FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2019/2020

Trabajo de Fin de Máster

TITULO: Construcción de una herramienta de predicción para la popularidad de las canciones en Spotify

***Alumno*: Cristina Nieto García**

***Tutor*: Aída Calviño Martínez**

Septiembre de 2020

UNIVERSIDAD COMPLUTENSE
MADRID

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

El objetivo de este trabajo es crear un modelo de predicción de la popularidad que tienen las canciones en *Spotify* [1], ya que con ello se puede visualizar la repercusión que puede llegar a tener una canción en los oyentes teniendo en cuenta características de la propia canción, del artista, etc. Para ello, se van a utilizar distintos métodos de Machine Learning, en concreto, redes Neuronales, regresión Lineal y distintos métodos basados en árboles para su posterior ensamblado.

1.1. Motivación

Actualmente, las técnicas de Machine Learning van ganando importancia a nivel empresarial en cuestión de estructuración y automatización. Esto se debe a que cada vez se generan volúmenes más altos de datos que se recopilan y almacenan para poder descubrir comportamientos y establecer patrones o correlaciones para la resolución de problemas y toma de decisiones.

Son numerosos los ámbitos a los que se aplican, desde la salud (detección y prevención de enfermedades), el deporte (previsión de resultados), marketing (algoritmos de recomendación de productos en base a los gustos del usuario), control del tráfico, turismo y multitud de usos en otros sectores [2].

En este análisis realizado, nos centramos en la industria musical. En la actualidad el consumo de música ha cambiado mucho con la llegada de las plataformas de *streaming* de audio, ya que la música está totalmente al alcance del usuario. En este caso, nos centraremos en la aplicación *Spotify*.

Para analizar esta industria desde el punto de vista de aplicación de algoritmos de aprendizaje, se plantea el problema teniendo en cuenta que la música estimula el cerebro humano haciendo que genere dopamina (hormona relacionada con el placer) [2].

Teniendo en cuenta esto, se han realizado algunos estudios en los que mediante técnicas de Machine Learning, han relacionado los sentimientos de los usuarios con la música, con el fin de determinar el éxito que podía tener ciertas canciones entre el público, o incluso el devenir de la economía analizando lo que se escucha en *Spotify*. Estos ejemplos se desarrollarán más en el Capítulo 2 [2,3].

Desde el punto de vista técnico, se van a entrenar modelos de regresión, redes neuronales y algoritmos basados en árboles y después compararlos para ver sus

ventajas e inconvenientes de aplicación, e intentar elegir el que mejor se ajuste a los datos y logre predecir con menor error.

1.2. Objetivos

El objetivo principal es proponer un modelo de predicción para la popularidad de una canción de *Spotify* mediante distintas técnicas de Machine Learning. Se van a extraer características de las canciones, del artista que las interpreta y del álbum al que pertenece. Nos basaremos para todo el proceso en la metodología de minería de datos SEMMA: muestreo, exploración, y modificación de los datos para posteriormente crear los modelos de predicción para finalizar evaluándolos.

Específicamente, los objetivos secundarios de este trabajo son:

- Creación de una base de datos de canciones relevante a través de la API de *Spotify* usando el lenguaje de programación Python.
- Realizar un análisis exploratorio en SAS Miner sobre la base de datos para poder realizar su depuración y destacar aspectos importantes de las variables.
- Depurar los datos también en SAS Miner para adecuar las distintas variables de entrada útiles, para su posterior análisis.
- Probar distintas selecciones de variables en SAS para optimizar los modelos de Machine Learning y escoger las variables que más aportan información.
- Utilizar métodos de aprendizaje estadístico en el lenguaje de programación R: regresión lineal, redes neuronales y modelos basados en árboles.
- Evaluación de todos los modelos entrenados con la finalidad de escoger el mejor modelo de predicción.

1.3. Estructura de la memoria

Para finalizar este capítulo de introducción, la memoria del actual proyecto se estructura de la siguiente manera:

- En el Capítulo 2 se realiza una revisión bibliográfica de la industria musical, explicando las plataformas de *streaming* de audio como *Spotify*, y se exponen algunos ejemplos de aplicación de Machine Learning en el ámbito de la música.
- En el Capítulo 3 se recoge la explicación y argumentación sobre los tipos de métodos de aprendizaje automático que se van a utilizar para predecir nuestra variable objetivo.
- En el Capítulo 4 se detalla el proceso de creación de la base de datos, las variables que vamos a utilizar en nuestro análisis y el proceso de depuración llevado a cabo.

- En el Capítulo 5 se detalla el estudio realizado con la creación de los modelos y se compararán y evaluarán.

- En el Capítulo 6 se exponen las conclusiones extraídas del análisis y estudio completo llevado a cabo en el anterior capítulo y se proponen líneas de investigación a realizar en un futuro.

Este documento contará con Anexos del código utilizado, pero también se puede consultar en GitHub: https://github.com/crisng4/TFM

CAPÍTULO 2

Estado del arte

En este capítulo vamos a exponer la evolución sobre la industria musical. También se hará un repaso de las plataformas de *streaming* actuales, terminando centrándonos en *Spotify* al ser la plataforma de la que hemos obtenido los datos para este estudio.

Para terminar el capítulo, se van a mencionar algunos ejemplos en los que se aplicaron técnicas de Machine Learning del sector de la música para poder entender y relacionar el uso de métodos para tratar los datos en este ámbito.

2.1. Evolución de los sistemas de reproducción de música

La música, desde los inicios de la historia del ser humano, ha sido una forma de expresión, comunicación y una herramienta cultural que ha ido evolucionando paralelamente a los avances tecnológicos, siendo un elemento clave en el ocio de nuestra sociedad.

El primer dispositivo capaz de reproducir y grabar música llamado **fonógrafo**, fue inventado por Tomas A. Edison en 1877, de esta forma se inició la comercialización de las grabaciones de los cantantes y músicos. En 1925 aparece el **tocadiscos**, dispositivo de reproducción analógica, que permitía la reproducción de discos de vinilo de forma eléctrica y no mecánica.

En el año 1963 Philips inventó el **cassette**, dispositivo icónico de las décadas de los 80 y 90, que consumía poca energía y podía ser transportado fácilmente. Posteriormente se empezaron a fabricar y llegaron los primeros equipos portátiles, siendo en 1979 cuando se inventó el **Walkman** gracias a Sony [4,5].

A partir de 1990 comenzó la venta de los discos compactos, más conocidos como **Compact Disc (CD),** convirtiéndose en el nuevo formato digital para grabar música, vídeos, imágenes y hasta texto. Desde entonces el CD predominó en el mercado de la música dada la buena calidad de grabación, el tamaño y durabilidad. A raíz de la gran acogida de los CDs, se inventaron los **Discman**, un dispositivo portátil con pilas y auriculares, que te permitía reproducir y escuchar todo tipo de CDs en cualquier lugar y momento [4].

En el año 1995 llegó el **reproductor MP3**, un formato de audio digital comprimido en el cual se puede grabar o introducir sonido en una memoria interna por medio de archivos en formato MP3 o similares, mediante conexión USB a un ordenador. Uno de los principales impulsores de la reproducción de música en dispositivos más pequeños, ligeros y cómodos fue la marca Apple con la creación y fabricación de los **iPod**. Con el transcurso de los años y el avance tecnológico, se modificaron los reproductores MP3

para dar paso a los reproductores **MP4** donde se podía introducir y reproducir, música, vídeos e imágenes [4,5].

Actualmente, gracias a los **smartphones** disponemos de todos los dispositivos con multitud de aplicaciones en nuestros teléfonos móviles. Con la revolución de la tecnología y la aparición del nuevo modelo de consumo de música "music on demand", encontramos las plataformas de ***streaming*** [4,5].

Este modelo nació con Napster (1999), precediendo a las aplicaciones de descarga de música gratuita, o MySpace, que nació como red social de grupos musicales emergentes. Posteriormente Kazaa relevó a Napster.

La primera tienda de música online fue iTunes sentando el precedente del modelo actual en el que podemos escuchar música por internet gracias a servicios de streaming por suscripción como ***Spotify***, **Amazon Music**, **Youtube** o **Deezer**, entre otros [6,7].

2.1.1. Plataformas de *streaming*

Inicio del *streaming*

La historia del *streaming* ha ido ligada a la evolución de internet, así como de las conexiones de banda ancha, la comercialización de internet a precios asequibles para mayor cantidad de usuarios, entre otros factores. El primer eslabón de la música en *streaming*, podrían considerarse los primeros canales de radio**,** ya que no se necesitaba tanta velocidad para sintonizar el audio de manera fluída y la forma más cercana que existía del término *en vivo*. Posteriormente esta difusión evolucionó al vídeo, pero al depender de la velocidad de transmisión y de banda, era para un número limitado de personas (2 o 3) y eventos máximos de 4 horas [8].

En 1997 la banda Severe Tire Damage, hizo historia al transmitir su concierto en directo a todo el mundo, y posteriormente la transmisión de un concierto de la sinfónica en el Paramount Theater. Todos estos hechos se corresponden con la antesala de lo que hoy se corresponde con las plataformas de *streaming*, siendo en 2005 clave en la proliferación de los vídeos gracias al lanzamiento del portal de Youtube [8,9].

***streaming* musical: principales competidores**

El *streaming* del género musical es, al igual que en las películas y series, un ámbito lleno de competidores entre los que destacan ***Spotify*, Apple Music, YouTube Music, Deezer y Amazon Music**; servicios que ofrecen en líneas generales el mismo producto: música online mediante una tarifa plana**.**

Spotify ha presentado los resultados correspondientes a 2019, observando el crecimiento de sus suscriptores, a pesar de que la rentabilidad del último trimestre ha sido negativa. Cierra el 2019 con 73 millones de euros en pérdidas, aunque el número de usuarios activos mensuales, tanto gratuitos como de pago, no ha dejado de crecer.

A día de hoy sigue siendo el máximo exponente en todos los sentidos. Posee 124 millones de usuarios que pagan la suscripción, ya sea con cuenta única o compartida.

Le sigue **Apple Music**. La compañía revela en ocasiones los datos de sus suscriptores, siendo los últimos que hay disponibles de 60 millones de usuarios activos mensuales. Todos ellos son de pago, al no ofrecer la compañía una opción gratuita como *Spotify*.

Los tres proveedores de *streaming* con menos suscriptores son Amazon Music, YouTube Music y Deezer. **Deezer** tiene en activo, según los últimos datos disponibles, 7 millones de usuarios de pago y 14 millones de usuarios que usan la modalidad gratuita, que solo permite escuchar música en modo aleatorio. Ha sido superado por **YouTube Music**, que en 2019 ha conseguido sumar 20 millones de suscriptores, aunque cuando los datos se publicaron, estaban contabilizados en el mismo conjunto usuarios de YouTube Premium y YouTube Music, cuando la primera suscripción incluye a la segunda.

Por último, **Amazon Music**, cuenta con 55 millones de suscriptores de pago sumando todas las modalidades ya que es un proveedor de varios servicios, quedando muy cerca de Apple Music [9].

En la Figura 2.1 podemos observar la tendencia creciente de los ingresos procedentes de los servicios de la música en *streaming* a nivel mundial entre 2010 y 2019.

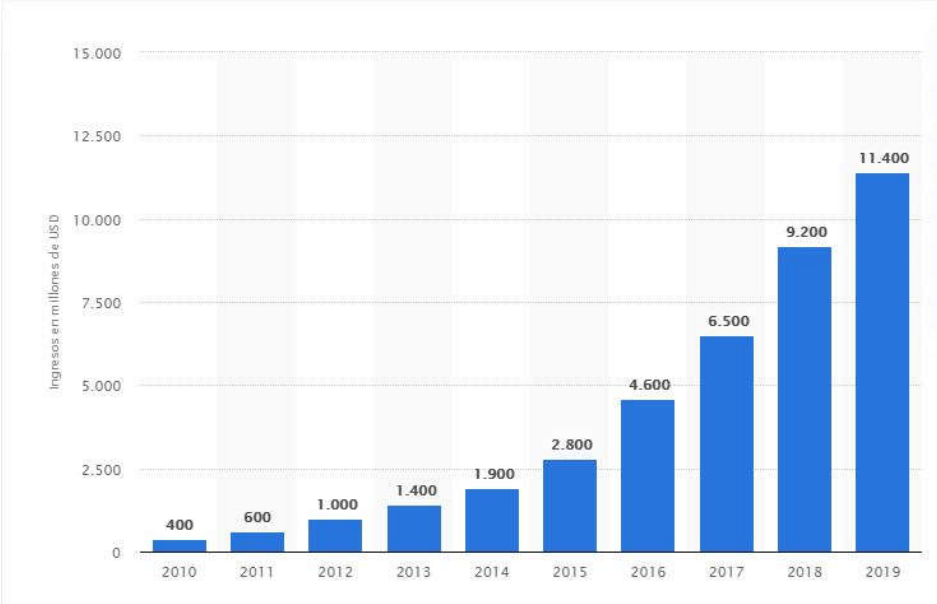

Figura 2.1: Ingresos (en dólares estadounidenses) de servicios de música en *streaming* por año en el mundo [10]

En la Figura 2.2, podemos ver reflejados los datos anteriormente mencionados en referencia a usuarios de pago de las plataformas analizadas, siendo *Spotify* la líder, y Deezer la que menor crecimiento experimenta. Cabe destacar que ***Spotify* acumula 271 millones de usuarios activos mensuales**, aunque en la gráfica solo están contemplados los usuarios Premium, dado que el resto de servicios son mayoritariamente de pago [9].

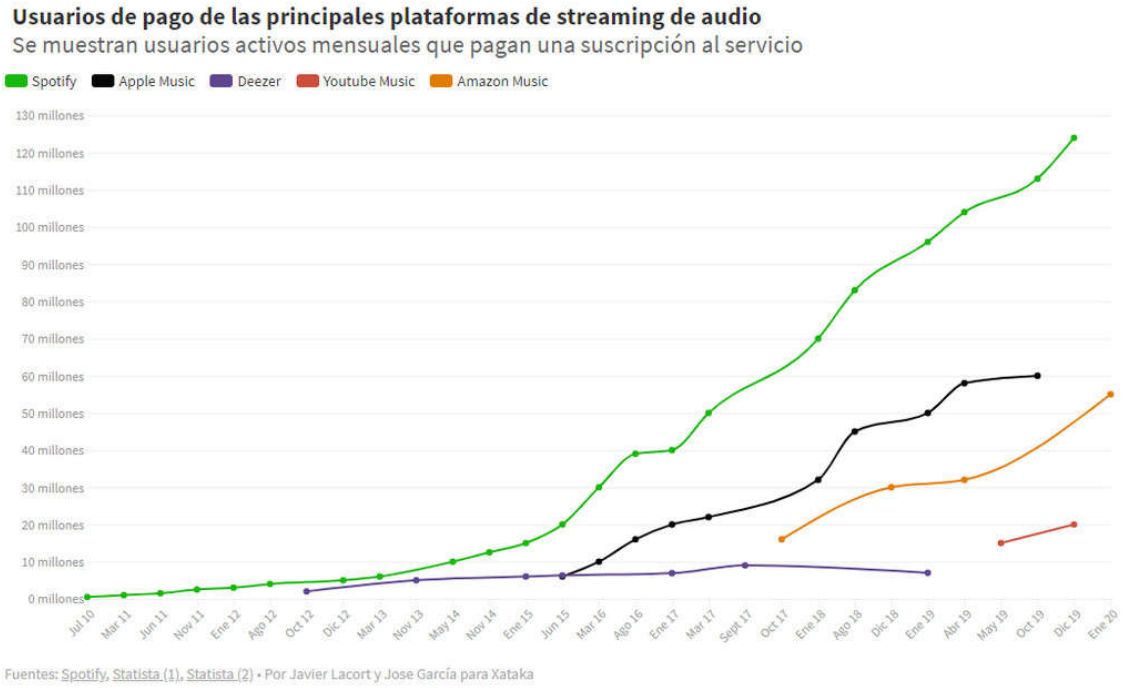

Figura 2.2: Usuarios de pago (en millones) de las plataformas de *streaming* de audio principales en la última década [9].

2.1.2. *Spotify*

De origen sueco, fue creado en 2006 por Daniel Ek, aunque finalmente su puesta en marcha de forma pública se produjo en octubre de 2008, concibiéndose como una aplicación para escuchar música online por ordenador sin necesidad de descarga. Solo estaba disponible, en principio, para algunos países europeos como Suecia, Finlandia, Noruega, Francia, Reino Unido y España. Hasta noviembre de 2011 no llegaría a EEUU.

Como se ha mencionado en el anterior apartado al analizar las modalidades que ofrece la plataforma, a día de hoy, *Spotify* dispone de la versión Free y Premium. La modalidad Free es gratuito para los usuarios. En un inicio disponía de un límite máximo de 20 horas mensuales de escucha, el cual fue eliminado en 2014 a cambio de publicidad esporádica, un número limitado de saltos de canciones y no poder acceder a la máxima calidad del audio. La versión Premium, de pago, te permite disfrutar de música sin anuncios, la posibilidad de descargar música, y la opción de pasar de canción sin límites, entre otros beneficios [11].

En la Figura 2.3 podemos observar en una gráfica los usuarios activos de la plataforma y los suscriptores Premium, mostrando un crecimiento progresivo de los usuarios de pago, los cuales se va acercando a la de los usuarios en activo; cumpliendo así, el objetivo de la plataforma:

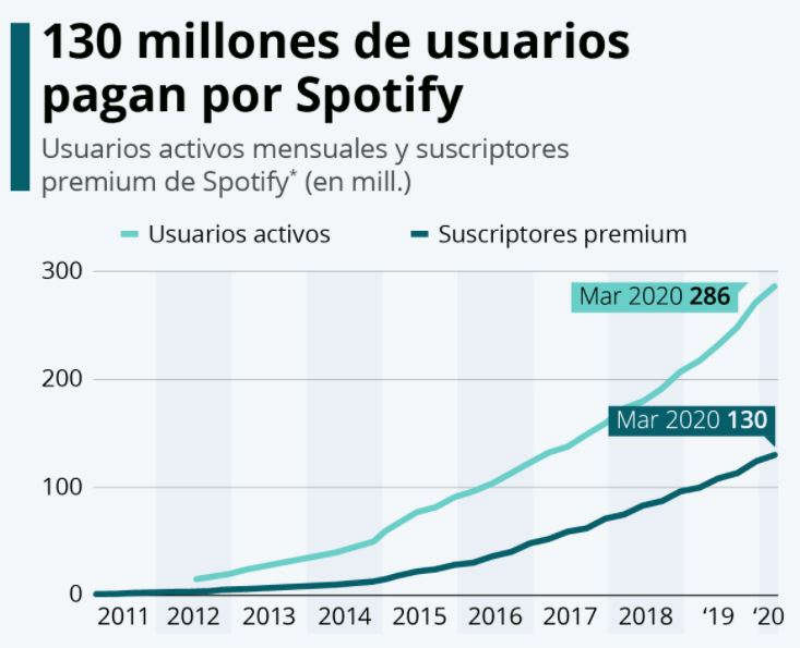

Figura 2.3: Usuarios activos mensuales y Premium (en millones) de las plataformas de *streaming* de audio principales en la última década [12].

Con la evolución de los años, *Spotify* ha tenido que adaptarse al impulso de la redes sociales**,** aliándose en 2016 con Facebook para que sus usuarios pudieran compartir música a través de Messenger; y llegando a un acuerdo con Twitter para que se puedan escuchar canciones de *Spotify* sin salir de la misma red social a través de tweets.

Entre otros servicios que ofrece la plataforma, ratificó en 2016 el éxito de su lista "Discover Weekly", que se basa en los descubrimientos semanales adaptados a cada tipo de usuario, los cuáles se generan según un algoritmo de recomendación que analiza sus gustos y búsquedas musicales. De esta forma ofrece un servicio único y personalizado a los usuarios [9].

***Spotify*, el reto de la rentabilidad**

Con todos los datos analizados previamente, *Spotify* siendo la plataforma más veterana, sigue muy delante de su competencia respecto al valor de usuarios, aunque no significa que obtenga una mayor rentabilidad**.** En el último trimestre de 2018, la empresa consiguió, por primera vez en su historia, ser rentable, pero los resultados finales muestran que la relación beneficio/pérdidas ha sido incluso peor que en 2018.

