79	57312	66.17 %
78	12551	14.49 %
77	3024	3.49 %
76	1675	1.93 %
1	1537	1.77 %
2	1474	1.70 %
3	1251	1.44 %

Tabla 17. Distribución de la variable *track_num_countries*

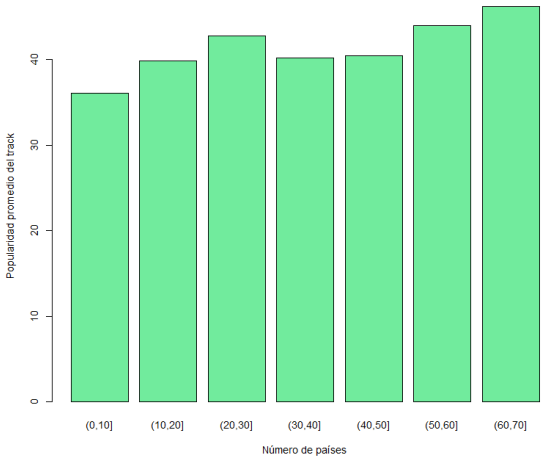


Figura 28. Relación entre el número de países y la popularidad del track

market	proporcion_market
market_Europa	95.81 %
market_America	95.35 %
market_Asia	93.84 %
market_Africa	92.45 %
market_Oceania	90.75 %

Tabla 18. Distribución del conjunto de variables *market*

track_country	n	porc	mean_popularity
AR	42470	49.03 %	43.20 %
FI	26113	30.15 %	38.45 %
BE	18025	20.81 %	36.68 %

Tabla 19. Distribución de la variable *track_country*

playlist_mean	n	porc
playlist_0	2244	2.59 %
playlist_10	4063	4.69 %
playlist_20	10388	11.99 %
playlist_30	20088	23.19 %
playlist_40	25496	29.43 %
playlist_50	18162	20.97 %
playlist_60	5382	6.21 %
playlist_70	785	0.90 %

Tabla 20. Distribución de la variable *playlist_mean*

genre	proporcion_genre	mean_popularity
genre_pop	31.22 %	48.00 %
genre_rock	16.17 %	45.44 %
genre_indie	13.29 %	43.68 %
genre_hiphop	8.26 %	48.90 %
genre_dance	7.15 %	48.68 %
genre_rap	6.59 %	53.04 %
genre_house	6.02 %	47.67 %
genre_electro	5.26 %	48.17 %
genre_folk	4.84 %	45.94 %

Tabla 21. Distribución del conjunto de variables *genre*

album_type	n	porc	mean_popularity
album	43095	49.75 %	39.29 %
single	38298	44.21 %	42.39 %
compilation	5215	6.021 %	35.22 %