La compañía ha cerrado 2019 con 6.764 millones de euros en ingresos, una cifra significativamente mayor que en los años anteriores, pero no se ha traducido en beneficio, puesto que la empresa ha perdido 77 millones de euros, siendo más notorias las pérdidas que en 2018. El beneficio bruto de la empresa fue de 1.722 millones de euros en 2019, pero a eso hay restarle 615 millones de desarrollo e investigación, 826 millones en ventas y marketing y 354 en gastos administrativos. El total de todo esto se corresponde con 1.795 millones de euros**.**

Uno de los segmentos en los que más ha crecido la empresa este año ha sido el de los podcasts, que ha experimentado un crecimiento del 16% en usuarios y un 200% en horas de escucha. Este servicio ha demostrado ser un buen mecanismo de retención de usuarios gratuitos que, en última instancia, acaban pasando a la versión Premium. Estiman que cerrarán el año 2020 con entre 328 y 348 millones de usuarios, los cuales entre 143 y 153 millones serán Premium, y prevén entre 150 y 250 millones de euros en pérdidas [9].

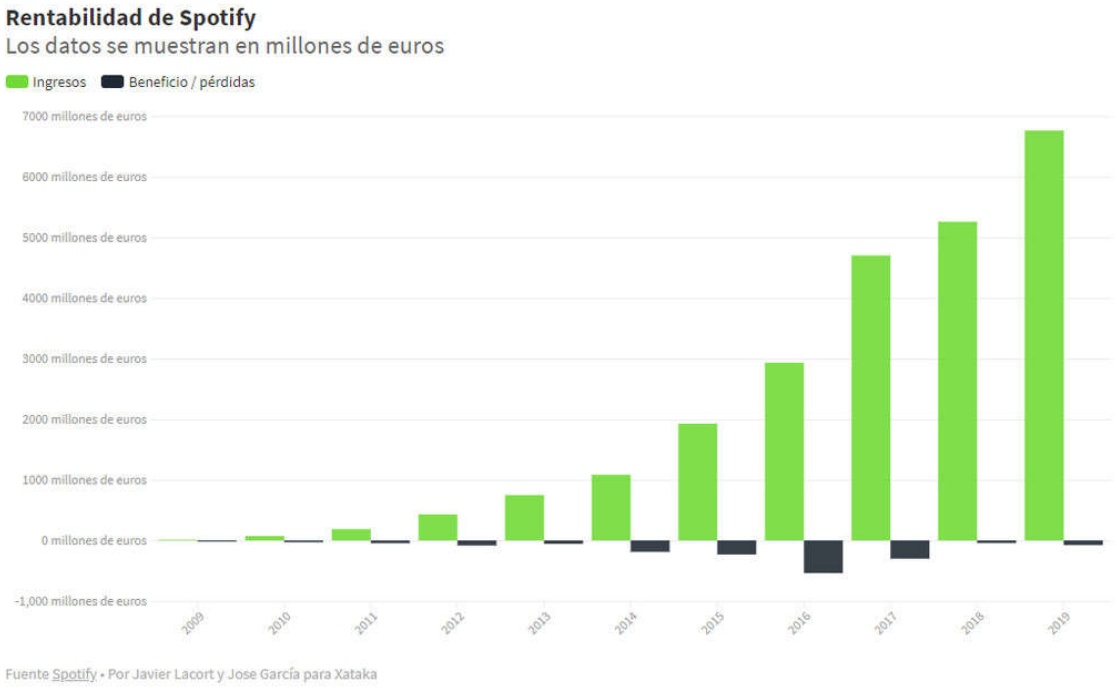

Figura 2.4: Rentabilidad de *Spotify* (Ingresos Vs Beneficios/Pérdidas) en la última década [9].

2.2. Aplicaciones de Machine Learning en la música

La inteligencia humana sigue siendo la pieza clave creativa en la música. La Inteligencia artificial se utiliza para reducir el tiempo que los compositores y productores necesitan dedicar a tareas repetitivitas y, una vez más, para utilizarlo como herramienta para ayudar a impulsar nuestras capacidades humanas.

Big Data**,** o almacenamiento masivo de datos, permite conocer al milímetro las tendencias de la sociedad. La industria musical es una de las que hacen un mayor uso de esta ventaja, siendo *Spotify* e iTunes, una fuente de información para las discográficas que, gracias a los servicios de música en *streaming*, consiguen saber qué canción o cantante puede ser un éxito de ventas. Además de esto, pueden a su vez localizar fans de un artista emergente: analizando las opiniones e impactos que éste causa en las redes sociales, las visitas de sus videos tanto oficiales como no oficiales de Youtube, las reproducciones en las plataformas de *streaming* y las compras de su producto [13].

Predicciones económicas analizando la música

El Indicador de Sentimiento Económico es una variable que utilizan los economistas para conocer la confianza de los consumidores y los diversos sectores en la economía, lo que sirve para hacer predicciones sobre como responderán estos a diferentes políticas.

Desde 1990, la Unión Europea elabora un Indicador de Sentimiento Económico realizando un cuestionario sobre la confianza en el mercado en los sectores del comercio al por menor, la industria, los servicios, los consumidores, la construcción y el comercio minorista; pero quizás debería tener en cuenta otras variables. Últimos estudios realizados, revelan que puede predecirse el comportamiento económico de un país según la música que escuchan los habitantes [3].

Como explican en *The Conversation* Kim Kaivanto y Peng Zhang (profesores de las universidades de Lancaster y Guizhou Minzu, respectivamente), las canciones tienen un componente emocional con el que cualquiera pueda identificarse, codificado en atributos musicales como la energía, el tempo y el volumen. Los proveedores de *streaming,* como *Spotify*, usan este tipo de variables para categorizar canciones según los gustos de los usuarios y recomendar música nueva y sugerir playlist o géneros [13,14].

Otra variable que cabe destacar son las letras, que pueden analizarse mediante el "procesamiento de lenguaje natural". El programa codifica la carga emocional positiva o negativa de las palabras, empareja las palabras con ocho emociones: alegría, tristeza, ira, miedo, disgusto, sorpresa, confianza y expectación. Posteriormente, se realiza el recuento de veces que cada emoción aparece en una canción.

Con todos estos datos, los investigadores pueden realizar un marco situacional de los sentimientos de los usuarios y extrapolar el sentimiento económico, analizando las 100 canciones más escuchadas en un mes. Reflejan resultados más concluyentes que los cuestionarios, ya que reflejan las elecciones reales y nos ofrecen una muestra más amplia.

Los investigadores de Claremont aplicaron esta técnica a las listas de éxitos de antes y después de la crisis de 2008. Descubrieron que, después de la crisis, aumentó la frecuencia de las palabras asociadas a la ira y el disgusto y disminuían las relacionadas con la confianza. Por lo tanto, los estados anímicos de los usuarios influyen en su elección musical, extrapolando los datos a la economía incluso a corto plazo [3].

Además de estos datos, *Spotify* tiene conocimiento de los hogares y ubicación de los suscriptores, lo que serviría para elaborar índices de opinión para diferentes regiones y grupos de personas basados también en el nivel adquisitivo [3].

“Despacito” de Luis Fonsi: un ejemplo de Machine Learning

La canción de “Despacito” se convirtió en el primer vídeo en YouTube, con más de tres millones de visitas. Se colocó en el primer lugar de la lista de las más escuchadas en *Spotify*, la primera canción latina en llegar a la cima; sumándose la versión de Justin Bieber [14,15].

La clave, según el neurocientífico Daniel Mullensiefen, se basa en algunos de los elementos que comparte con casi todos los éxitos del verano: ser más rápida que el promedio de las canciones, produce recuerdos y crea conexiones emocionales.

Otro elemento destacable de esta canción ha sido la repetición. Un estudio publicado en “PlosOne” y realizado por expertos de la Universidad de Oporto (Portugal), escaneó el cerebro de decenas de voluntarios y descubrió que estos se conectaban emocionalmente más a un tema cuando los sonidos les resultaban familiares y repetitivos. La repetición también influye en el efecto viral.

Todo ello lleva a plantear la cuestión de si existe la fórmula para crear los éxitos del verano, gracias a *Spotify* y The Echos Nest, empresa de Big Data Musical, junto con Firts Choice para crear una fórmula de la canción del verano perfecta. De acuerdo con sus resultados, para alcanzar el éxito hay que lograr un equilibrio entre cinco variables: ritmo, cuan bailable es la canción, alegría, acústica y energía. Uniendo todas ellas, los responsables del análisis concluyeron que cuantos menos instrumentos eléctricos tenga, menos voces tratadas digitalmente y más percusión, más éxito generará la canción.

Tras analizar 150 de las canciones más escuchadas en los últimos cinco veranos, en una base de más de 30 millones de piezas musicales, los expertos llegaron a la conclusión de que la ecuación perfecta sería la siguiente:

Ritmo + Energía x 1.48 + Bailable x 1.17 + Acústica x 0.17 + Alegría x 1.14 = ***canción del verano***

(2.1)

Muchos expertos coinciden en que el factor clave tras el éxito de Despacito se basa en el estribillo, una ruptura intencional, desglosando la palabra y el énfasis sobre el mismo, es decir, la ruptura del tempo justo antes de la entrada del estribillo en la que se canta “Des-pa-ci-to”.

Además de esto, lo que proporciona a la canción el éxito no se corresponde solo con las variables previamente mencionadas, sino que es esencial que exista una historia con la que la gente se identifique. Este aspecto sería clave para ser capaces de provocar la reacción y emoción esperada en los oyentes, y no todos los estudios contemplan dicho aspecto [14,15].

CAPÍTULO 3

Metodología

En este capítulo se van a explicar las técnicas y modelos de Machine Learning que se han aplicado para intentar resolver el problema que se plantea: predecir la popularidad de una canción. Tenemos en cuenta que es un problema de regresión al ser cuantitativa nuestra variable objetivo; y de aprendizaje supervisado, ya que se entrena el modelo con datos de entrada asociados a datos de salida para buscar patrones de comportamiento.

3.1. Regresión Lineal

Este modelo trata de predecir una variable dependiente o de repuesta a partir de ciertas variables independientes tratando de explicar relaciones lineales entre ellas. Se trata de explicar las variables la relación que hay entre las variables de respuesta y las explicativas a través de una función lineal.

De forma analítica, su ecuación sería:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_m X_m + \mathcal{E} \quad (3.1)$$

- Y: se corresponde con la salida del modelo, y es una variable aleatoria continua.
- $\beta 0$: es el valor de la variable dependiente Y cuando todas las variables independientes son cero.
- βi : representa el efecto promedio que tiene el incremento en una unidad de la variable dependiente Xi sobre la variable de respuesta, manteniéndose constantes el resto de variables. Se conocen como coeficientes parciales de regresión.
- $\mathcal{E}$: es el error cometido, la diferencia entre el valor real y el estimado por la regresión lineal, es decir, la parte de la variable de respuesta que no se puede explicar con las variables independientes.

Para el aprendizaje del modelo, y así realizar la predicción deseada, es necesario estimar el valor de los parámetros con la técnica de los mínimos cuadrados. Se busca minimizar el error cometido por el modelo (diferencia entre el valor real y el estimado), por ello, con este estimador se minimiza la suma del cuadrado de los errores. A la hora de estimar los parámetros, hay que tener en cuenta que las variables independientes no estén muy correlacionadas entre sí y que el número de parámetros ha de ser muy inferior al número de observaciones [16, 17].

Regresión lineal es un buen algoritmo porque es robusto, rápido y útil cuando la relación entre las variables objetivo y las independientes no es demasiado compleja. También es menos propenso al sobreajuste.

En el siguiente apartado, se va a explicar el método de selección de variables que realizaremos posteriormente a nuestra base de datos.

3.1.1. Selección de variables

Antes de comenzar a probar los modelos que se van a describir en este capítulo, en el Capítulo 5 de modelización, se van a realizar distintas selecciones de variables mediante regresión lineal para ver cuál es la mejor para probar los modelos. Se van a detallar los criterios seguidos para crear dichas variables.

Existen 3 métodos de selección de variables [16]:

- **Forward o hacia delante:** se parte desde cero, y se va introduciendo una a una las variables que se determinen que producen una mejora en el modelo hasta que todas las variables que aportan información estén dentro de la selección. Destacar que una vez que una variable entra en el modelo, no puede salir.

- **Backward o hacia atrás:** en este método se parte teniendo todas las variables, y se van eliminando las que influyan menos en el modelo una a una, hasta que se queden las variables importantes en el modelo. Cuando una variable se elimina no puede entrar de nuevo en el modelo,

- **Stepwise o paso a paso:** este método es una combinación de los anteriores. Es como el método forward pero se pueden eliminar las variables que entran si no aportan información al modelo. A la hora de eliminar las variables que no son importantes, se hace como en backward. En cada iteración se evalúan todas las variables.

Para los anteriores métodos de selección, se van a utilizar los siguientes criterios de información:

- **AIC:** criterio de información de Akaike
- **BIC:** criterio de información bayesiano
- **SBC**: criterio de Schwarz

3.2. Redes Neuronales

Las redes neuronales se diseñaron basándose en el funcionamiento de las redes biológicas del cerebro humano, imitando su proceso de aprendizaje. Las neuronas biológicas están formas por tres partes: dendritas, el soma y el axón. Las dendritas captan impulsos nerviosos emitidos por otras neuronas, que posteriormente se procesan en el soma y se transmiten a otras neuronas. Este proceso se llama sinapsis y se liberan neurotransmisores. Dependiendo del neurotransmisor liberado, la neurona que recibe el impulso eléctrico puede excitarse o inhibirse, por los que la neurona receptora puede que responda, o no, con otro estímulo.

Volviendo a las redes neuronales artificiales, se define ese impulso nervioso que se trasmite entre neuronas, como la suma de las entradas a la neurona multiplicadas por sus pesos correspondientes w_i. Esta suma se procesa en la neurona con la función de activación, que determina el valor que transmitirá a otra neurona si se activa. Éste valor, se procesa en el interior de la célula mediante una función de activación que devuelve un valor que se envía como salida de la neurona. Resumiendo, una red neuronal se modeliza: $Y = f(X_1, X_2, X_3, \ldots, X_i)$ siendo f suele ser una función no lineal.

Para poder comprender y visualizar el funcionamiento de las neuronas, y las analogías entre las biológicas y artificiales en cuanto a su funcionamiento, a continuación, se muestran dos imágenes sobre ello. En la primera (Figura 3.1) se pueden observar las distintas partes de una neurona biológica, y en la segunda figura (Figura 3.2) se observa el esquema con el proceso que sigue una neurona artificial [18].

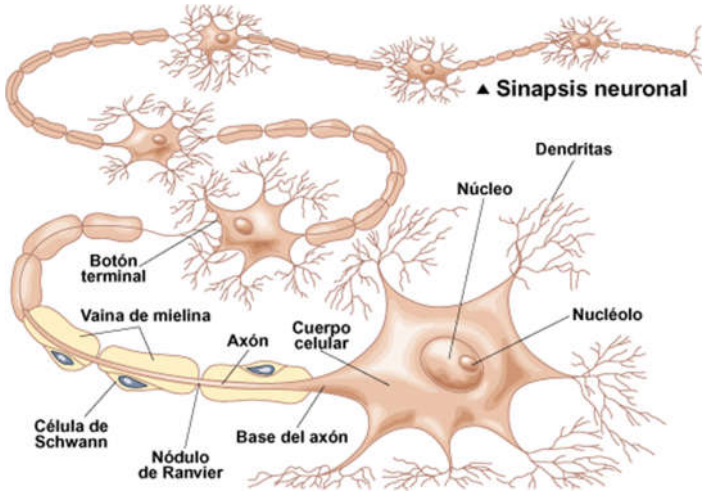

Figura 3.1: Partes de una neurona biológica. [19]

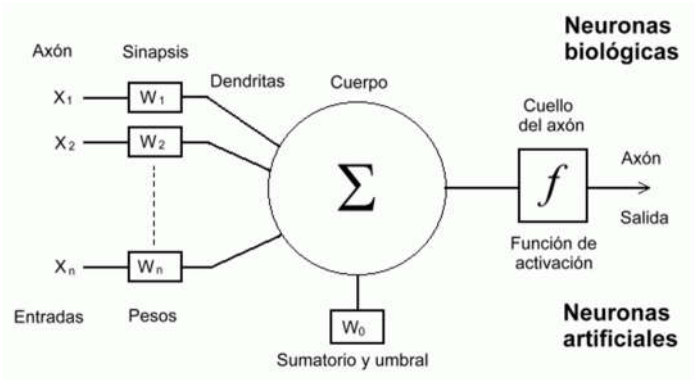

Figura 3.2: Esquema de una neurona artificial junto con las partes del proceso biológico. [20]

Una vez comprendido cómo funciona el proceso de sinapsis, vamos a analizar su estructura. Las neuronas se agrupan mediante capas como podemos ver en la Figura 3.3.

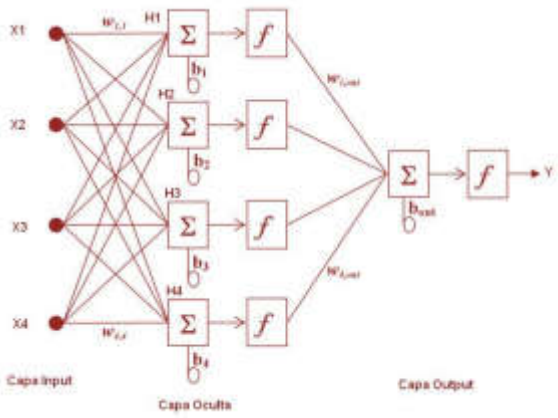

Figura 3.3: Estructura de un ejemplo de una red neuronal [18].

La capa de entrada está compuesta por las variables de entrada, las capas ocultas se encuentran intermedias entre la capa de entrada y de salida (donde se genera la salida del algoritmo con la variable respuesta.

Las ventajas de las redes neuronales es que se adaptan bien a un número elevado de variables input, y se pueden tener varias variables output, además de tratar la multicolinealidad. Las desventajas que presenta es que necesita mucha información, hay que controlar la convergencia de los algoritmos, y se puede demorar el tiempo de procesado.

A continuación, se van a mostrar los parámetros que se suelen utilizar para configurar redes neuronales [18]. En este trabajo no se pueden utilizar algunos de ellos porque vienen configurados por defecto en R:

- *Size*: número de nodos en las capas ocultas.
- *Función de activación*: es la función que define la salida de un nodo dada una entrada o un conjunto de entradas. Este parámetro se puede variar en SAS con varias funciones de activación (tangente, lineal, logaritmo), mientras que en R sólo se utiliza por defecto, para la función de *avnnet*, la función *tanh* (tangente).
- *Algoritmo de optimización*: se encargan hallar pesos para minimizar la función de error. Este parámetro tampoco se puede variar en R, cuyo valor por defecto es *Broyden-letcher-Goldfarb - Shanno (BFGS).* Mientras que en SAS se pueden asignar varios algoritmos de optimización (LEVMAR, BROP, QUANIEW).
- *weight decay o learning rate*: indica el costo del algoritmo de optimización para actualizar los pesos.

3.3. Modelos basados en árboles

Estos algoritmos engloban un conjunto de técnicas supervisadas no paramétricas que, de forma iterativa, dividen los datos en regiones simples basadas en intervalos de los valores de las variables independientes con un enfoque de división binaria. Para ello, se

divide la población en conjuntos homogéneos en función de la variable independiente más significativa. Es necesario predefinir algunos parámetros antes para definir la construcción de dicho árbol y cuando finalizar el modelo [18,21].

Las partes más significativas de un árbol podemos verlas (en la Figura 3.4) [16]:

• Un nodo y el conjunto de sus sucesores recibe el nombre de rama.

• Los nodos en los que se divide él mismo son nodos hijos.

• El nodo predecesor de un nodo es el padre.

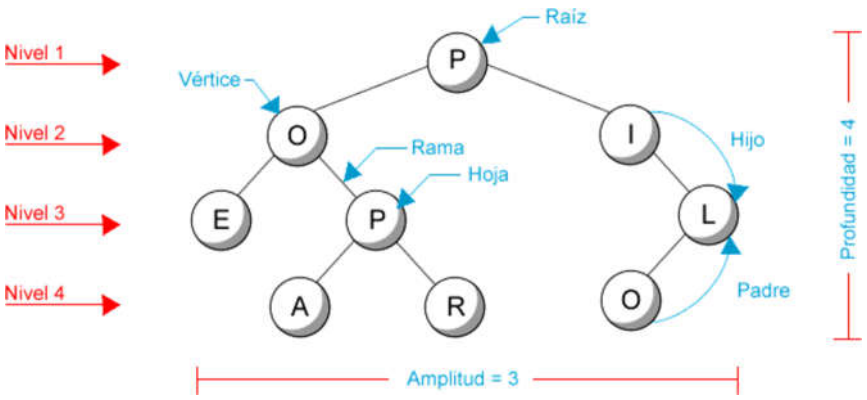

Figura 3.4: Partes y estructura de un algoritmo basado en árboles

[22].

Estos algoritmos procesan bien datos faltantes o categorías infrarrepresentadas, pero no cuentan con gran capacidad predictiva y tienden al sobreajuste de los datos [18,23].

3.3.1. Bagging

Antes de explicar este algoritmo, vamos a explicar el término Bootstrap. Es una técnica que se basa en estimar un número de datos a partir de una muestra para posteriormente promediar las predicciones de las muestras [24].

Entendiendo esto, Bagging (Bootstrap Aggregation) combina las predicciones de varios algoritmos de aprendizaje automático para realizar predicciones más precisas que cualquier modelo individual. Se dividen los datos en muestras para predecir la variable objetivo y realizar una media de las predicciones de las muestras. Con ello se reduce la varianza y, no siempre, el sesgo [24,25].

No se ajusta un solo árbol sino varios en paralelo formando un “bosque” donde cada árbol aporta su predicción, y finalmente se toma la media de las variables continuas o la clase más frecuente de las variables categóricas. El parámetro que hay que controlar en esta técnica son el número de muestras, es decir, el número de árboles incluidos. Si se aumenta el número de árboles se producirá menos sobreajuste, y el modelo será más preciso [24,25].

Resumiendo, los pasos de este proceso son:

Dados los datos de tamaño N,

1) Repetir m veces i) y ii):

(i) Seleccionar N observaciones con reemplazamiento de los datos originales

(ii) Aplicar un árbol y obtener predicciones para todas las observaciones originales N

2) Promediar las m predicciones obtenidas en el apartado 1)

Este tipo de modelo se suele comportar bien cuando hay relaciones débiles entre las variables independientes, relaciones no lineales y muchas variables categóricas [18].

Los parámetros utilizados para Bagging van a ser los mismos que ara Random Forest pero con distinto valor en el número de variables utilizadas por lo que se van a detallar en apartado de Random Forest.

3.3.2. Random Forest

Random Forest Es una modificación del Bagging que consiste en incorporar aleatoriedad en las variables utilizadas para segmentar cada nodo del árbol, para ello, combina varios sets de variables y de observaciones para poder definir un modelo que no se sobreajuste a los datos proporcionados. Para construir los modelos de Random Forest, haremos las mismas combinaciones que en los modelos de Bagging pero añadiremos la variable que recoge el número de variables input que va a utilizar.

Resumiendo, los pasos de este proceso son:

Dados los datos de tamaño N,

1) Repetir m veces i), ii), iii):

(i) Seleccionar N observaciones con reemplazamiento de los datos originales

(ii) Aplicar un árbol de la siguiente manera:

En cada nodo, seleccionar p variables de las k originales y de las p

elegidas, escoger la mejor variable para la partición del nodo.

(iii)Obtener predicciones para todas las observaciones originales N

2) Promediar las m predicciones obtenidas en el apartado 1)

Se podría decir que este modelo trata de evitar el sobreajuste producido por la selección de variables, además de las ventajas que tenía el Bagging. En este caso se remuestrea las observaciones y las variables para poder conseguir un modelo más preciso y pudiendo detectar relaciones no lineales y complejas entre variables [18].

Los parámetros que ajustaremos más adelante para entrenar los modelos, tanto Bagging como Random Forest, serán:

- *ntrees*: número de árboles que se van a construir para promediarlos.
- *node size*: número de observaciones mínimas por hoja.
- *mtry:* número de variables utilizadas para segmentar cada nodo del árbol. En el caso de Bagging se tienen en cuenta todas las variables input de árbol, y en Random Forest se prueban varios números de variables.

3.3.3. Gradient Boosting

Este algoritmo se basa en un método iterativo de construcción de árboles en el que las predicciones y los residuos se actualizan con el decrecimiento dado por el negativo del gradiente. El objetivo es ir adaptando los árboles para minimizar los residuos encontrando algún patrón en ellos.

Se trata de minimizar la función de coste de los residuos, se empieza con la mejor aproximación de la variable respuesta con un modelo simple y se calculan los residuos, y con ellos se va hacer un nuevo modelo que intente minimizar la función de coste. Los modelos siguientes, se calculan en función a los datos más complicados para que el modelo se pueda ajustar. Este proceso es iterativo y se repite m veces creando modelos que minimicen el error de los anteriores. Finalmente, se combinan todas las variables independientes dando algunos pesos a cada una.

A lo largo de este proceso, hay que tener cuidado con el sobreajuste de los datos. A pesar de esto, es una buena técnica para el buen tratamiento de valores ausentes, es fácil de implementar ya que hay que monitorizar pocos parámetros. Cuando los datos y sus relaciones son complejas, Gradient Boosting es mejor que Random Forest, mientras que si los datos son sencillos es al contrario [18,21].

Los modelos que construiremos en el Capítulo 5, serán en función de los valores que asignaremos a los siguientes parámetros:

- *ntrees*: número de árboles que se van a construir para promediarlos.
- *n.minobsinnode*: número de observaciones mínimas por hoja.
- *shrinkage:* parámetro de regularización para corregir posibles estimaciones erróneas, es la tasa de aprendizaje de Gradient Boosting.

3.3.3. Xgboost

Es una variante del modelo Gradient Boosting con una corrección del algoritmo que consiste en la utilización de la regularización para evitar el sobreajuste. En este caso, la regularización en Xgboost intervierne en el proceso de optimización interna del algoritmo durante la estimación de los parámetros. Se hizo popular en la plataforma de competiciones Kaggle en 2016 por su alta precisión y velocidad de entrenamiento en la competición “High Boson Challenge” por su buena posición en el ranking de la competición.

Cuando se construyen los árboles con una función donde se penaliza el número mínimo de hojas y el parámetro estimado de cada hoja, se añadieron dos parámetros de penalización γ, λ. Con esto se logró controlar el sobreajuste para árboles complejos.

$$\Omega(f_t) = \underbrace{\gamma T}_{\text{Number of leaves}} + \underbrace{\tfrac{1}{2}\lambda \textstyle\sum_{j=1}^{T} w_j^2}_{\text{L2 norm of leaf scores}} \qquad (3.2)$$

Con este modelo se pueden utilizar distintas funciones objetivo, pero hay riesgo de que los parámetros de regularización dependan en gran parte de los datos. Añadir que no siempre es mejor que Gradient Bossting [18].

Los parámetros que se van a monitorizar modelos que construiremos en el Capítulo 5, serán en función de los valores que asignaremos a los siguientes parámetros:

- *ntrees*: número de árboles que se van a construir para promediarlos.
- *n.minobsinnode*: número de observaciones mínimas por hoja.
- *eta:* tasa de aprendizaje de Extreme Greadient Boosting.
- *max.depth:* profundidad del árbol o número de nodos de bifurcación de los árboles de de decisión usados en el entrenamiento.
- *min_child_weight:* número de observaciones mínimas por hoja final.
- *nround:* número de iteraciones para realizar el proceso de ajuste. Cuantas más iteraciones sean se obtendrán mejores resultados, pero tardará más la ejecución.

3.3.4. Support Vector Machines

Aunque el problema que hemos planteado en este trabajo es de regresión, primero vamos a entender cómo funciona los algoritmos Support Vector Machines que tratan problemas de clasificación:

Este tipo de algoritmos plantean el problema de separación lineal de clases con métodos algebraicos para poder buscar el hiperplano óptimo de separación. Se basa en la definición del "maximal margin" que es una función lineal con la que se divide de manera equitativa y diferencia las distintas clases para mejorar tanto el sesgo como la varianza de los resultados. En esta técnica, el conjunto de datos de entrada y salida se trata como vectores [18]

Para encontrar dicho hiperplano (función lineal), SVM intenta encontrar vectores de soporte. Estos se eligen de tal manera que el hiperplano óptimo esté a una distancia máxima posible de ambos vectores de apoyo (maximiza el margen), de forma que ambas clases queden bien diferenciadas.

Una vez entendido el modelo SVM, el modelo SVR será el que vamos a aplicar, puesto que nuestro problema planteado es un problema de regresión y no de clasificación.

Se utilizan los mismos principios que SVM pero con algunos cambios: el objetivo es encontrar una curva que minimice la desviación de los puntos hacia ella. También se utiliza un margen margen de tolerancia (ε) cerca del vector con el fin de minimizar el error teniendo en cuenta que parte de ese error es tolerado. Los datos que estén dentro del margen de decisión serán los datos que van a considerar. Esto nos da un modelo de mejor ajuste. La función de optimización sería minimizar esto [28]:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) \qquad (3.4)$$

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

También es importante tener en cuenta el "soft margin" puesto que es necesario permitir observaciones mal clasificadas para evitar el sobreajuste porque en la práctica no se obtienen divisiones perfectas. Por ello, hay que añadir dos variable ξ_i y ξ_i^* de holgura y la constante C de regularización del margen, que está relacionada inversamente con la anchura del margen y el error para la construcción del hiperplano. A mayor C (menores "residuos" ξ_i y ξ_i^*), menor margen. A menor C (permitimos mayores residuos ξ, ξ_i^*i), más permiso para fallar y más margen.

Se trata de que el vector de parámetros *w* maximice el margen. Podemos ver la explicación gráfica de esto en la Figura 3.5.

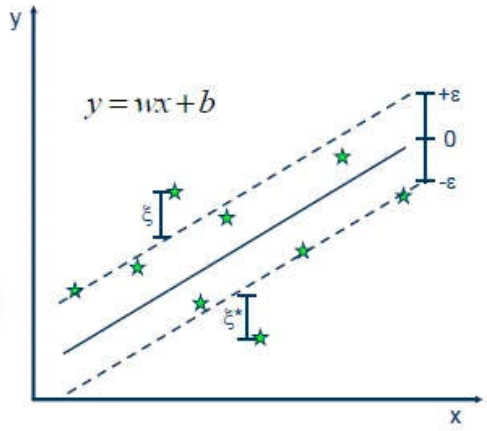

Figura 3.5: Demostración de la variable de holgura del margen que se utiliza en SVR para regresión [29].

Por último, si el problema no es lineal se trabajará en un espacio de dimensión superior donde sí tenga sentido la separación lineal (*kernel*):

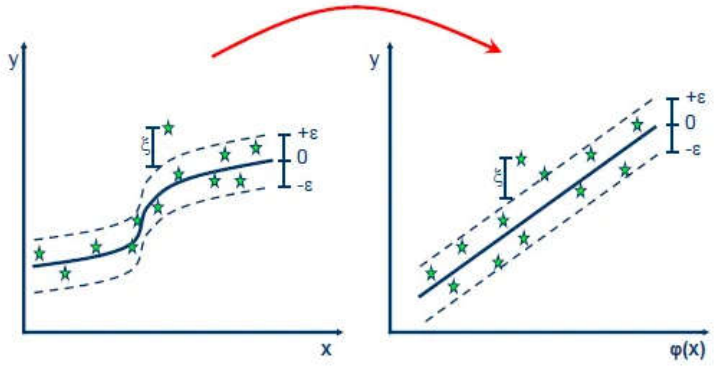

Figura 3.6: Separación entre clases cuando el problema no es lineal y se utiliza una dimensión superior [18].

Cuando se aumenta la dimensión, los cálculos se hacen muy complejos por lo que se utiliza el "truco *Kernel*". En el que el algoritmo al depender de los productos escalares permite trabajar en una dimensión controlada con la función *kernel* que cumple:

$$K(x,y)=<\varphi(x), \varphi(y)> \quad (3.5)$$

Existen varios tipos de *Kernel*, los que vamos a utilizar en este trabajo son lineal y RBF.

Linear function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j$
RBF function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ $= \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma = \frac{1}{2\sigma^2}$

Figura 3.7: Tipos de *Kernel* [18].

Como conclusión, tanto SVM y SVR son algoritmos muy flexibles y potentes, pero lentos a veces para el proceso de optimización [18].

Para probar los modelos SVM lineal y radial, vamos a variar los siguientes parámetros:

- *C*: constante de regularización del margen. Este parámetro se prueba tanto con el *kernel* lineal y radial.
- *sigma*: es un parámetro que tiene que estar en consonancia con el coste para crear la rejilla para la construcción del *kernel* radial.

3.4. Modelos Ensemble

Este método se basa en la construcción de predicciones a partir de la combinación de varios modelos. Nos vamos a centrar la técnica que vamos a utilizar en este trabajo, de combinado de modelos: Stacking.

A la hora de ensamblar, cuanto menor sea la correlación entre clasificadores, más se reducirá el error con el ensamblado.

Esta técnica se utiliza para cualquier combinación de modelos. Para combinarlos, hay tres opciones:

1. **Averaging (promediado):** se calcula el promedio de las predicciones de los modelos que se van a combinar. El promedio se puede ponderar. En este trabajo utilizaremos esta opción.

2. **Voto (para clasificación):** se predice el resultado con mayoría entre las predicciones. En este caso nuestro problema a resolver es de regresión, por lo que esta opción no la contemplamos.

3. **Combinación a partir de otro algoritmo:** 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 la varianza del error en general, casi nunca empeoran los modelos. Como inconvenientes es que son modelos complejos y pueden sobreajustarse con facilidad a los datos [18].

3.5. Técnicas de evaluación de modelos

En este apartado se van a recapitular varias técnicas que nos permitirán evaluar los modelos que entrenemos en el Capítulo 5.

Validación cruzada

Para poder comparar y medir los modelos, necesitamos evaluar su capacidad predictiva y el rendimiento de la generalización del modelo. Esta técnica trata de evitar los posibles problemas relacionados con el sobreajuste y la sobregeneralización del modelo.

Permite entrenar los modelos de Machine Learning en un conjunto diferente del que se evalúan (datos de train y de test). Para ello, en lugar de dividir una sola vez los datos en conjunto de entrenamiento y de pruebas; este proceso se realiza varias veces, entrenando varios modelos [18].

El problema que tiene este método es que implica una gran carga computacional al repetir varias veces el mismo proceso, pero se obtiene un modelo más estable.

El tipo de validación cruzada que vamos a utilizar en este trabajo es la de K grupos. Los pasos son los siguientes:

1) Se dividen los datos aleatoriamente en k grupos

2) Se realiza iterativamente la siguiente operación:

Desde i=1 hasta k

Dejar aparte el grupo i

Construir el modelo con los grupos restantes

Estimar el error (por ejemplo ASE) al predecir el grupo i: Error i

Fin

3) Una medida de error de predicción del modelo será la suma o media de los Error i. [18]

En la Figura 3.8 podemos ver gráficamente los pasos explicados:

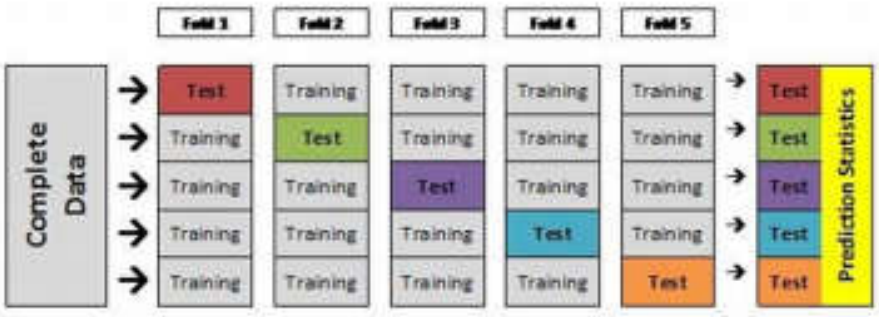

Figura 3.8: Pasos de la validación cruzada (k grupos) [18].

MSE y R^2

Otras medidas que se van a calcular para evaluar los modelos son:

- MSE: mide el error cuadrático medio de las predicciones realizadas por el modelo que se está evaluando. Para cada predicción, calcula la diferencia entre las predicciones y el valor real de la variable objetivo, y luego hace un promedio.

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

Error cuadrático medio (MSE)

(3.5)

En esta fórmula, y_i es el resultado real esperado y $\hat{y}_i$ es la predicción del modelo [16].

- R^2:. es la proporción (porcentaje) de la varianza total de la variable explicada por el modelo que explica su relación con una o más variables independientes. Refleja la bondad del ajuste de un modelo a la variable que pretender explicar. Por lo general, mientras mayor sea el R^2, mejor será el ajuste del modelo.

$$R^2 \approx 1 - \frac{ASE}{\sigma^2}$$

(3.6)

Siendo ASE el error cuadrático medio de las variables objetivo, y el cuadrado *sigma* se corresponde con la varianza de la variable objetivo [16].

CAPÍTULO 4

Descripción y origen de los datos

A lo largo de este capítulo se va a explicar la obtención y construcción (en Python) de la base de datos que se va a utilizar a lo largo de este trabajo con la API de *Spotify*. También se realizará una descripción de las variables y, finalmente, se realizará el análisis exploratorio de la base de datos resultante, junto con la depuración de los datos para, en el posterior capítulo poder aplicar los distintos modelos de Machine Learning.

4.1. Obtención de la base de datos

Como se ha comentado anteriormente, para construir la base de datos a modelizar, se ha utilizado la API de *Spotify* creada para desarrolladores. Para ello, utilizaremos la biblioteca Spotipy de Python para la descarga de los datos musicales, mediante funciones para autenticación y el acceso a los End points , devolviendo los metadatos en formato JSON [30,31].

A continuación, detallaremos los problemas e inconvenientes con la API que han aparecido a lo largo de la programación para la obtención de los datos, puesto que influyeron en el planteamiento de este estudio. Posteriormente planteamos el objetivo final, junto con el procedimiento de la descarga y adecuación de los datos, explicando las funciones de la API necesarias para ello.

4.1.1. Problemas e inconvenientes con la API

A la hora de plantear el problema a resolver y la posterior descarga de los datos, han surgido una serie de problemas que es destacable detallar puesto que han influido en la dirección que se ha tomado finalmente para este trabajo.

Variable objetivo

En un principio, la variable que se quería predecir era el número de reproducciones que podía tener una canción en el momento que se descargan los datos, ya que esto varía constantemente en el tiempo. Revisando la documentación se puede ver que no es posible la descarga de esta variable con la API de *Spotify* [31].

Por ello, se encontró una variable relacionada con el número de reproducciones: la popularidad de la canción. La documentación explica:

"La popularidad de una pista es un valor entre 0 y 100, siendo 100 el más popular. La popularidad se calcula mediante un algoritmo y se basa, en su mayor parte, en el

número total de reproducciones que ha tenido la pista y qué tan recientes son esas reproducciones.

En términos generales, las canciones que se reproducen mucho ahora tendrán una mayor popularidad que las canciones que se tocaron mucho en el pasado. Las pistas duplicadas (por ejemplo, la misma pista de un sencillo y un álbum) se clasifican de forma independiente. La popularidad del artista y del álbum se deriva matemáticamente de la popularidad de la pista. Tenga en cuenta que el valor de popularidad puede retrasarse unos días en la popularidad real: el valor no se actualiza en tiempo real." [32]

Spotify no proporciona mucha información sobre los algoritmos que utilizan para calcular esta variable en su documentación, pero sí nos queda una idea clara de que la popularidad está basada en el número de reproducciones. También hay que tener en cuenta que hay un posible desfase de su valor, al no calcularse en tiempo real.

Descarga de datos de años anteriores

En un principio, el criterio que se planteó para la descarga de los datos de un número suficiente de canciones para este estudio; fue que para los 56 países en los que estaba disponible *Spotify*, se descargarían las listas de reproducción destacadas (junto con sus canciones) desde años anteriores en los que ya había disponibles listas de reproducción.