Tabla 22. Distribución de la variable *album_type*

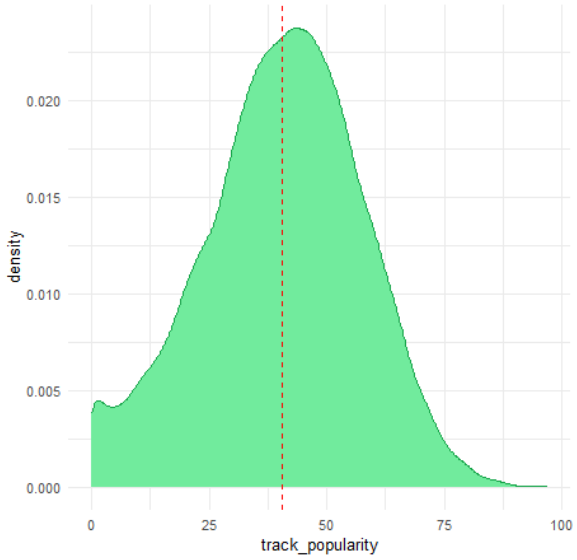


Figura 29. Distribución de la variable objetivo *track_popularity*

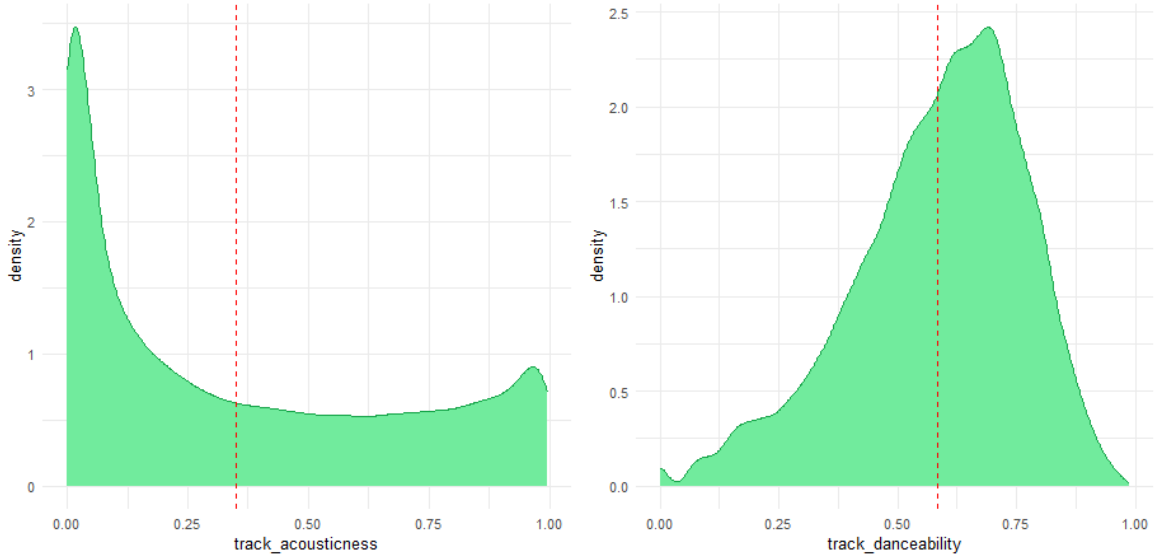


Figura 30. Distribución de las variables *track_acousticness* y *track_danceability*