El problema con el que nos encontramos, es que a la hora de llamar a la función *featured_playlist* de la API, y pasándole el argumento de la fecha en formato ISO 8601 ISO 8601: aaaa-MM-ddTHH: mm: ss., la función no reportaba ningún error, pero no nos devolvía las canciones que podrían estar en las playlist más importantes de años pasados. Lo que la función retornaba era la información solicitada para el día de la semana en concreto que se le pasaba, pero de la semana, es decir, si se intentaba descartar los datos de las listas de reproducción, destacadas de España para el 2 de Julio del año 2015 (jueves), la API nos proporciona los datos del jueves de esta semana [30].

Ante este posible error de la API, se decidió hacer un cambio de criterio y probar hacer la descarga con fecha por defecto el día que se realizó la descarga para todos los países. Finalmente obtuvimos unas 5000 canciones, pero existían canciones duplicadas, por lo que nos quedábamos con 3000 canciones, lo cual nos pareció una base de datos un tanto escasa.

Observando la documentación, se observó que existía otra función que calculaba las canciones mejores de un artista, por lo que se cogieron los 1515 artistas que aparecían en las canciones ya descargadas y se llamó a esta función para extraer sus mejores canciones. Finalmente unimos ambas partes de la descarga en una sola base de datos de 14092 canciones sin duplicar.

Límite de tasa de descarga y parámetro *timeout*

La API web comparte el ancho de banda y los recursos de forma equitativa entre todos los usuarios. Si un usuario se conecta a la aplicación con sus credenciales se le aplica

el límite de tasa, independientemente de si varios usuarios se conectan con las mismas credenciales. Si al hacer una petición de descarga, se obtiene el código de error 429, significa que se ha limitado la tasa al realizar muchas peticiones. Para saber el tiempo que queda para poder volver a realizar peticiones se puede ver en el parámetro Retry-After (en segundos) [30].

Por ello, es un factor importante puesto que nos limita la descarga, y ha dilatado los tiempos de programación del código de descarga. Hay algunas funciones de *spotipy* que puedes pasarle varios objetos para que devuelva la función varios a la vez, el problema es que, para extraer unas características de las canciones, hay que utilizar una función que sólo permita pasarle un objeto nada más, por lo que se ha tenido que extraer toda la información de una en una.

También se ha tenido que modificar el parámetro *timeout* de la aplicación, puesto que por defecto son 5 segundos, y cuando salta este error la descarga termina, por ello, se ha tenido que aumentar para que se pudiesen realizar todas las peticiones.

4.1.2. Planteamiento final del problema

Como ya se ha comentado con los dos primeros inconvenientes que se han tenido con la API de *Spotify*, vamos a tratar de predecir la variable de la popularidad de la canción. La base de datos que hemos construido para este estudio consta de dos partes: en las canciones que se encuentran en las listas de reproducción más importantes de los países a los que da servicio *Spotify*, y en las 10 mejores canciones (como máximo) de cada artista que aparece.

4.1.3. Procedimiento de la descarga y adecuación de los datos

En este apartado se van a explicar las funciones de los distintos módulos de la biblioteca Spotipy utilizadas para que el código adjunto en el Anexo V sea más comprensible. Estas funciones se han utilizado en las dos partes de construcción de la base de datos [30,31].

- *SpotifyClientCredentials:* esta función pertenece al módulo *oauth2* que se encarga de la autenticación del usuario en la API. Esta función crea un objeto con las credenciales necesarias para conectarse. Para obtener dichas credenciales, hay que darse de alta en la web de *Spotify* para desarrolladores. A esta función se le puede pasar varios parámetros, en nuestro caso, hemos utilizado los siguientes:
    - *client id:* id del usuario al darse de alta.
    - *client_secret:* contraseña.
    - *requests_timeout*: como ya se comentó anteriormente, es el número de segundos que esperará una solicitud a poder descargar los datos. Por defecto es 5, y se ha aumentado a 100.

- *Spotify:* pertenece al módulo *client* donde se encarga de crear la sesión del usuario en la API, también pertenecen a este módulo las funciones de las que

se extraen datos de *Spotify*, ya que dependen del objeto que se crea con esta función. Se crea un objeto con la sesión, del que dependerán el resto de funciones para descagar las canciones. Los parámetros utilizados han sido:

  - *client_credentials_manager*: objeto creado con la anterior función *SpotifyClientCredentials*

- *featured_playlists:* con esta función se obtienen las listas más destacadas de cada país, con ello conseguiremos los IDs de las playlist. Los parámetros finalmente utilizados, descartando el de la fecha como ya comentábamos anteriormente, ha sido:
  - *country*: país del que se quiere obtener las listas. El país debe estar en formato de ISO 3166-1 alpha-2. En el Anexo I se muestra una tabla con todos los países donde se encuentra *Spotify* y su nomenclatura de acuerdo a esta norma.

- *playlist_tracks:* obtenemos las canciones que forman parte de la playlist que se solicita a partir de algunos parámetros como su ID, el mercado donde se encuentra, en este caso hemos utilizado:
  - *ID* de la playlist obtenido anteriormente de la lista de *featured_playlists*.
- *Track:* devuelve la información de una canción. También existe la función *Tracks* que devuelve de varias canciones que habría sido muy útil para hacer que la descarga de datos hubiese sido menos tediosa, pero no todas las funciones de esta API permitían pasar una lista de canciones para extraer sus características. Por ello su parámetro utilizado:
  - *ID* de la canción.
- *audio_analysis:* esta función proporciona la información de bajo nivel sobre la estructura y el contenido de la canción con características como el ritmo, el tono y timbre. Para ello la canción se fragmenta en varios segmentos para extraer las características. Estas secciones se definen por grandes variaciones de ritmo o timbre. Añadir que se ha creado una función *ratio* para calcular la media, el valor mínimo y máximo que adoptan en los segmentos de la canción, las variables: *loudness, tempo, key, mode* y *time_signature*. Necesitamos:
  - *ID* de la canción (esta función era a la que no se puede pasar una lista de identificadores).
- *audio_features:* obtenemos característicasde las canciones a nivel general como su duración, y algunos indicadores propios de *Spotify* como lo bailable que puede ser una canción. Necesitamos:
  - *ID* de la canción
- *album:* obtenemos información sobre el álbum en el que se lanzó la canción. Utilizamos:
  - *ID* del álbum.
- *artist:* obtenemos información sobre el artista de la canción como el número de seguidores que tiene. El parámetro necesario:
  - *ID* del artista.
- *artist_top_tracks:* se obtienen las 10 mejores canciones (como máximo) de un artista. Necesitamos:
  - *ID* del artista.

4.2. Descripción de las variables

Con todo el proceso anterior, obtuvimos 117 variables y 14094 observaciones, pero en esas variables había información duplicada porque algunas funciones dan variables repetidas y porque algunas variables no íbamos a utilizarlas, como el nombre del artista, de la canción y álbum, enlaces a la ubicación de la información en *Spotify*, etc. Se ha hecho esta preselección de variables para eliminar las que a priori no aportan información para la construcción de los modelos.

Las variables utilizadas, distinguiendo las de intervalo de las categóricas, son las siguientes:

- Variables de intervalo:
  - *popularity_track:* como ya se ha comentado anteriormente, la popularidad toma un valor entre 0 y 100, siendo 100 el más popular. Se calcula, en su mayor parte, a partir del número de reproducciones, y cómo de recientes son. A partir de esta variable, se calcula la popularidad del artista de la canción, y del álbum, por ello, eliminaremos estas variables ya que se obtienen a partir de la popularidad de la variable objetivo.
  - *acousticness_track:* esta variable adopta valores entre 0 y 1. Es una medida de confianza sobre si la canción es acústica. Por ello, cuanto más se acerque a 1 su valor, es que hay una confianza alta de que la canción es acústica.
  - *danceability_track:* se describe como de "bailable" es la canción. Adopta valores entre el 0 y 1, siendo 1 el valor más "bailable" de la canción. En la documentación se describe que esta variable es en función del tempo, estabilidad y fuerza del ritmo, además de la regularidad general de la canción.
  - *duration_ms_track:* duración de la canción en milisegundos.
  - *energy_track*: recoge la percepción sobre la intensidad y actividad de la canción. Puede valer entre 0 y 1, por lo que las pistas más enérgicas, rápidas y ruidosas son las que más puntuación tendrán en esta medida. Se tiene en cuenta el rango de la canción, volumen, timbre, frecuencia de inicio y entropía.
  - *instrumentalness_track*: estima si una canción tiene voces en su composición. Entre 0 y 1 se encuentra esta medida, por lo que una pista muy hablada como puede ser el rap, se acercará más a 0 y si no contiene apenas voces, estará más cerca de 1 por lo que será más "instrumental" la canción. Los sonidos "Ooh" y "aah" que puedan aparecer en alguna canción, no se consideran voces.
  - *key_track*: es el tono estimado por *Spotify*
  - *key_max_track*, *key_mean_track* y *key_min_track: c*alculamos con la función *ratio* el valor máximo, mínimo y la media de key a partir de los valores que hay en todas las secciones en las que se divide la canción en la función *audio_analysis.*

- *liveness_track:* del 0 a 1 se recoge si la canción se ha interpretado en vivo.
- *loudness_track: es el volumen de la canción en dB. R*epresenta la calidad del sonido, que está relacionado con la amplitud. Los valores típicos oscilan entre -60 y 0 dB. *Spotify* lo calcula promediando los valores de sonoridad de la canción.
- *loudness_max_track, loudness_mean_track* y *loudness_min_track: c*alculamos con la función *ratio* el valor máximo, mínimo y la media de loudness a partir de los valores que hay en todas las secciones en las que se divide la canción en la función *audio_analysis.*
- *mode_track:* se indica la modalidad de la canción estimada por *Spotify*, en función de su contenido melódico del 0 al 1. Éste será mayor cuando se acerque a 1 y menor en caso contrario.
- *mode_max_track, mode_mean_track* y *mode_min_track:* calculamos con la función *ratio* el valor máximo, mínimo y la media de loudness a partir de los valores que hay en todas las secciones en las que se divide la canción en la función *audio_analysis.*
- *speechiness_track:* se representa entre 0 y 1 si hay presencia de palabras habladas. Cuanto más se acerque a 1, la canción probablemente será hablada y no tenga música, si está entre 0,33 y 0,66 las pistas constan tanto de música como de voz, y menor a 0,33 será música solamente sin habla.
- *tempo_track:* es la velocidad o el ritmo de la canción, derivada de la duración de ésta. *Spotify* estima este valor en pulsaciones por minuto (BPM).
- *tempo_max_track, tempo_mean_track* y *tempo_min_track:* calculamos con la función *ratio* el valor máximo, mínimo y la media de tempo a partir de los valores que hay en todas las secciones en las que se divide la canción en la función *audio_analysis.*
- *time_signature_track:* es el valor estimado de la signatura de tiempo de la canción. Marca la medida del compás, que está compuesto por varias medidas de tiempo.
- *time_signature_max_track, time_signature_mean_track* y *time_signature_min_track:* calculamos con la función *ratio* el valor máximo, mínimo y la media de time signature a partir de los valores que hay en todas las secciones en las que se divide la canción en la función *audio_analysis.*
- *total_followers_artist*: número de seguidores que tiene el artista de la canción.
- *total_tracks_album:* número total de canciones que tiene el álbum disponible.
- *track_number_track*: recoge el número de la canción en el álbum, si este tuviese varios discos, reflejaría el número de discos donde se encuentra la canción.
- *valence_track:* describe la positivad o negativad que transmite una canción. Adoptará valores más cercanos a 1 las canciones más alegres que transmiten sentimientos positivos, y más cercanas a 0 las más tristes (con emociones negativas).

- *Variables categóricas:*
  - *id_track:* es el identificador único de la canción. Posteriormente excluiremos esta variable.
  - *album_type:* recoge el tipo de álbum que puede ser “single”, “álbum”, “compilation”.
  - *label_album:* productora del álbum.
  - *disc_number_track:* es el número de disco en el que se encuentra la canción, suele tener el valor 1 a menos que el álbum conste de más de un disco. Los valores que se toman en esta base de datos es de 1 a 6, por ello tratamos esta variable como categórica.
  - *explicit_track:* recoge si la canción tiene letras explícitas o no (puede adoptar los valores “True”, “False”, “Unknown”. Una canción se cataloga que es explícita cuando su contenido hace referencia a violencia, discriminación o sexo.
  - *genres_artist:* tupla de géneros a los que se asocia el artista. Si dicha tupla está vacía, es que el artista aún no ha sido clasificado.

4.3. Depuración y análisis exploratorio

Primer análisis

En este apartado utilizaremos la herramienta SAS Miner ya que es muy intuitiva y muy útil para visualizar y tratar los datos para su adecuación de cara a construir los modelos de predicción. Explicaremos los nodos utilizados en esta herramienta, y todos los pasos realizados para dejar la base de datos preparada de cara a la selección de variables.

Una vez importados los datos y asignar los roles de las variables utilizadas, con el nodo de Explorador de Estadísticos, obtenemos el resultado de la Figura 4.1. Visualizando esta información, podemos ver varios puntos:

- Presencia de valores Missing en las variables: en las variables de intervalo no tenemos ningún valor ausente, esto tiene sentido porque a lo largo de la programación del código se ha intentado evitar que los resultados que devuelven las funciones de *Spotify* estuvieran vacíos. Por lo que tiene lógica, pero, aun así, al controlarse que el objeto (que contiene todas las variables) no fuese vacío en sí; aunque puede darse el caso de que este no esté vacío pero alguna variable sí podría estarlo. Por ello, como podemos ver en las variables categóricas genres_artist existe el valor “[]”, y este valor es ausente, por ello, y porque este valor es la segunda categoría más repetida de la variable, procederemos a eliminarla.

- Presencia de valores erróneos en las variables: en este caso no podemos saber si existen valores atípicos en la mayoría de las variables puesto que no son cantidades acotadas. Podemos ver como las variables que recogen las características de las canciones están bien codificadas (adoptan sólo valores 0 ó 1) y en el resto de variables podemos ver que no hay errores al no existir cantidades negativas, menos en la variable

que recoge el volumen (loudness) ya que esta medida puede ser negativa debido a atenuaciones.

Estadísticos de sumarización de la variable de clase
(máximo imprimido 500 observaciones)

Rol de los datos=TRAIN

Rol de los datos	Nombre de la variable	Rol	Número de niveles	Ausente	Moda	Porcentaje moda	Moda2	Porcentaje Moda2
TRAIN	album_type	INPUT	3	0	single	52.85	album	45.53
TRAIN	disc_number_track	INPUT	5	0	1	99.33	2	0.54
TRAIN	explicit_track	INPUT	2	0	False	89.15	True	10.85
TRAIN	genres_artist	INPUT	513	0	[]	11.92	['argentine hip hop', 'trap arge	4.62
TRAIN	label_album	INPUT	513	0	Columbia	3.77	Sony Music Latin	3.54

Estadísticos descriptivos de la variable de intervalo
(máximo imprimido 500 observaciones)

Rol de los datos=TRAIN

Variable	Rol	Media	Desviación estándar	No ausente	Ausente	Mínimo	Mediana	Máximo	Asimetría	Curtosis
acousticness_track	INPUT	0.383844	0.31154	14094	0	0.000014	0.314	0.996	0.409884	-1.22896
danceability_track	INPUT	0.627915	0.147539	14094	0	0.0625	0.643	0.987	-0.38358	-0.25643
duration_ms_track	INPUT	224125.1	60737.78	14094	0	28106	214293	1042133	2.13878	11.74561
energy_track	INPUT	0.555278	0.212633	14094	0	0.00684	0.567	0.999	-0.25176	-0.70889
instrumentalness_track	INPUT	0.056288	0.182241	14094	0	0	7.19E-6	0.973	3.622408	12.23082
key_max_track	INPUT	9.128423	2.114336	14094	0	0	10	11	-1.61388	2.930374
key_mean_track	INPUT	4.704555	2.063282	14094	0	0	5	11	0.089066	-0.30056
key_min_track	INPUT	1.32262	1.720728	14094	0	0	1	11	1.996406	4.860455
key_track	INPUT	5.315595	3.586581	14094	0	0	5	11	-0.01192	-1.28238
liveness_track	INPUT	0.163591	0.126879	14094	0	0.0148	0.115	0.988	2.664235	8.812514
loudness_max_track	INPUT	-6.05238	3.06554	14094	0	-31.757	-5.506	2.503	-1.32917	3.423705
loudness_mean_track	INPUT	-10.7326	4.225074	14094	0	-38.4235	-10.1625	-1.12267	-0.86547	1.191716
loudness_min_track	INPUT	-23.4937	11.52493	14094	0	-60	-21.25	-2.714	-0.83343	0.281761
loudness_track	INPUT	-8.23988	3.5542	14094	0	-35.748	-7.608	0.642	-1.19745	2.531229
mode_max_track	INPUT	0.988222	0.10789	14094	0	0	1	1	-9.05168	79.94432
mode_mean_track	INPUT	0.121257	0.326438	14094	0	0	0	1	2.32079	3.386545
mode_min_track	INPUT	0.121257	0.326438	14094	0	0	0	1	2.32079	3.386545
mode_track	INPUT	0.633035	0.481994	14094	0	0	1	1	-0.5521	-1.69543
speechiness_track	INPUT	0.079068	0.082662	14094	0	0.0225	0.0461	0.934	2.975131	11.64985
tempo_max_track	INPUT	121.8919	28.8809	14094	0	37.88	120.604	227.718	0.491869	-0.24256
tempo_mean_track	INPUT	117.7934	28.82888	14094	0	24.47417	116.9275	216.1954	0.453849	-0.21482
tempo_min_track	INPUT	102.3868	48.07097	14094	0	0	110.618	215.744	-0.87194	0.341707
tempo_track	INPUT	119.0591	28.38359	14094	0	36.534	118.159	216.334	0.496036	-0.18623
time_signature_max_track	INPUT	4.08124	0.374636	14094	0	1	4	5	0.754968	4.560882
time_signature_mean_track	INPUT	3.694196	0.507517	14094	0	1	4	5	-1.3587	1.496052
time_signature_min_track	INPUT	3.499007	0.908403	14094	0	1	4	5	-1.89741	2.555584
time_signature_track	INPUT	3.922591	0.364709	14094	0	1	4	5	-3.79355	24.76531
total_followers_artist	INPUT	1679873	5381441	14094	0	3	95334	68550925	6.139204	50.61144
total_tracks_album	INPUT	7.037605	7.894375	14094	0	1	4	244	4.831568	80.85945
track_number_track	INPUT	3.608912	3.939174	14094	0	1	2	111	3.656445	50.81875
valence_track	INPUT	0.485538	0.239439	14094	0	0.027	0.468	0.981	0.196068	-0.94803
popularity_track	TARGET	46.45949	20.47151	14094	0	0	48	100	-0.33379	-0.43056

Figura 4.1: Resultados del Nodo Explorador de Estadíticos de SAS Miner de las variables originales.

- Valores de asimetría y curtosis en las variables: la asimetría mide cuánto está centrada la distribución de una variable, y la curtosis mide la mayor o menor concentración de datos alrededor de la media. Según el Teorema de la débil normalidad la asimetría y la curtosis para que la variable sea normal, tienen que encontrar en el intervalo comprendido entre -2 y 2. No lo cumplen las variables *duration_ms_track, instrumentalness_track,mode_max_track,speechiness_track,me_signature_max_track, time_signature_track, total_followers_Artist, total_Tracks_album* y *track_number_track*.
.

- Podemos ver que hay variables con categorías muy poco representadas: sus categorías apenas representan el 1% de la información de los datos como en *disc_number_track.* Esta variable tiene 5 niveles, pero están 4 infrarrepresentados, por ello, vamos a proceder a agrupar esos 4 niveles en uno. Se podría eliminar la variable directamente, pero como vamos a hacer selección de variables, dejaremos que sean los

modelos de selección los que elijan o descarten la variable. Además, para la variable label_album, procedemos a eliminarla debido a que tiene 513 niveles, y no está claro que se pudiese agrupar las categorías debido a que se trata de los nombres de productoras del álbum.

También se ha obtenido para las variables de intervalo Figura 4.3 con el coeficiente de correlación de Pearson.

- Podemos ver que las variables de intervalo no están altamente correlacionadas con la variable objetivo. No obstante, la que muestra una mayor correlación positiva es *total_followers_artist* la que muestra una mayor correlación negativa es *instrumentalness_track*. Podemos verlo en el siguiente gráfico:

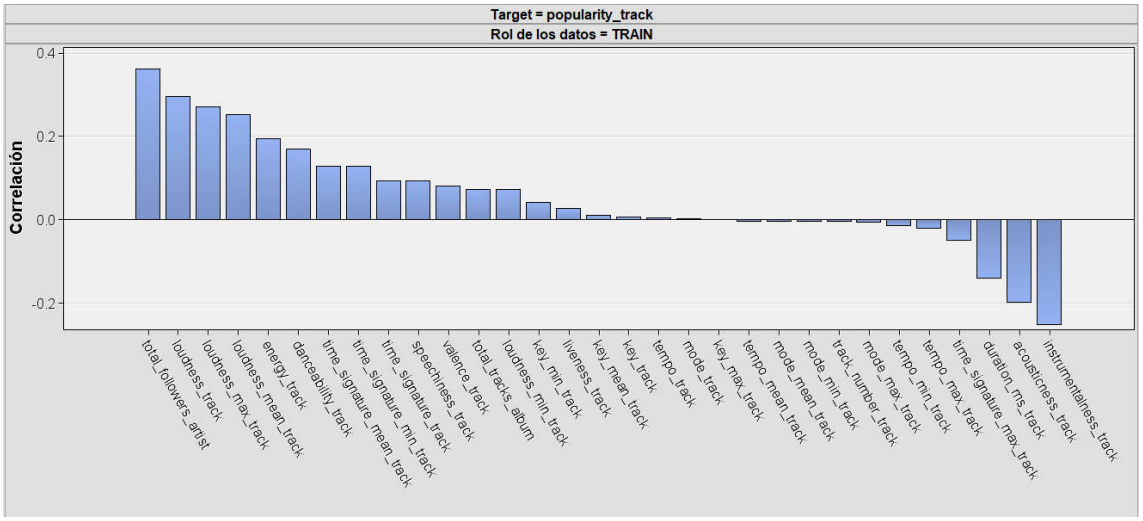

Figura 4.2: Gráfico de Correlación de Pearson de las variables de intervalo con la variable objetivo.

Obtenemos el histograma con la distribución que sigue la variable objetivo:

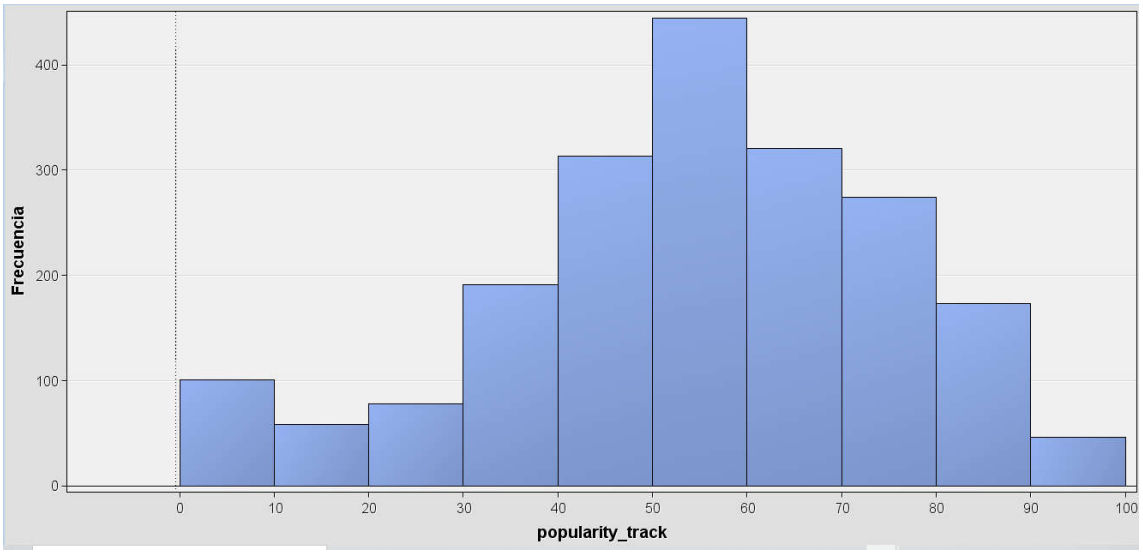

Figura 4.3: Histograma de la variable objetivo.

La variable *popularity_track* sigue aproximadamente una distribución normal, donde sus valores más frecuentes son entre 50 y 60 de popularidad.

Corrección de los errores detectados y reemplazo de valores

Utilizaremos el nodo Reemplazo para reagrupar la variable *disc_number_Track:* como la categoría que más aparece es cuando vale “1”, agruparemos en el grupo “2” cuando esta variable adopta los valores “2,3,4,5,6” y se origina la variables *REP_dis_number_track*.

A continuación, eliminaremos las variables *genres_artist* y *label_album,* para ello utilizaremos el nodo Descartar. Después de estos cambios, las variables categóricas quedarían así:

Estadísticos de sumarización de la variable de clase
(máximo imprimido 500 observaciones)

Rol de los datos=TRAIN

Rol de los datos	Nombre de la variable	Rol	Número de niveles	Ausente	Moda	Porcentaje moda	Moda2	Porcentaje Moda2
TRAIN	REP_disc_number_track	INPUT	2	0	1	99.33	2	0.67
TRAIN	album_type	INPUT	3	0	single	52.85	album	45.53
TRAIN	explicit_track	INPUT	2	0	False	89.15	True	10.85

Figura 4.4: Resultados del Nodo Explorador de Estadísticos de SAS Miner después de recategorizar y eliminar algunas variables categóricas.

Se exporta la base de datos resultante tras estos cambios, puesto que este set de variables originales se probará junto con un nuevo set de variables transformadas que crearemos a continuación.

Transformación de variables

En esta parte del análisis se van a realizar transformaciones en algunas variables para intentar lograr que el modelo prediga mejor la variable objetivo. Las variables de intervalo se transformarán con el criterio “Correlación máxima”, pero las variables categóricas no las vamos a transformar puesto que tienen pocos niveles. Mediante este método se selecciona la transformación que maximiza el coeficiente de correlación lineal con la variable. En la siguiente imagen podemos observar las transformaciones realizadas sobre las variables de intervalo:

Transformaciones calculadas
(máximo imprimido 500 observaciones)

Input Name	Role	Input Level	Name	Level	Formula
danceability_track	INPUT	INTERVAL	PWR_danceability_track	INTERVAL	(max(danceability_track-0.0625, 0.0)/0.9245)**0.25
energy_track	INPUT	INTERVAL	SQRT_energy_track	INTERVAL	sqrt(max(energy_track-0.00684, 0.0)/0.99216)
instrumentalness_track	INPUT	INTERVAL	SQRT_instrumentalness_track	INTERVAL	sqrt(max(instrumentalness_track-0, 0.0)/0.973)
key_mean_track	INPUT	INTERVAL	PWR_key_mean_track	INTERVAL	(max(key_mean_track-0, 0.0)/11)**4
key_min_track	INPUT	INTERVAL	EXP_key_min_track	INTERVAL	exp(max(key_min_track-0, 0.0)/11)
key_track	INPUT	INTERVAL	PWR_key_track	INTERVAL	(max(key_track-0, 0.0)/11)**4
liveness_track	INPUT	INTERVAL	SQRT_liveness_track	INTERVAL	sqrt(max(liveness_track-0.0148, 0.0)/0.9732)
loudness_max_track	INPUT	INTERVAL	PWR_loudness_max_track	INTERVAL	(max(loudness_max_track--31.757, 0.0)/34.26)**4
loudness_mean_track	INPUT	INTERVAL	EXP_loudness_mean_track	INTERVAL	exp(max(loudness_mean_track--38.42352381, 0.0)/37.300857143)
loudness_min_track	INPUT	INTERVAL	PWR_loudness_min_track	INTERVAL	(max(loudness_min_track--60, 0.0)/57.286)**4
loudness_track	INPUT	INTERVAL	PWR_loudness_track	INTERVAL	(max(loudness_track--35.748, 0.0)/36.39)**4
mode_max_track	INPUT	INTERVAL	EXP_mode_max_track	INTERVAL	exp(max(mode_max_track-0, 0.0))
mode_mean_track	INPUT	INTERVAL	SQR_mode_mean_track	INTERVAL	(max(mode_mean_track-0, 0.0))**2
mode_min_track	INPUT	INTERVAL	SQR_mode_min_track	INTERVAL	(max(mode_min_track-0, 0.0))**2
mode_track	INPUT	INTERVAL	EXP_mode_track	INTERVAL	exp(max(mode_track-0, 0.0))
speechiness_track	INPUT	INTERVAL	LOG_speechiness_track	INTERVAL	log(max(speechiness_track-0.0225, 0.0)/0.9115 + 1)
tempo_mean_track	INPUT	INTERVAL	LOG_tempo_mean_track	INTERVAL	log(max(tempo_mean_track-24.474166667, 0.0)/191.72120833 + 1)
tempo_min_track	INPUT	INTERVAL	LOG_tempo_min_track	INTERVAL	log(max(tempo_min_track-0, 0.0)/215.744 + 1)
tempo_track	INPUT	INTERVAL	PWR_tempo_track	INTERVAL	(max(tempo_track-36.534, 0.0)/179.8)**4
time_signature_max_track	INPUT	INTERVAL	PWR_time_signature_max_track	INTERVAL	(max(time_signature_max_track-1, 0.0)/4)**4
time_signature_min_track	INPUT	INTERVAL	PWR_time_signature_min_track	INTERVAL	(max(time_signature_min_track-1, 0.0)/4)**4
total_followers_artist	INPUT	INTERVAL	LOG_total_followers_artist	INTERVAL	log(max(total_followers_artist-3, 0.0)/68550922 + 1)
total_tracks_album	INPUT	INTERVAL	SQRT_total_tracks_album	INTERVAL	sqrt(max(total_tracks_album-1, 0.0)/243)
valence_track	INPUT	INTERVAL	LOG_valence_track	INTERVAL	log(max(valence_track-0.027, 0.0)/0.954 + 1)

Figura 4.5: Resultados del Nodo Explorador de Estadíticos de SAS Miner de las variables originales continuas.

No hemos descartado las variables originales, para que el propio método de selección determine cuáles son preferibles entre las transformadas y éstas. Introducimos un nodo de Explorador de Estadísticos para ver la correlación entre las variables transformadas con la variable objetivo, y compararlo con las originales.

En la Figura 4.10 podemos ver que las correlaciones de Pearson con la variable objetivo no son mucho mayores que los de las variables originales. Aun así vamos a dejar que las selecciones de variables que probaremos, decidan con qué variables nos quedamos. Guardamos esta nueva base de datos (con variables transformadas y originales) para la posterior selección del siguiente capítulo.

Estadísticos de correlación
(máximo imprimido 500 observaciones)

Rol de los datos=TRAIN Tipo=PEARSON Target=popularity_track

Input	Correlación
LOG_total_followers_artist	0.38967
total_followers_artist	0.36145
PWR_loudness_track	0.30510
loudness_track	0.29606
PWR_loudness_max_track	0.28467
loudness_max_track	0.27203
EXP_loudness_mean_track	0.25488
loudness_mean_track	0.25345
SQRT_energy_track	0.19797
energy_track	0.19477
PWR_danceability_track	0.17088
danceability_track	0.16947
PWR_time_signature_min_track	0.13338
time_signature_mean_track	0.12976
time_signature_min_track	0.12780
LOG_speechiness_track	0.10048
PWR_loudness_min_track	0.09788
time_signature_track	0.09448
speechiness_track	0.09413
LOG_valence_track	0.08885
valence_track	0.08156
total_tracks_album	0.07411
loudness_min_track	0.07354
SQRT_total_tracks_album	0.05335
key_min_track	0.04177
EXP_key_min_track	0.03816
SQRT_liveness_track	0.02801
liveness_track	0.02760
PWR_key_mean_track	0.01539
key_mean_track	0.01197

PWR_key_track	0.01125
PWR_tempo_track	0.00968
key_track	0.00701
tempo_track	0.00502
EXP_mode_track	0.00378
mode_track	0.00378
key_max_track	0.00169
tempo_mean_track	-0.00251
SQR_mode_mean_track	-0.00303
SQR_mode_min_track	-0.00303
mode_mean_track	-0.00303
mode_min_track	-0.00303
LOG_tempo_mean_track	-0.00304
track_number_track	-0.00372
mode_max_track	-0.00497
EXP_mode_max_track	-0.00497
tempo_min_track	-0.01413
LOG_tempo_min_track	-0.01825
tempo_max_track	-0.01909
time_signature_max_track	-0.04753
PWR_time_signature_max_track	-0.07623
duration_ms_track	-0.14024
acousticness_track	-0.19801
instrumentalness_track	-0.24958
SQRT_instrumentalness_track	-0.26069

Figura 4.6: Resultados del Nodo Explorador de Estadísticos de SAS Miner de las variables originales y transformadas (correlación de Pearson).

CAPÍTULO 5

Modelización

En la primera parte de este capítulo se van probar varios métodos de selección de variables, detallando qué criterios se han seguido y qué selección será la elegida entre las candidatas para probarla en los distintos modelos de Machine Learning que se detallaron en el Capítulo 3.

En la segunda parte se detallará qué pruebas se han hecho en cada modelo probado, eligiendo los modelos más competitivos para posteriormente realizar el ensamblado de dichos modelos y poder compararlos.

5.1. Selección de variables

Para probar las distintas selecciones de variables teniendo en cuenta los métodos comentados en el Capítulo 3 (apartado 3.1.1), vamos a realizarlo en SAS. El código se encuentra en el Anexo II.

Cuando terminamos la depuración de la base de datos en el capítulo anterior, guardamos dos sets de variables: originales y transformadas (con variables transformadas y originales). Vamos a utilizar las dos bases de datos probando los tres métodos de selección de variables forward, backward y stepwise con cada una. Y, además, por cada criterio de selección, vamos a probar los tres criterios de información AIC, BIC y SBC. Lo que hace un total de 18 selecciones de variables posibles.

Para poder realizar estas pruebas en SAS con regresión lineal, vamos a utilizar la macro *%randomselect* que realiza un proceso repetido con el criterio de selección deseado, sorteando cada vez un 80% de datos train del archivo con 100 semillas distintas. Durante el proceso, se realiza una cuenta de las veces que, para cada criterio de información, la regresión escoge una selección de variables. Por ello se escogerán las selecciones de variables que más veces se hayan contado para cada criterio de información.

A continuación, se ha incluido una tabla donde aparecen todas las selecciones de variables (a veces se ha elegido más de una), en total han sido 22 sets de variables posibles. Finalmente, renumeramos los 12 modelos (sin duplicados) de selecciones de variables que vamos a comparar para elegir la más adecuada para nuestros modelos de predicción

En esta tabla de resumen de la Figura 5.1, aparecen las variables (tanto originales como transformadas) que aparecen en alguna selección de variables. En la columna aparecen las variables elegidas, y se han ido marcando por fila las celdas con un tick si aparecen en la selección de variables correspondiente.

	Selección 1	Selección 2	Selección 3	Selección 4	Selección 5	Selección 6	Selección 7	Selección 8	Selección 9	Selección 10	Selección 11	Selección 12
album_type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
explicit_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
acousticness_track	✓		✓	✓	✓	✓	✓		✓	✓		
danceability_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
duration_ms_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
instrumentalness_track	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
energy_track	✓		✓	✓	✓	✓	✓	✓	✓		✓	
key_min_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
loudness_min_track	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓
loudness_track	✓	✓	✓	✓	✓	✓			✓	✓		✓
loudness_mean_track	✓			✓		✓			✓			
time_signature_min_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mode_track	✓		✓	✓		✓	✓					✓
time_signature_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
total_followers_artist	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mode_mean_track	✓		✓	✓		✓						
total_tracks_album	✓	✓	✓	✓	✓	✓						✓
valence_track	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
speechiness_track	✓		✓	✓		✓	✓		✓	✓		
time_signature_max_track	✓		✓	✓		✓	✓		✓	✓	✓	
tempo_max_track			✓	✓					✓	✓		
tempo_mean_track			✓	✓								
tempo_min_track			✓	✓								
key_max_track							✓		✓	✓		
tempo_track			✓	✓					✓	✓		
track_number_track	✓		✓	✓		✓	✓		✓	✓		
loudness_max_track									✓	✓		
EXP_key_min_track							✓		✓	✓		
LOG_total_followers_artist							✓	✓	✓	✓	✓	
LOG_speechiness_track							✓		✓	✓		
PWR_loudness_track							✓	✓			✓	
EXP_loudness_mean_track									✓			
EXP_mode_track									✓	✓		
PWR_loudness_min_track									✓		✓	
PWR_time_signature_max_track							✓	✓	✓	✓	✓	
LOG_valence_track							✓					
SQRT_instrumentalness_track							✓	✓	✓	✓		
LOG_tempo_mean_track									✓	✓		
SQRT_total_tracks_album							✓	✓	✓	✓	✓	
LOG_tempo_min_track									✓	✓		
PWR_loudness_max_track									✓	✓		
PWR_tempo_track									✓	✓		
SQRT_energy_track										✓		
Nº total de variables de la selección	21	13	24	25	15	21	26	16	34	32	16	14

Figura 5.1: Tabla de las selecciones de variables sin duplicados.

Podemos ver que los sets de variables más sencillos son 2, 8,11 y 12 ya que tienen la mitad aproximadamente de variables con respecto al resto. Es lógico que para las últimas selecciones se escojan más variables ya que se han tenido en cuenta el doble.

5.2. Regresión lineal

A continuación, se va a proceder hacer las regresiones lineales con los sets de variables. Para ello utilizamos la macro *%cruzada* que realiza validación cruzada repetida con el modelo de regresión planteado y tiene como salida el archivo con los errores promedio cometidos en cada ejecución de validación cruzada para cada semilla.

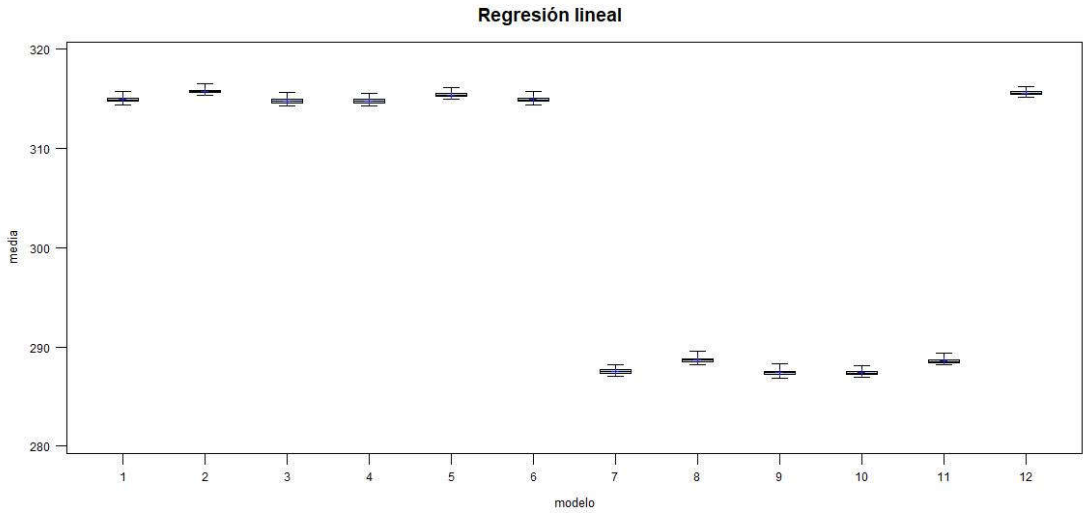

Figura 5.2: Regresión lineal de todas las selecciones de variables de la Tabla 2. El eje de abscisas representa los 12 modelos y el eje de ordenadas representa su MSE.