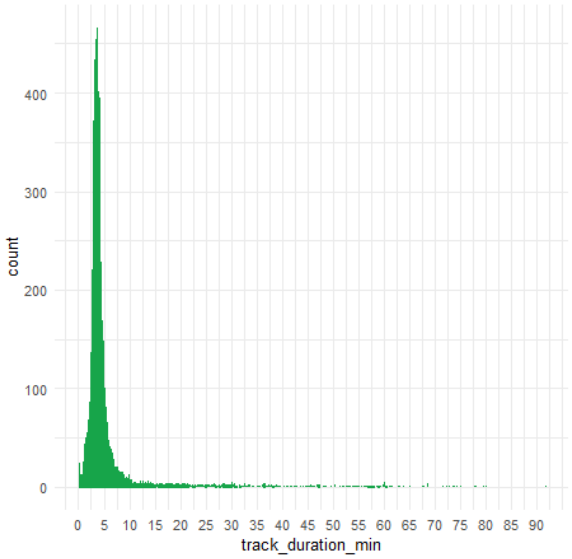


Figura 31. Distribución de la variable *track_duration_min*

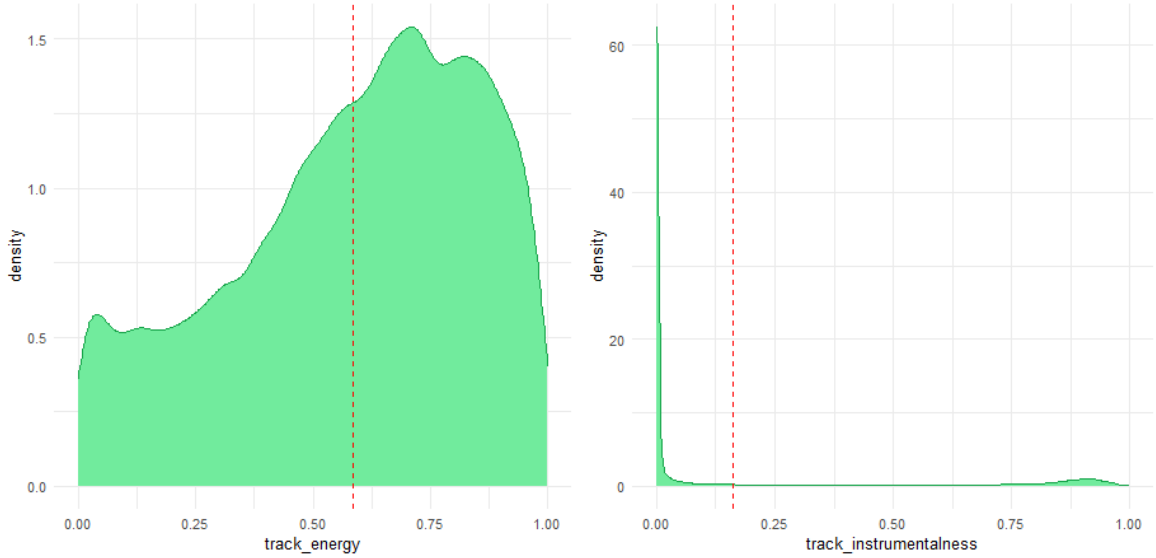


Figura 32. Distribución de las variables *track_energy* y *track_instrumentalness*