En el gráfico podemos ver que los modelos del 1 al 6 y también el 12, tienen más error cuadrático medio que los modelos del 7 al 11, lo que se puede concluir, que funcionan mejor los sets de variables con variables originales y transformadas. Nos quedamos con los modelos:

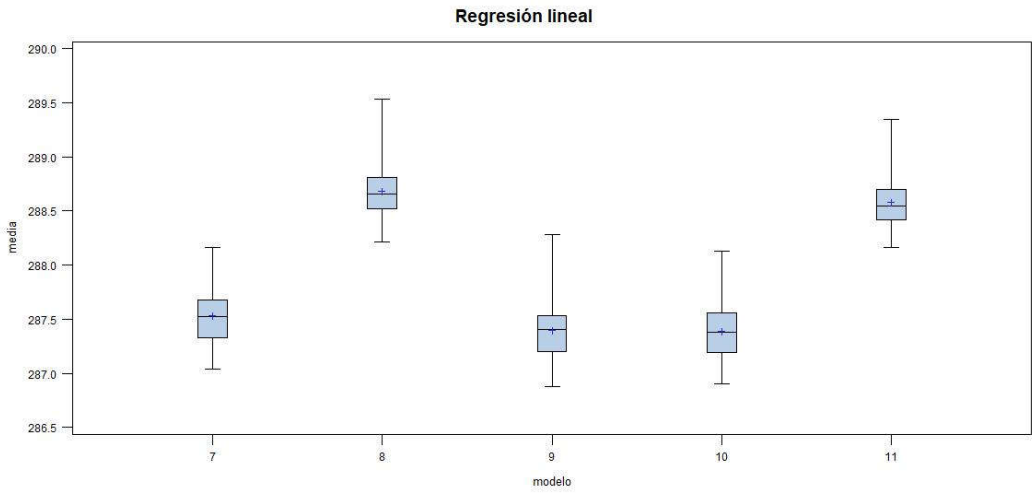

Figura 5.3: Regresión lineal de las selecciones de variables con transformadas. El eje de abscisas representa los 12 modelos y el eje de ordenadas representa su MSE.

Los modelos 8 y 11 son los más sencillos (tienen 16 variables) pero más tienen más MSE que los demás. Los modelos 9 y 10 tienen 34 variables, y el modelo 7 tiene 26. Nos decantamos por el modelo 7 porque tiene un MSE parecido a los modelos 9 y 10 y varianza parecidos, pero es más sencillo. Por lo que elegimos este por el principio de parsimonia.

Calculando su R^2 para ver la bondad de ajuste que tiene el modelo de regresión elegido, obtenemos 0.3138, lo que quiere decir que con este modelo sólo somos capaces de explicar aproximadamente el 31% de la variación en la variable de respuesta. Se tendrá en cuenta este valor de cara a los siguientes modelos que probaremos, para ver si mejora.

Nuestra selección de variables para probar el resto de modelos es:

Selección de variables 7
album_type
explicit_track
EXP_key_min_track
LOG_speechiness_track
LOG_total_followers_artist
LOG_valence_track
PWR_loudness_track
PWR_time_signature_max_track
SQRT_instrumentalness_track
SQRT_total_tracks_album
acousticness_track
danceability_track
duration_ms_track
energy_track
instrumentalness_track
key_max_track
key_min_track
loudness_min_track
mode_track
speechiness_track
time_signature_max_track
time_signature_min_track
time_signature_track
total_followers_artist
track_number_track
valence_track

Figura 5.4: Tabla con las variables seleccionadas.

En los siguientes apartados, los modelos se van a programar en R y no en SAS. El código de R se encuentra en el Anexo III de este trabajo.

Aclarar que todas las funciones que se van a utilizar en R para los modelos son con validación cruzad, para las cuales es necesario estandarizar las variables continuas y crear variables dummies para las categóricas. La nomenclatura de los modelos vendrá dada por los valores de sus variables separados entre "_".

Los modelos escogidos se resaltarán con un recuadro verde.

5.3. Redes Neuronales

Para el tuneado de la red neuronal, vamos a utilizar la macro *cruzadaavnnet* en R, a la que pasaremos distintos valores de los parámetros de la red que explicamos en el Capítulo 3. Para ello se han realizado unos bucles anidados por orden de aparición de cada variable.

Comenzaremos variando el número de nodos (primer bucle), con ello pretendemos encontrar el mejor modelo intentando que no se produzca sobreajuste. Tendremos en cuenta también la complejidad de los datos (aunque esto nos lleve a sobreparametrizar el modelo). Posteriormente se modifica en el segundo bucle, el valor del parámetro weight decay o learning rate. Este hiperparámetro indica el costo del algoritmo de optimización para actualizar los pesos.

Los valores con los que vamos a variar los parámetros mencionados:

- size = 5,10,15,20
- *decay* = 0.001,0.01,0.1

Variando estos parámetros, tenemos 12 combinaciones de modelos que representamos ordenados por su MSE en la Figura 5.5.

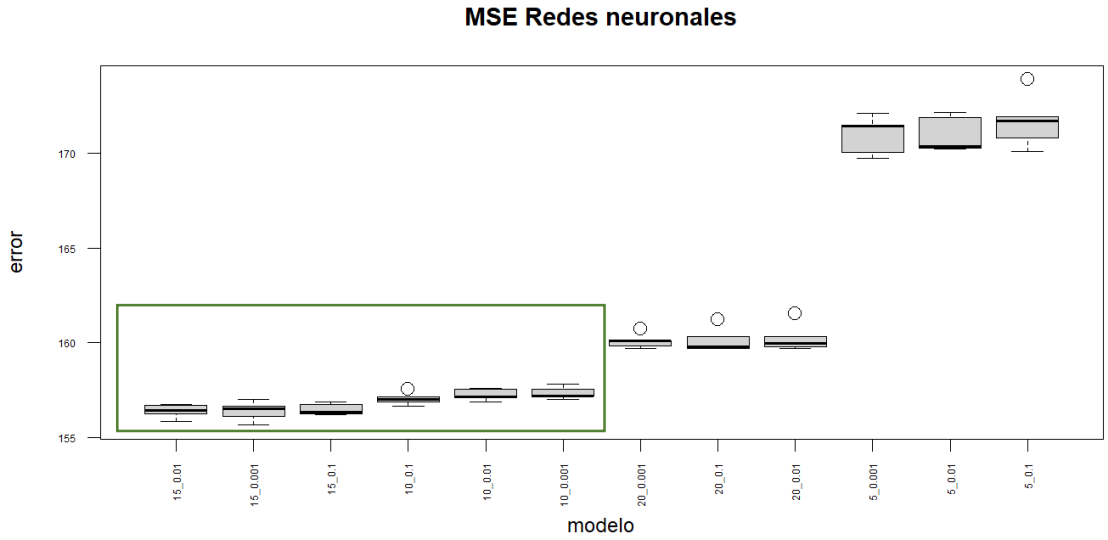

Figura 5.5: Redes neuronales con la selección de variables elegida. El eje de abscisas representa los 12 modelos y el eje de ordenadas representa su MSE.

Se observa que los modelos 3 últimos modelos son los que más MSE tienen; y ocurre lo mismo con los modelos 7,8 y 9 aunque tienen menor error cuadrático medio. Representamos los 5 primeros modelos de nuevo para poder verlos mejor:

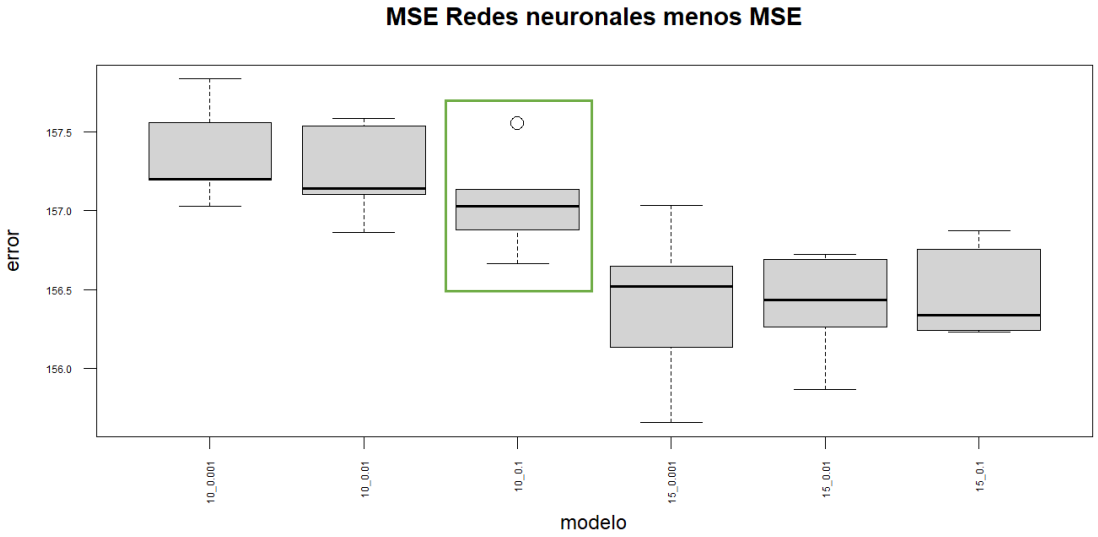

Figura 5.6: Redes neuronales con menos MSE. El eje de abscisas representa los 6 modelos y el eje de ordenadas representa su MSE.

Escogemos el tercer modelo porque tiene una buena varianza, aunque tenga un poco mayor el MSE respecto a los modelos 7, 8 y 9. El modelo escogido es una red neuronal de 10 nodos y con decay de 0.1.

Calculando su R^2 obtenemos 0.6247. Con este modelo somos capaces de explicar aproximadamente el 62% de la variación en la variable de respuesta. Comparado con regresión (31%), se ha mejorado considerablemente la bondad de ajuste de este modelo.

5.4. Modelos basados en árboles

Todos los tipos de árboles que se van a robar en este apartado se van a programar también en R.

5.4.**1.** Bagging

Para la construcción de árboles mediante el modelo Bagging, se variará el número de observaciones por hoja, y el número de árboles que se construirán para promediarlos. El número de variables en este caso se deja con el número de todas las variables independientes Para ello, utilizar la macro *cruzadarf* en R con reemplazamiento, a la que pasaremos distintos valores los parámetros del árbol mediante unos bucles anidados por orden de aparición de cada variable.

A continuación, se listan los valores dados a los parámetros para los modelos:

- *node size*: 10,20,30,40.
- *ntrees*: 100, 200, 500, 1000.
- mtry: 26 (todas las variables).

Tenemos 16 combinaciones modelos. Representamos ordenados por su MSE en la Figura 5.7.

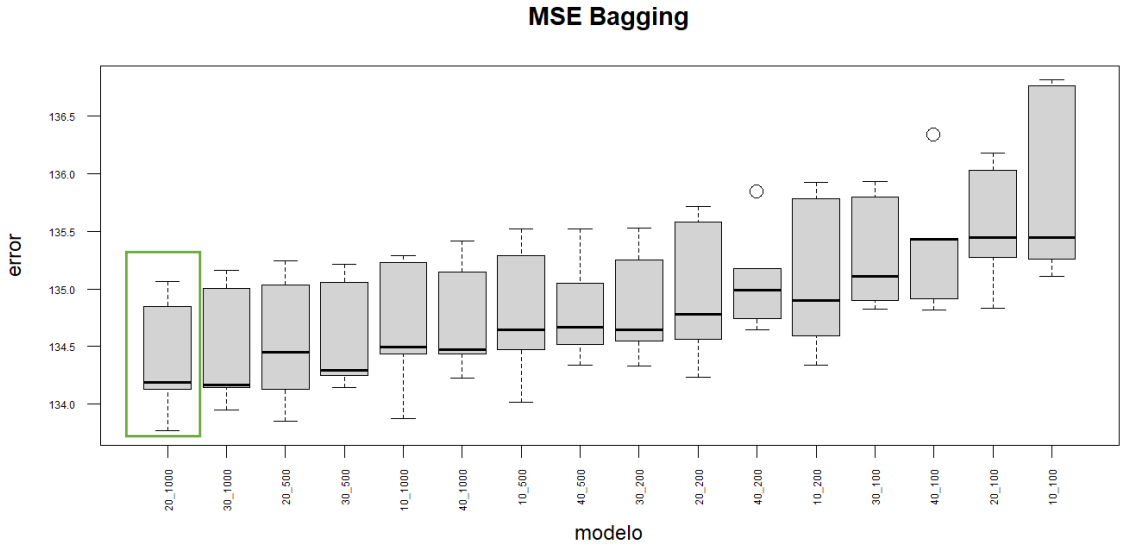

Figura 5.7: Bagging con la selección de variables elegida. El eje de abscisas representa los 16 modelos y el eje de ordenadas representa su MSE

Observando los modelos, se puede ver que cuanto más aumentan los parámetros del número de observaciones por hoja, y el número de árboles que se crean, el error cuadrático medio va disminuyendo.

Escogemos el modelo 8 debido a que es de los que menos MSE tiene y varianza no es de los modelos que más tiene. No se ha cogido el modelo que está en la posición 11 (40_200) porque tiene un MSE más elevado, pero era de los mejores modelos en cuanto a varianza. El modelo elegido cuenta con 20 observaciones por hoja y se construirán 1000 árboles (20_1000). El R^2 de este modelo es de 0.6776. Comparado con regresión y redes neuronales, se ajusta más a la variación de la variable objetivo.

5.4.2. Random Forest

Como se ha explicado anteriormente, es una modificación del Bagging que consiste en incorporar aleatoriedad en las variables utilizadas para segmentar cada nodo del árbol. Para ello, combina varios sets de variables y de observaciones para poder definir un modelo que no se sobreajuste a los datos proporcionados.

Para construir los modelos de Random Forest, haremos las mismas combinaciones que en los modelos de Bagging pero añadiremos un parámetro que recoge el número de variables input que va a utilizar. Para ello, utilizamos la macro *cruzadarf* en R con reemplazamiento, a la que pasaremos distintos valores los parámetros del árbol como se comentó en el algoritmo anterior. Se prueban los mismos valores para las variables que se tienen en común con Bagging, pero tenemos 64 modelos de Random Forest al incluir una variable más.

Los parámetros que vamos a probar:

- *node size*: 10, 20, 30,40.
- *ntrees*: 100, 200, 500, 1000.
- mtry: 3,7,13 y 26 (estudiamos de Bagging para compararlo en el mismo gráfico).

Debido a que son muchos modelos a representar, en el Anexo IV vamos a incluir los gráficos para todos los modelos. Para poder ver los mejores, nos quedamos con los 20 mejores modelos (Figura 5.8)

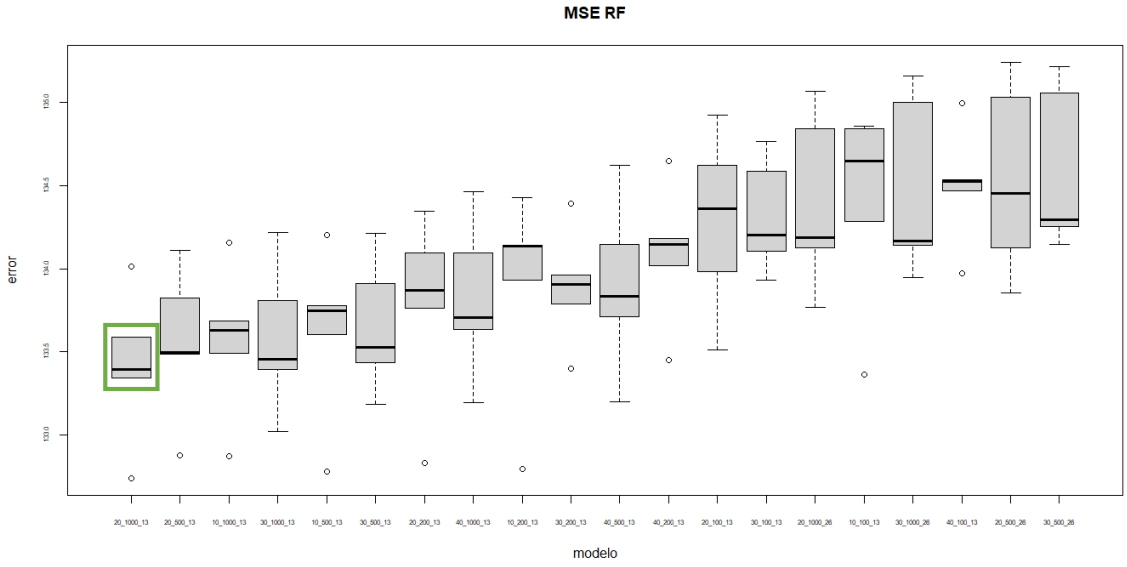

Figura 5.8: Random Forest con la selección de variables elegida. El eje de abscisas representa los 20 mejores modelos y el eje de ordenadas representa su MSE

Por ser el modelo con menor error cuadrático medio de todos, nos quedamos con el primer modelo. Este modelo tiene 20 observaciones por hoja, y 1000 árboles construidos (estos parámetros son iguales que los del mejor modelo de Bagging), pero con 13 variables (20_1000_13).

Si comparamos los dos, Random Forest es mejor modelo porque tiene menos MSE y varianza que el modelo de Bagging con los mismos parámetros (menos el número de variables).

El R^2 de este modelo es de 0.6804. Es el mejor R^2 que se ha obtenido por el momento, sigue siendo un poco mejor que el R^2 de Bagging (0.6776).

5.4.3. Gradient Boosting

Para construir los modelos con este algoritmo, se van a monitorizar los parámetros de observaciones mínimas en cada hoja final, la constante *shrinkage* para corregir posibles

estimaciones erróneas y el número de árboles que se van a construir. Utilizamos la macro *cruzadagbm*.

Parámetros monitorizados:

- *ntrees*: 100,200,500,1000.
- *n.minobsinnode*: 10,100,500,1000,2000.
- *shrinkage:* 0.001,0.05,0.1,0.3, 0.5.
- *interaction.depth:* El parámetro que mide la profundidad del árbol se ha dejado constante a 2.

Se construyen 100 árboles. Debido a que son muchos modelos a representar, como ocurría con Random Forest, en el Anexo IV. Vamos a incluir los gráficos para todos los modelos. Para poder ver los mejores, nos quedamos con los 20 mejores modelos (Figura 5.9)

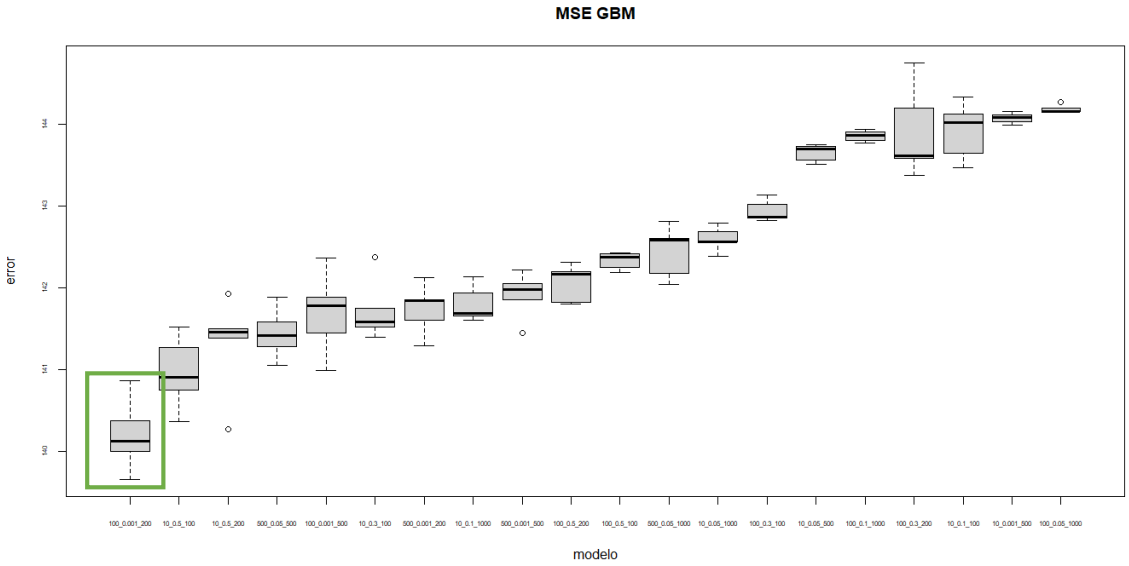

Figura 5.9: Gradient Boosting con los modelos finales escogidos. El eje de abscisas representa los 20 mejores modelos y el eje de ordenadas representa su MSE.

El modelo con menos MSE es el primero sin lugar a dudas, no es el modelo con menos varianza pero no es excesiva. El modelo escogido (100_0.001_200) que consta de 100 observaciones por hoja, 200 árboles construidos y *shrinkage* 0.001. Su R^2 no es de los mejores que se han obtenido: 0.48426.

5.4.4. Xgboost

Para construir los modelos con este algoritmo, se van a monitorizar los mismos parámetros que en Gradient Boosting: los parámetros de observaciones mínimas en cada hoja final, la constante *shrinkage* para corregir posibles estimaciones erróneas y el número de árboles que se van a construir. También se añade la penalización γ , λ (en este caso de estudio será 0). Utilizamos la macro *cruzadaxgbm*.

Los parámetros que se van a monitorizar son los siguientes:

- *eta:* 0.1,0.05,0.03,0.01,0.001
- *max.depth:* 2,4,6
- *min_child_weight:* 5,10,20.
- *nround:* 100,500,1000,5000.

Se construyen 180 árboles. Como en el caso de los dos modelos anteriores, al ser muchos modelos a representar, en el Anexo IV vamos a incluir los gráficos para todos los modelos. Para poder ver bien los modelos, nos quedamos con los 20 mejores modelos (Figura 5.10)

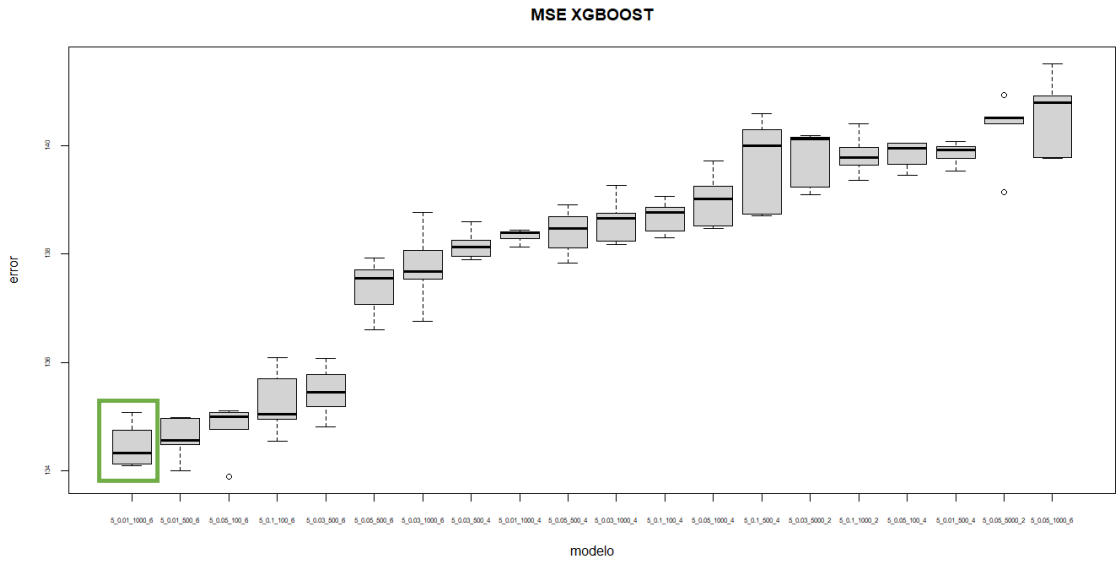

Figura 5.10: Xgboost con los 20 mejores modelos escogidos. El eje de abscisas representa los modelos del subgrupo 1 y el eje de ordenadas representa su MSE.

Por ser el modelo con menor error cuadrático medio de todos, nos quedamos con el primer modelo (5_0.01_100_6). Este modelo tiene los parámetros *child* con valor 5, eta que equivale a 0.01, número de rondas a 1000 y profundidad de 6. Si comparamos los dos, X Gradient Boosting es mejor modelo que el que se ha escogido como el mejor de Gradient Boosting, porque tiene menos MSE y varianza.

Su R^2 de este modelo es de 0.6791. Con este modelo sólo capaces de explicar aproximadamente el 67% de la variación en la variable de respuesta. Para Gradient Boosting es muy parecido, pero algo inferior.

5.4.5. SVM

Como ya se comentó en el Capítulo 3, se iban a probar dos tipos de SVM: lineal y radial.

SVM lineal

Para construir los modelos con este *kernel*, se va a monitorizar el parámetro *C* de regularización de los árboles que se van a construir. Utilizamos la función *cruzadaSVMbin*.

Los valores que va a tomar *C*: 0.01,0.1,1,10

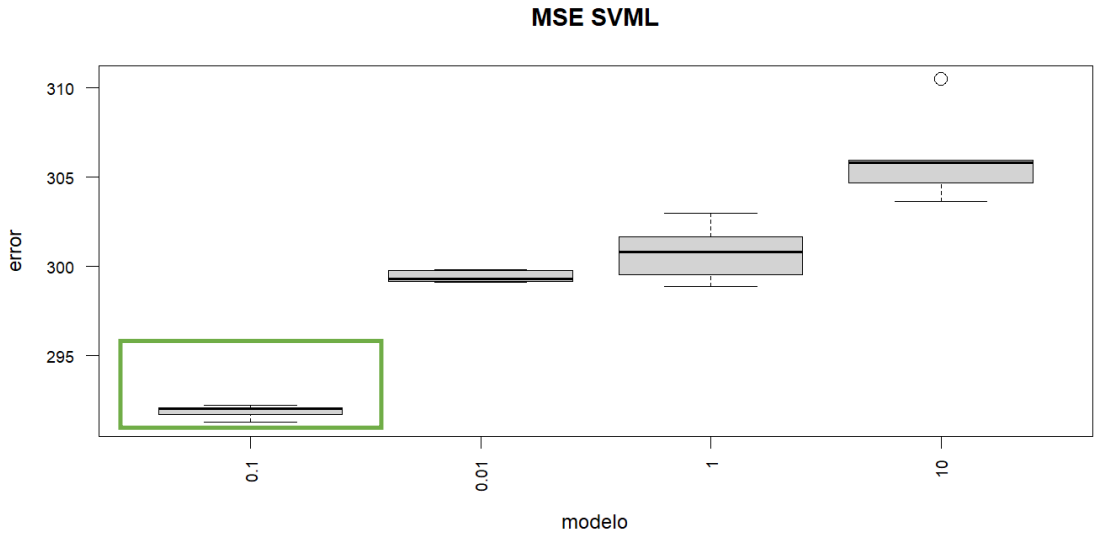

Figura 5.11: SVM lineal con los modelos finales escogidos. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

En la Figura 5.11 Podemos ver representados los cuatro modelos probados. Escogemos el modelo que utiliza el parámetro C con el valor 0.1. Ya que es el que mejor MSE tiene.

El R^2 de este modelo es de 0.3035. Es de los peores R^2 calculados hasta ahora, sólo lo supera por muy poco la regresión.

SVM RBF

En este caso, se van a probar los modelos en función de dos parámetros: *C* y *sigma*. Utilizamos la macro *cruzadaSVMbinRBF*.

Los valores con los que vamos a variar los parámetros mencionados:

- *C* = 0.01,0.1,1,10
- *sigma* = 0.01,0.1,1,10

Tenemos 16 combinaciones de modelos que representamos ordenados por su MSE en la Figura 5.12:

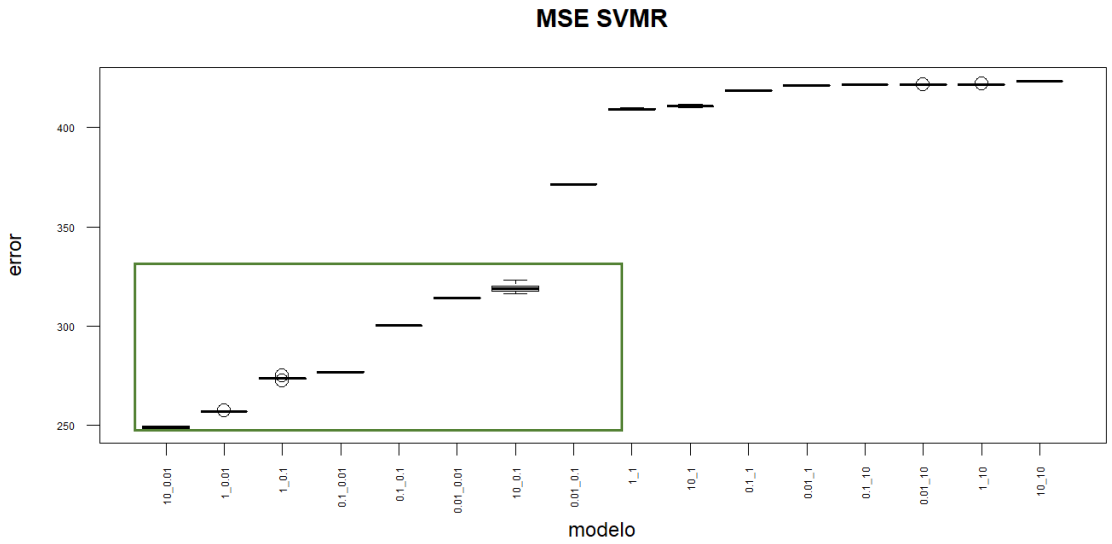

Figura 5.12: SVM RBF con los modelos iniciales escogidos. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE.

Se puede observar que los mejores modelos son los dos primeros. Para poder verlos mejor representamos:

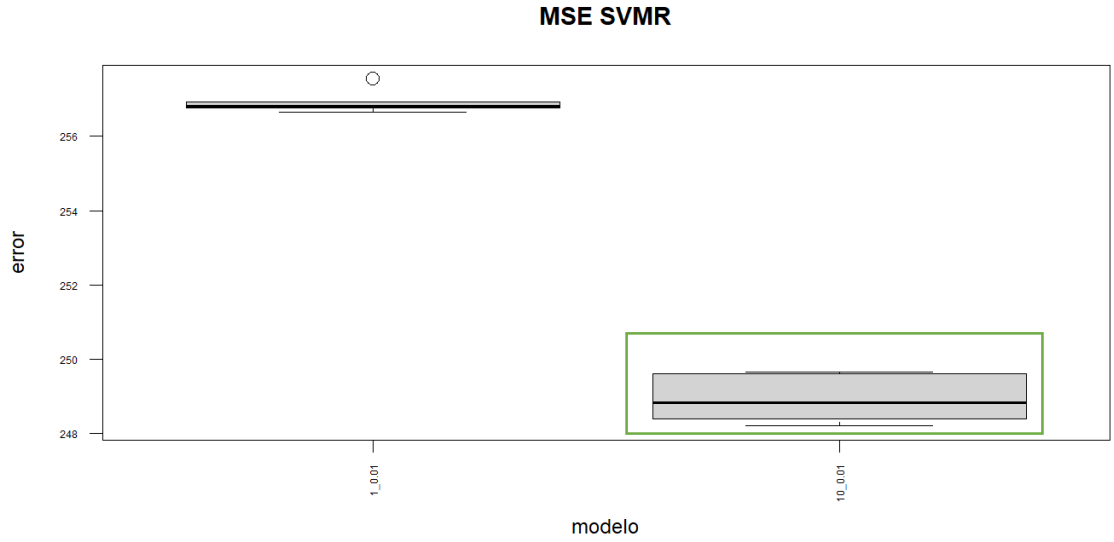

Figura 5.13: SVM RBF con los modelos finales escogidos. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE.

Finalmente, nos quedamos con el segundo modelo que utiliza el parámetro C con el valor 0.1. Ya que es el que mejor MSE tiene. Calculando su R^2 obtenemos 0.4059. Aunque sea mejor que en SVM lineal y un poco mejor que regresión, es de los peores R^2 que hemos obtenido con los modelos que hemos probado.

5.5. Comparación de modelos

A continuación, vamos a representar todos los mejores modelos elegidos en el apartado anterior y se van a comparar con su MSE y R^2:

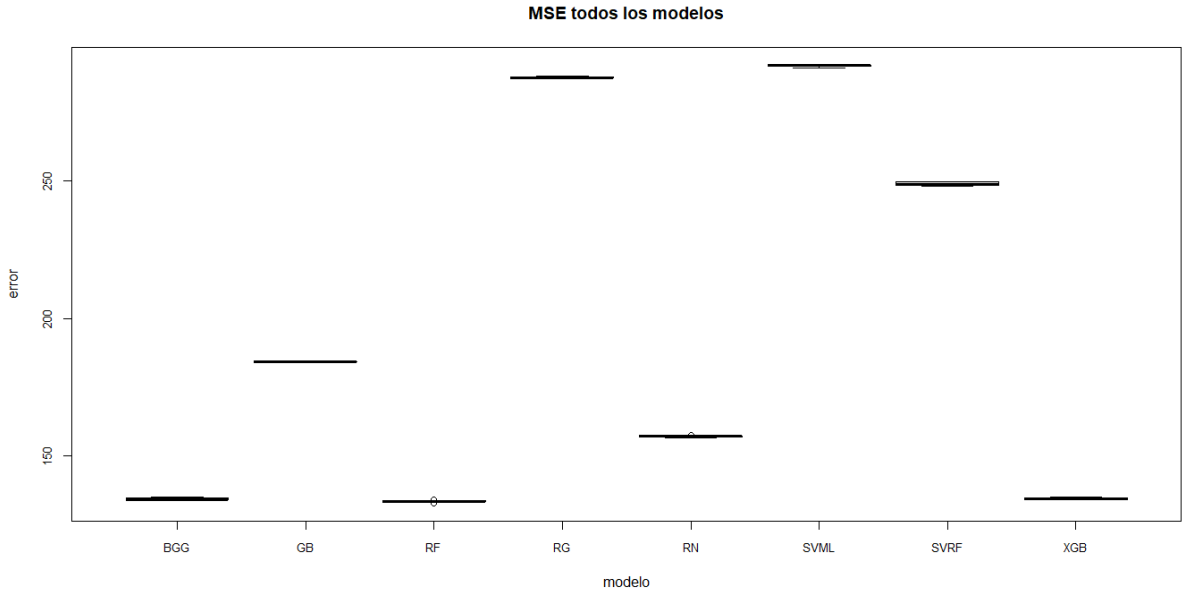

Figura 5.14: Todos los modelos seleccionados. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE.

Modelo	R2
Random Forest	0.6804
Xgboost	0.6791
Bagging	0.6776
Redes neuronales	0.6248
Grandient Boosting	0.4843
SVM Radial	0.4059
Regresión lineal	0.3139
SVM Lineal	0.3036

Figura 5.15: Tabla de los mejores modelos escogidos y su R^2.

Como se puede ver en esta tabla, los modelos más competitivos son Random Forest, Xgboost, Bagging y redes neuronales ya que son los que más R^2 tienen. Por ello, son los modelos que pueden explicar casi el 70% de la variación de la variable objetivo.

En la Figura 5.16 podemos ver como los modelos con menos MSE son Bagging, Random Forest y Xgboost. Vamos a observarlos con más claridad:

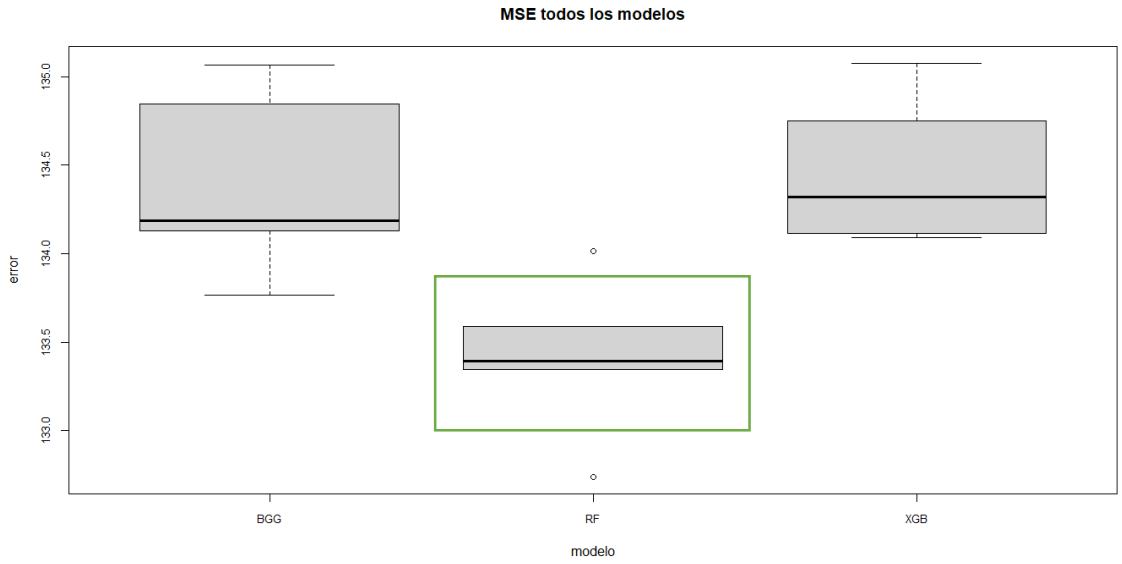

Figura 5.16: Bagging, Random Forest y Gradient Boosting. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE.

Claramente el mejor modelo es Random Forest ya que tiene un error cuadrático medio más bajo que el de los otros dos modelos y menos varianza. También es el que más R^2 tiene del resto de modelos, pero es muy pequeña la diferencia con ellos en ese aspecto.

Vamos a tratar de entender un poco que ha hecho este modelo, para ello, vamos a ver la importancia de las variables independientes que componen este modelo. La importancia es la cuantificación de la contribución individual de una variable input al modelo.

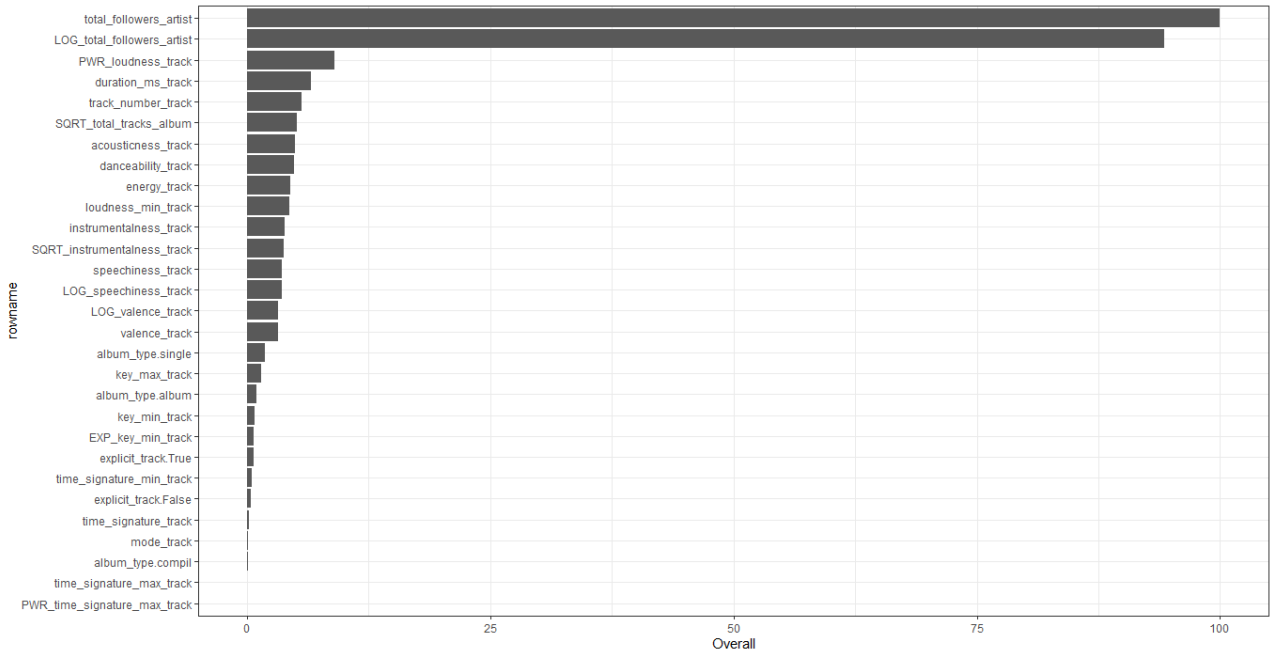

Figura 5.17: Gráfico de representación de la importancia de las variables en %.