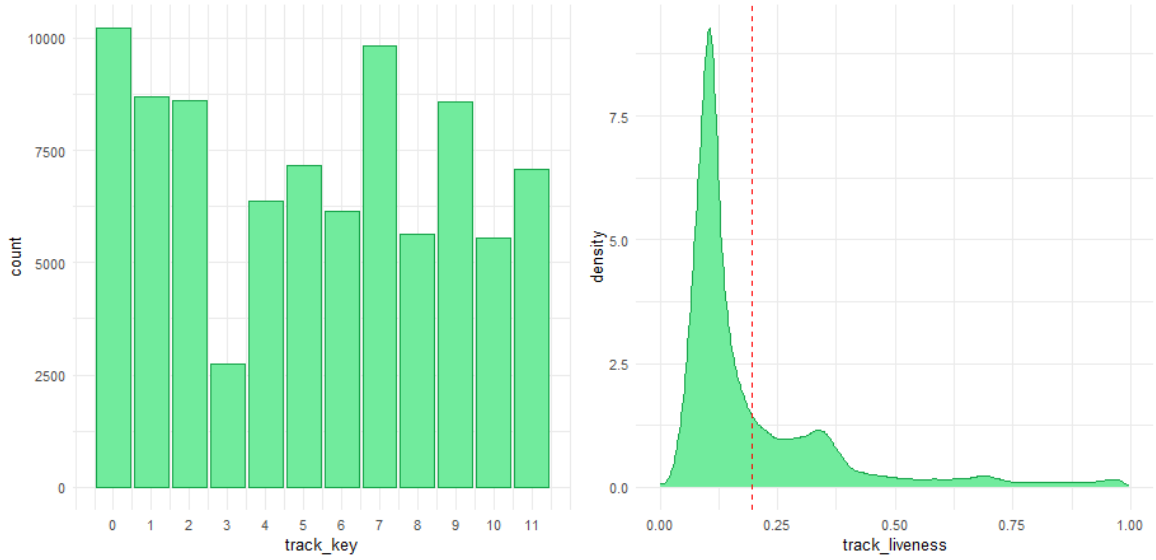


Figura 33. Distribución de las variables *track_key* y *track_liveness*

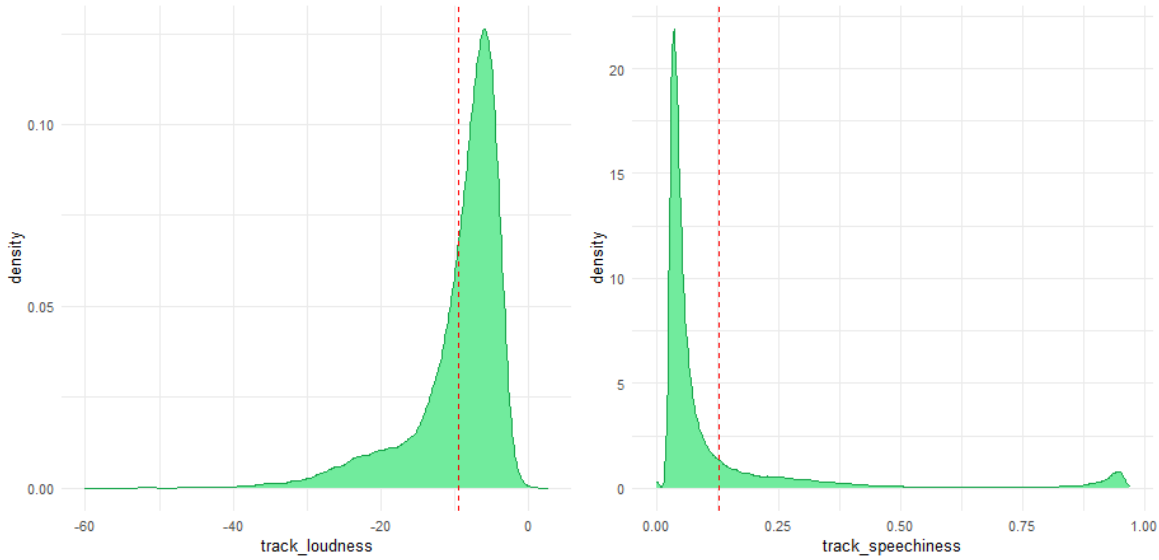


Figura 34. Distribución de las variables *track_loudness* y *track_speechiness*

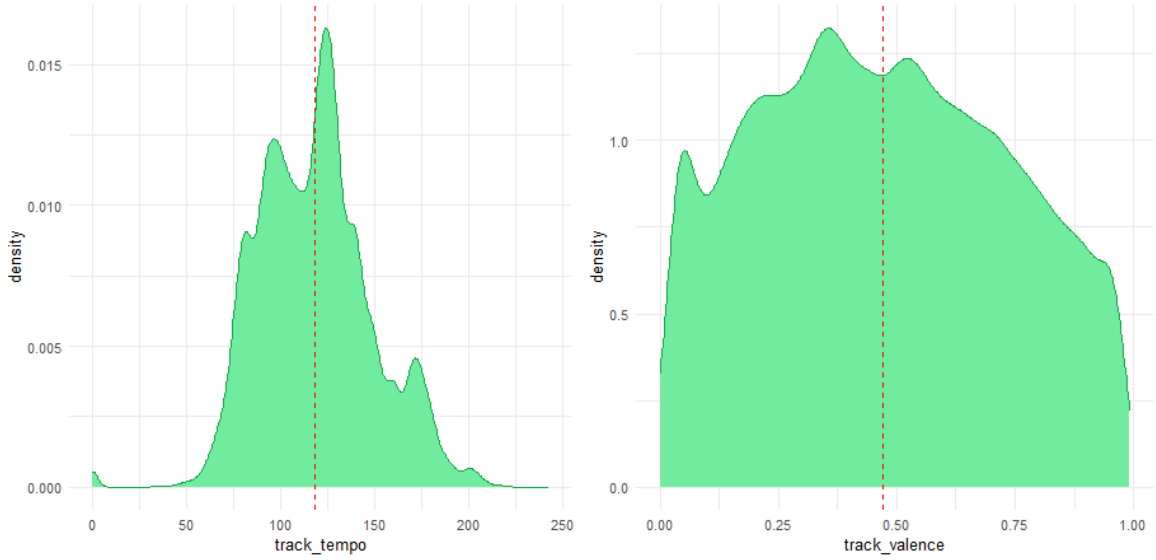


Figura 35. Distribución de las variables *track_tempo* y *track_valence*

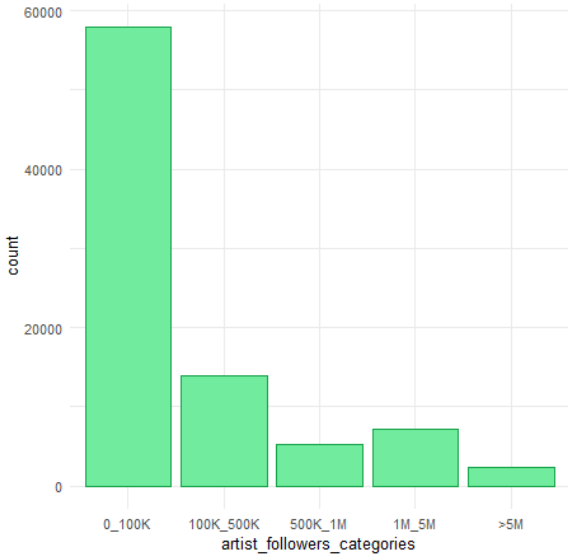


Figura 36. Distribución de la variable *artista_followers_categories*

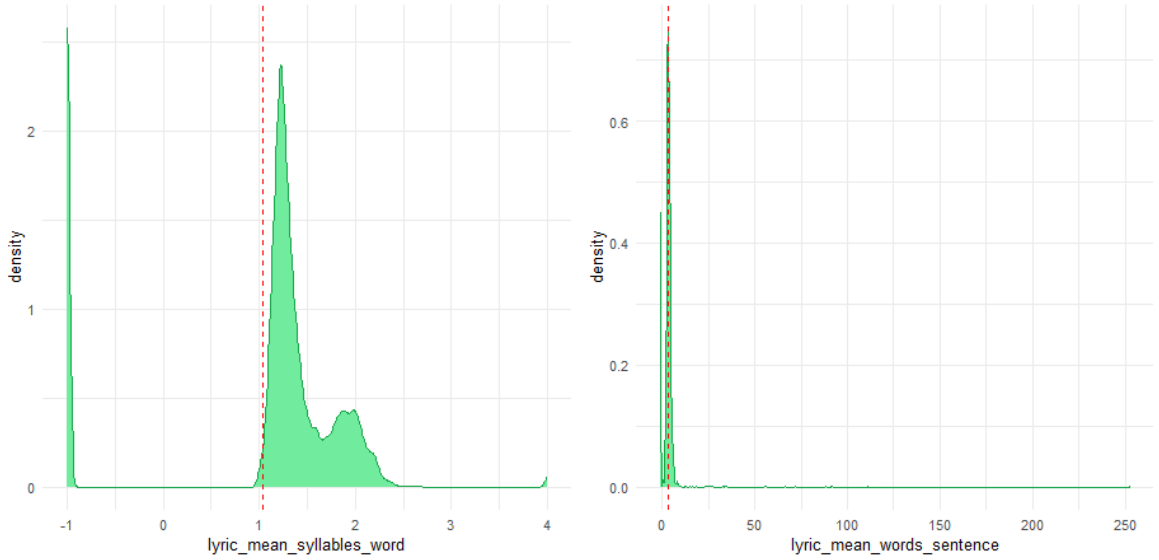


Figura 37. Distribución de las variables *lyric_mean_syllables_word* y *lyric_mean_words_sentence*

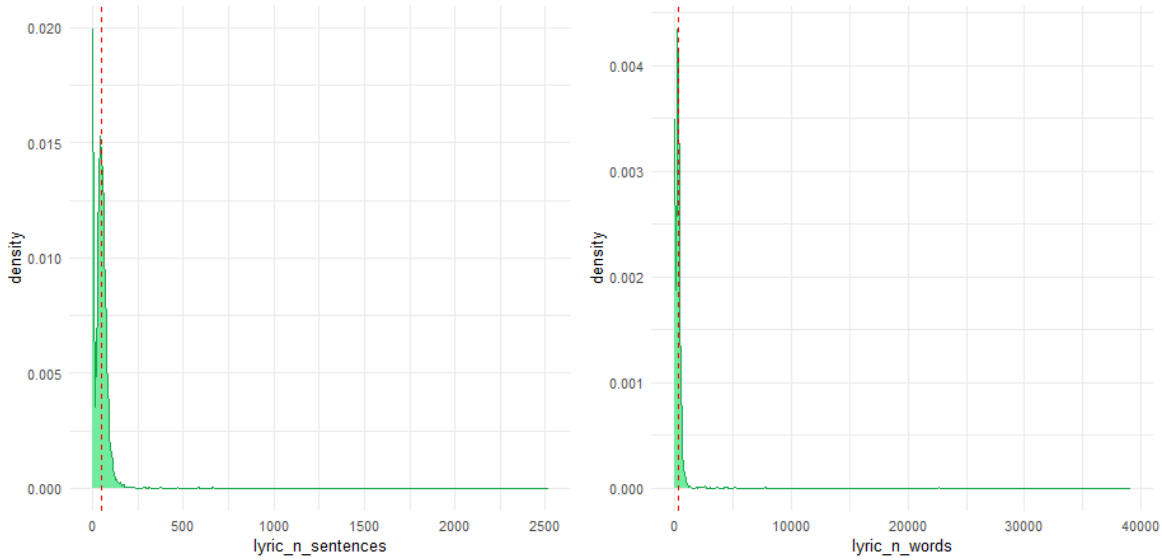


Figura 38. Distribución de las variables *lyric_n_sentences* y *lyric_n_words*

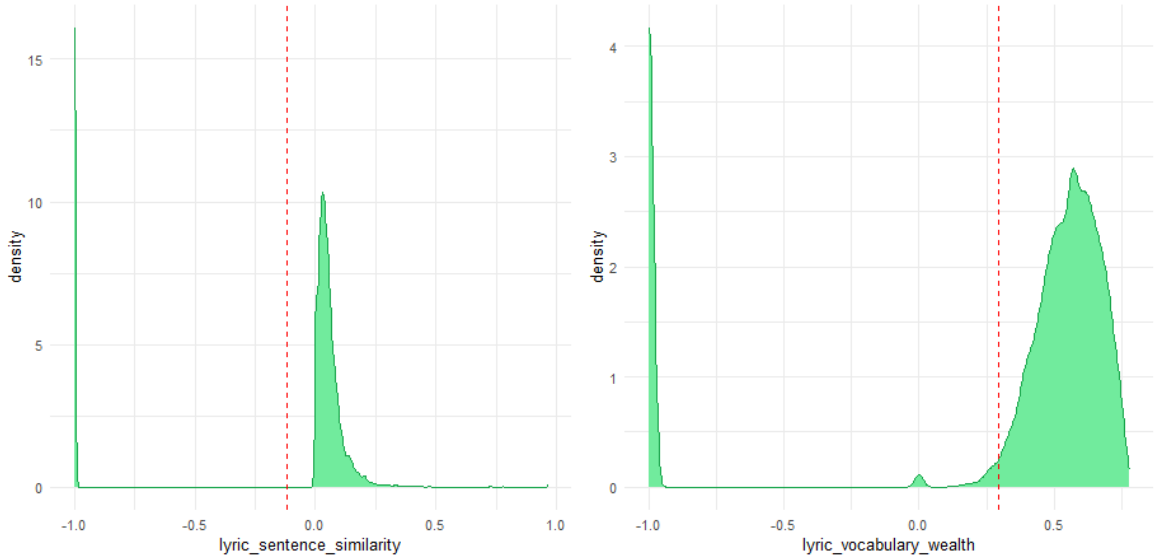


Figura 39. Distribución de las variables *lyric_sentence_similarity* y *lyric_vocabulary_wealth*

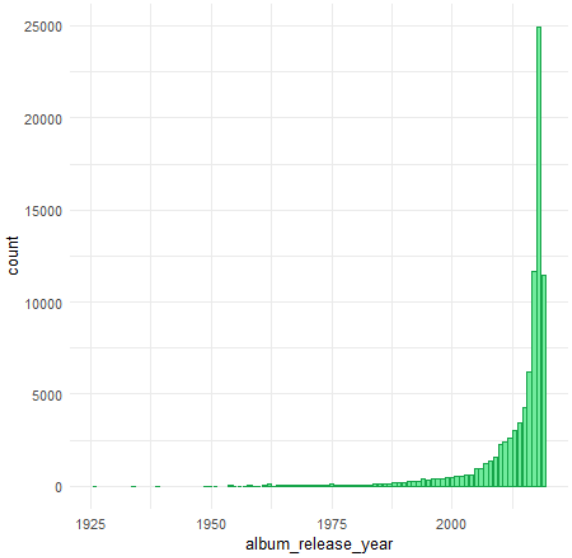


Figura 40. Distribución de la variable *album_release_year*

II. Configuración de los modelos de ensamblado

Ensamblados	Unión de algoritmos
Regresión Lineal (predi1)	Regresión
Redes Neuronales (predi2)	Redes
Bagging (predi3)	Bagging
Random Forest (predi4)	Random Forest
Gradient Boosting (predi5)	Gradient Boosting
XGBoost (predi6)	XGBoost
SVM Lineal (predi7)	SVM Lineal
SVM Radial (predi8)	SVM Radial
Ensamblado 1 (predi9)	Regresión + Redes
Ensamblado 2 (predi10)	Regresión + Bagging
Ensamblado 3 (predi11)	Regresión + Random Forest
Ensamblado 4 (predi12)	Regresión + Gradient Boosting
Ensamblado 5 (predi13)	Regresión + XGBoost
Ensamblado 6 (predi14)	Regresión + SVM Radial
Ensamblado 7 (predi15)	Redes + Bagging
Ensamblado 8 (predi16)	Redes + Random Forest
Ensamblado 9 (predi17)	Redes + Gradient Boosting
Ensamblado 10 (predi18)	Redes + XGBoost
Ensamblado 11 (predi19)	Redes + SVM Radial
Ensamblado 12 (predi20)	Bagging + Random Forest
Ensamblado 13 (predi21)	Bagging + Gradient Boosting
Ensamblado 14 (predi22)	Bagging + XGBoost
Ensamblado 15 (predi23)	Bagging + SVM Radial
Ensamblado 16 (predi24)	Random Forest + Gradient Boosting
Ensamblado 17 (predi25)	Random Forest + XGBoost
Ensamblado 18 (predi26)	Random Forest + SVM Radial
Ensamblado 19 (predi27)	Gradient Boosting + XGBoost
Ensamblado 20 (predi28)	Gradient Boosting + SVM Radial
Ensamblado 21 (predi29)	XGBoost + SVM Radial
Ensamblado 22 (predi30)	Redes + Bagging + Random Forest
Ensamblado 23 (predi31)	Redes + Bagging + Gradient Boosting
Ensamblado 24 (predi32)	Redes + Bagging + XGBoost
Ensamblado 25 (predi33)	Redes + Bagging + SVM Radial
Ensamblado 26 (predi34)	Redes + Random Forest + Gradient Boosting
Ensamblado 27 (predi35)	Redes + Random Forest + XGBoost
Ensamblado 28 (predi36)	Redes + Random Forest + SVM Radial
Ensamblado 29 (predi37)	Redes + Gradient Boosting + XGBoost
Ensamblado 30 (predi38)	Redes + Gradient Boosting + SVM Radial
Ensamblado 31 (predi39)	Redes + XGBoost + SVM Radial

Ensamblado 32 (predi40)	Bagging + Random Forest + Gradient Boosting
Ensamblado 33 (predi41)	Bagging + Random Forest + XGBoost
Ensamblado 34 (predi42)	Bagging + Random Forest + SVM Radial
Ensamblado 35 (predi43)	Bagging + Gradient Boosting + XGBoost
Ensamblado 36 (predi44)	Bagging + Gradient Boosting + SVM Radial
Ensamblado 37 (predi45)	Bagging + XGBoost + SVM Radial
Ensamblado 38 (predi46)	Random Forest + Gradient Boosting + XGBoost
Ensamblado 39 (predi47)	Random Forest + Gradient Boosting + SVM Radial
Ensamblado 40 (predi48)	Random Forest + XGBoost + SVM Radial
Ensamblado 41 (predi49)	Gradient Boosting + XGBoost + SVM Radial
Ensamblado 42 (predi50)	Redes + Bagging + Random Forest + Gradient Boosting
Ensamblado 43 (predi51)	Redes + Bagging + Random Forest + XGBoost
Ensamblado 44 (predi52)	Redes + Bagging + Gradient Boosting + XGBoost
Ensamblado 45 (predi53)	Redes + Random Forest + Gradient Boosting + XGBoost
Ensamblado 46 (predi54)	Bagging + Random Forest + Gradient Boosting + XGBoost
Ensamblado 47 (predi55)	Redes + Bagging + Random Forest + Gradient Boosting + XGBoost

Tabla 23. Configuración de los modelos de ensamblado