Variable	Overall (en %)
total_followers_artist	100,0000
LOG_total_followers_artist	94,2178
PWR_loudness_track	8,9944
duration_ms_track	6,5808
track_number_track	5,6255
SQRT_total_tracks_album	5,1251
acousticness_track	4,9799
danceability_track	4,8330
energy_track	4,4449
loudness_min_track	4,3627
instrumentalness_track	3,8429
SQRT_instrumentalness_track	3,7570
speechiness_track	3,6039
LOG_speechiness_track	3,5885
LOG_valence_track	3,2007
valence_track	3,1981
album_type,single	1,8282
key_max_track	1,5144
album_type,album	0,9528
key_min_track	0,7466
EXP_key_min_track	0,7307
explicit_track,True	0,7026
time_signature_min_track	0,5013
explicit_track,False	0,4338
time_signature_track	0,2479
mode_track	0,1127
album_type,compil	0,0982
time_signature_max_track	0,0090
PWR_time_signature_max_track	0,0000

Figura 5.18: Tabla de representación de la importancia de las variables en %.

Observamos en las figuras 5.17 y 5.18 que las variables que más contribuyen al modelo son total_followers_artist y LOG_total_followers_artist (variables original y transformada). Puede tener lógica que sean las variables más influyentes ya que si un artista es muy famoso porque cuenta con muchos seguidores, la canción será más popular. Aunque existan casos de que canciones de artistas desconocidos se hagan muy populares.

Resumiendo, estas dos variables con diferencia son las que más contribuyen (100% y 94% resectivamente), ya que la siguiente variable que más contribuye es PWR_loudness_track con un 9%. Las variables categóricas apenas son importantes, al igual que las 11 variables últimas de la tabla (Figura 5.18)

5.6. Ensamblado

Como ya se comentaba en el Capítulo 3, este método se basa en la construcción de predicciones a partir de la combinación de varios modelos. En este trabajo realizaremos la técnica de combinado de modelos: Stacking.

Se han realizado un total de 50 ensamblados, han sido combinaciones tanto de 2, 3, 4 y 5 modelos ganadores del apartado anterior. Se ha excluido el modelo ganador de Bagging porque es un caso de Random Forest y ya se ha demostrado que es un modelo menos competitivo. Las combinaciones se pueden ver en la en el gráfico de la Figura.

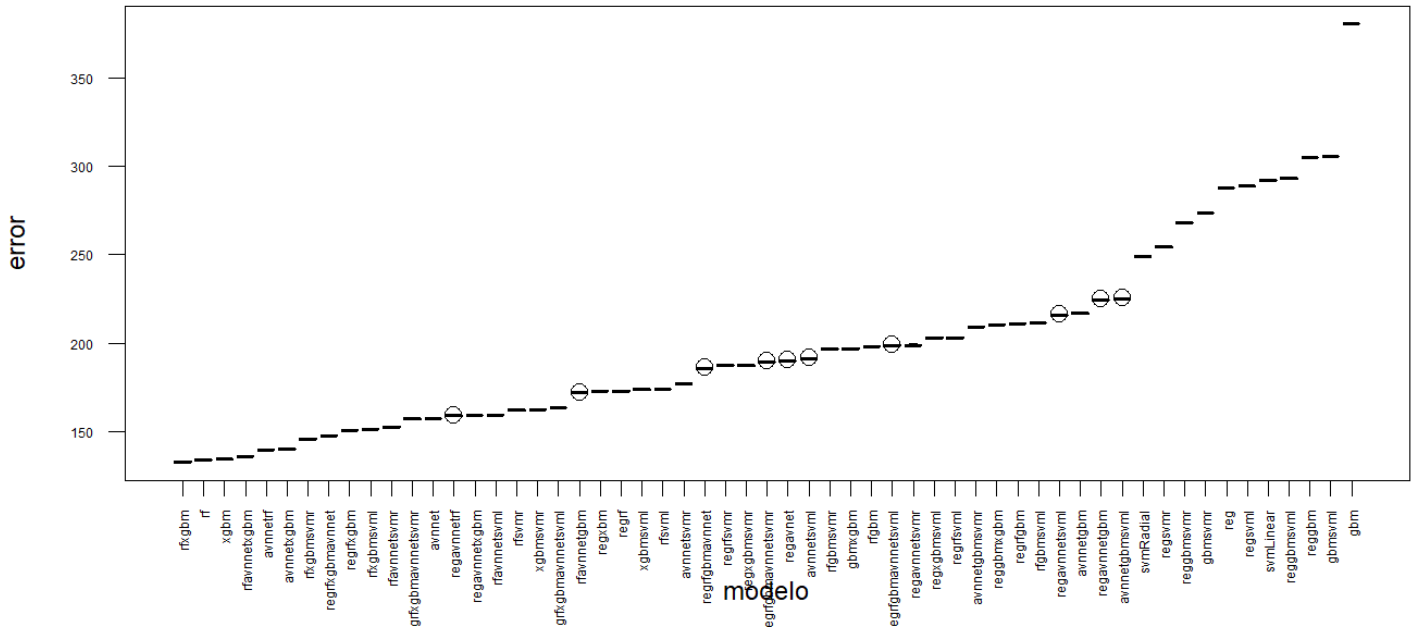

Figura 5.19: Gráfico de los modelos ensamblados. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE.

Como se puede observar en la figura, los mejores modelos (porque tienen menos error cuadrático medio) son los cuatro primeros:

Random Forest + Xgboost
Random Forest
Xgboost
Random Forest +RN + Xgboost

Figura 5.20: Tabla mejores modelos de ensamblado.

Representamos los modelos detallados en la tabla 5.20:

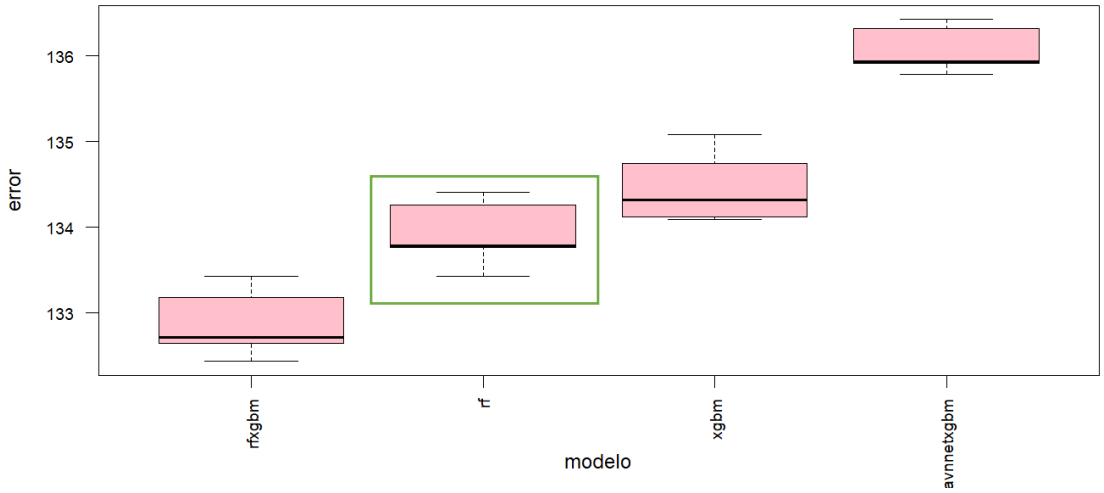

Figura 5.21: Gráfico de los mejores modelos ensamblados. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE.

Se puede observar que el modelo de ensamblado ganador, porque tiene el MSE más bajo. se basa en los dos mejores modelos como comentamos en el apartado anterior (Random Forest y Xgboost). Los cuatro modelos son parecidos en varianza, pero escogemos el modelo Random Forest por el criterio de parsimonia. Es decir, escogemos el modelo más sencillo porque la diferencia de MSE entre este y el ensamblado ganador no es muy significativa.

Concluimos con que el modelo Random Forest ganador tiene 20 observaciones por hoja, 1000 árboles construidos y con 13 variables.

Cabe destacar que el ensamblado es una buena técnica para mejorar los modelos sin duda, mientras que con los modelos SVM no se han obtenido buenos resultados para el planteamiento de este problema.

También se puede comentar que Random Forest ha superado a Bagging, por lo que el parámetro de seleccionar un cierto número de variables ha sido muy útil. Y Gradient Boosting no ha tenido buenos resultados frente a Xgboost con sus parámetros de regularización.

CAPÍTULO 6

CONCLUSIONES Y LÍNEAS FUTURAS

Al principio de este trabajo, se habían marcado una serie de objetivos que se han ido cumpliendo y desarrollando a lo largo de este proyecto:

- Creación de una base de datos de canciones relevante a través de la API de *Spotify,* usando el lenguaje de programación Python.
  - Se ha logrado crear un base de datos considerable donde se ha recogido información de 14094 canciones: características de la música, del artista y del álbum. Todo ello se diseñó que no hubiese valores ausentes.
  - El proceso de obtención de las canciones ha sido en dos partes:
    - A partir de las listas de canciones más importantes en todos los países en los que se encuentra *Spotify*.
    - Con los artistas que aparecían en la parte anterior, se ha extraído sus 10 canciones más importantes (como máximo).

- Realizar un análisis exploratorio en SAS Miner sobre la base de datos para poder realizar su depuración y destacar aspectos importantes de las variables.
- Depurar los datos también en SAS Miner para adecuar las distintas variables de entrada útiles, para su posterior análisis.
  - En esta fase de preprocesamiento de los datos, se han modificado y descartado algunas variables con el fin de que no tener variables con poca información.
  - Se han realizado transformaciones de los predictores con el fin de intentar aumentar la correlación con la variable objetivo y mejorar los modelos. Por ello, de cara a la selección de variables, se han utilizados dos sets: variables originales y transformadas.

- Probar distintas selecciones de variables en SAS para optimizar los modelos de Machine Learning y escoger las variables que más aportan información.
  - Con los dos sets de variables que se han escogido (variables y transformadas, se han realizado varias pruebas con distintos criterios de información (AIC, BIC, SBC) y algoritmos (Backward, Forward y Stepwise).

- Utilizar métodos de aprendizaje estadístico en el lenguaje de programación R: regresión lineal, redes neuronales y modelos basados en árboles.
  - Los métodos de aprendizaje estadístico son detallados en el Capítulo 3 y puestos en práctica en el Capítulo 5, variando sus parámetros y entrenándolos mediante validación cruzada.

- Se prueba la técnica de ensamblado a partir de las predicciones obtenidas de los métodos de aprendizaje automático.

- Evaluación de todos los modelos entrenados con la finalidad de escoger el mejor modelo de predicción.
  - Mediante el MSE (error cuadrático medio) y R^2 (mide la bondad de ajuste de un modelo) se han evaluado los modelos y escogido finalmente Random Forest.

Estos objetivos específicos tenían un cometido, resolver el problema propuesto: predecir la popularidad que tiene una canción en Spotify. Como primer análisis, son buenos los resultados que hemos obtenido ya que nuestros mejores modelos son capaces de explicar casi el 70% de variación de la variable de respuesta, explicando su relación con las variables independientes.

Para continuar con este estudio en el futuro, se proponen ciertas pautas para mejorar y continuar con el análisis:

- Disposición de mejor infraestructura a la hora de entrenar los modelos. Las ejecuciones de algunos algoritmos pueden ser tediosas a la hora de querer optimizar los tiempos. Una idea para solventar esto podría ser disponer de un servidor virtual.

- Emplear otras variantes de las técnicas de Machine Learning, como las redes neuronales convolucionales. Así como otras herramientas que permitan variar más parámetros de los algoritmos.

- Ampliar las observaciones de la base de datos, extrayendo las otras canciones que aparecen en el álbum que aparezca para cada canción.

- Para obtener más variables para la base de datos, sería interesante extraer datos de las redes sociales para las canciones, y así se sale un poco del mundo "hermético" de *Spotify* a la hora de dar información sobre sus métricas.

Con estas mejoras se podría continuar abordando el problema que se ha planteado sobre la predicción de popularidad de las canciones y poder plantear nuevos retos relacionados con la industria musical.

ANEXO I: Tabla con países *Spotify*

Mostramos la tabla países Spotify codificado según ISO 3266-1 alpha-2 que se hace referencia en el Capítulo 4.(Figura I .1: Tabla con países donde está en el mercado *Spotify*)

Código ISO 3266-1 alpha-2	País
AD	Andorra
AR	Argentina
AT	Austria
AU	Australia
BE	Bélgica
BG	Bulgaria
BO	Bolivia
BR	Brasil
CH	Suiza
CL	Chile
CO	Colombia
CR	Costa Rica
CY	Chipre
CZ	República Checa
DK	Dinamarca
DO	República Dominicana
EC	Ecuador
EE	Estonia
ES	España
FI	Finlandia
FR	Francia
GB	Reino Unido
GR	Grecia
GT	Guatemala
HK	Hong Kong
HN	Honduras
HU	Hungría
IE	Irlanda
IS	Islandia
IT	Italia
LI	Liechtenstein
LT	Lituania
LU	Luxemburgo
LV	Letonia
MC	Mónaco
MT	Malta
MY	Malasia
NI	Nicaragua
NL	Países Bajos nota 4
NO	Noruega
NZ	Nueva Zelanda
PA	Panamá
PE	Perú
PH	Filipinas
PL	Polonia
PT	Portugal
PY	Paraguay

Código ISO 3266-1 alpha-2	País
RO	Rumania
SE	Suecia
SG	Singapur
SI	Eslovenia
SK	Eslovaquia
SV	El Salvador
TR	Turquía
TW	Taiwán
UY	Uruguay

ANEXO II: Código SAS para selección de variables

```
libname datos 'C:\Users\cris\Documents\TFM';
/*********************************************************VARIABLES
ORIGINALES*********************************************************/
proc print data=datos.variables_originales_train;run;
data=datos.variables_originales_train;run;

/*STEPWISE*/
%macro
randomselect(data=,listclass=,vardepen=,modelo=,criterio=,sinicio=,sfi
nal=,fracciontrain=,directorio=&directorio);
options nocenter linesize=256;
proc printto print="&directorio\kk.txt";run;
data _null_;file "&directorio\cosa.txt" linesize=2000;run;
%do semilla=&sinicio %to &sfinal;
proc surveyselect data=&data rate=&fracciontrain out=sal1234
seed=&semilla;run;
ods output   SelectionSummary=modelos;
ods output    SelectedEffects=efectos;
ods output    Glmselect.SelectedModel.FitStatistics=ajuste;
proc glmselect data=sal1234  plots=all seed=&semilla;
  class &listclass;
  model &vardepen= &modelo/ selection=stepwise(select=&criterio
choose=&criterio) details=all stats=all;
run;
ods graphics off;
ods html close;
data union;i=5;set efectos;set ajuste point=i;run;
data _null_;semilla=&semilla;file "&directorio\cosa.txt" mod
linesize=2000;set union;put effects ;run;
%end;
proc printto ;run;
data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto @@;
if efecto ne 'Intercept' then output;
run;
proc freq data=todos;tables efecto /out=sal;run;
proc sort data=sal;by descending count;
proc print data=sal;run;

data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto $ &&;
run;
proc freq data=todos;tables efecto /out=salefec;run;
proc sort data=salefec;by descending count;
proc print data=salefec;run;
data _null_;set salefec;put efecto;run;
%mend;
```

```
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=AIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=BIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=SBC,
sinicio=12345,
```

```
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;


/*BACKWARD*/
%macro
randomselect(data=,listclass=,vardepen=,modelo=,criterio=,sinicio=,sfi
nal=,fracciontrain=,directorio=&directorio);
options nocenter linesize=256;
proc printto print="&directorio\kk.txt";run;
data _null_;file "&directorio\cosa.txt" linesize=2000;run;
%do semilla=&sinicio %to &sfinal;
proc surveyselect data=&data rate=&fracciontrain out=sal1234
seed=&semilla;run;
ods output   SelectionSummary=modelos;
ods output    SelectedEffects=efectos;
ods output    Glmselect.SelectedModel.FitStatistics=ajuste;
proc glmselect data=sal1234  plots=all seed=&semilla;
  class &listclass;
  model &vardepen= &modelo/ selection=backward(select=&criterio
choose=&criterio) details=all stats=all;
run;
ods graphics off;
ods html close;
data union;i=5;set efectos;set ajuste point=i;run;
data _null_;semilla=&semilla;file "&directorio\cosa.txt" mod
linesize=2000;set union;put effects ;run;
%end;
proc printto ;run;
data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto @@;
if efecto ne 'Intercept' then output;
run;
proc freq data=todos;tables efecto /out=sal;run;
proc sort data=sal;by descending count;
proc print data=sal;run;

data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto $ &&;
run;
proc freq data=todos;tables efecto /out=salefec;run;
proc sort data=salefec;by descending count;
proc print data=salefec;run;
data _null_;set salefec;put efecto;run;
%mend;

%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
```

```
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=AIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=BIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=SBC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;

/*FORWARD*/
```

```
%macro
randomselect(data=,listclass=,vardepen=,modelo=,criterio=,sinicio=,sfi
nal=,fracciontrain=,directorio=&directorio);
options nocenter linesize=256;
proc printto print="&directorio\kk.txt";run;
data _null_;file "&directorio\cosa.txt" linesize=2000;run;
%do semilla=&sinicio %to &sfinal;
proc surveyselect data=&data rate=&fracciontrain out=sal1234
seed=&semilla;run;
ods output   SelectionSummary=modelos;
ods output    SelectedEffects=efectos;
ods output    Glmselect.SelectedModel.FitStatistics=ajuste;
proc glmselect data=sal1234  plots=all seed=&semilla;
  class &listclass;
  model &vardepen= &modelo/ selection=forward(select=&criterio
choose=&criterio) details=all stats=all;
run;
ods graphics off;
ods html close;
data union;i=5;set efectos;set ajuste point=i;run;
data _null_;semilla=&semilla;file "&directorio\cosa.txt" mod
linesize=2000;set union;put effects ;run;
%end;
proc printto ;run;
data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto @@;
if efecto ne 'Intercept' then output;
run;
proc freq data=todos;tables efecto /out=sal;run;
proc sort data=sal;by descending count;
proc print data=sal;run;

data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto $ &&;
run;
proc freq data=todos;tables efecto /out=salefec;run;
proc sort data=salefec;by descending count;
proc print data=salefec;run;
data _null_;set salefec;put efecto;run;
%mend;

%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=AIC,
sinicio=12345,
```

```
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=BIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_originales_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=SBC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;



/*********************************************************VARIABLES
TRANSFORMADAS*********************************************************/
proc print data=datos.variables_transformadas_train;run;
data=datos.variables_transformadas_train;run;

/*STEPWISE*/
```

```
%macro
randomselect(data=,listclass=,vardepen=,modelo=,criterio=,sinicio=,sfi
nal=,fracciontrain=,directorio=&directorio);
options nocenter linesize=256;
proc printto print="&directorio\kk.txt";run;
data _null_;file "&directorio\cosa.txt" linesize=2000;run;
%do semilla=&sinicio %to &sfinal;
proc surveyselect data=&data rate=&fracciontrain out=sal1234
seed=&semilla;run;
ods output   SelectionSummary=modelos;
ods output    SelectedEffects=efectos;
ods output    Glmselect.SelectedModel.FitStatistics=ajuste;
proc glmselect data=sal1234  plots=all seed=&semilla;
  class &listclass;
  model &vardepen= &modelo/ selection=stepwise(select=&criterio
choose=&criterio) details=all stats=all;
run;
ods graphics off;
ods html close;
data union;i=5;set efectos;set ajuste point=i;run;
data _null_;semilla=&semilla;file "&directorio\cosa.txt" mod
linesize=2000;set union;put effects ;run;
%end;
proc printto ;run;
data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto @@;
if efecto ne 'Intercept' then output;
run;
proc freq data=todos;tables efecto /out=sal;run;
proc sort data=sal;by descending count;
proc print data=sal;run;

data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto $ &&;
run;
proc freq data=todos;tables efecto /out=salefec;run;
proc sort data=salefec;by descending count;
proc print data=salefec;run;
data _null_;set salefec;put efecto;run;
%mend;

%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
```

```
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=AIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=BIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
```

```
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=SBC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;


/*BACKWARD*/
%macro
randomselect(data=,listclass=,vardepen=,modelo=,criterio=,sinicio=,sfi
nal=,fracciontrain=,directorio=&directorio);
options nocenter linesize=256;
proc printto print="&directorio\kk.txt";run;
data _null_;file "&directorio\cosa.txt" linesize=2000;run;
%do semilla=&sinicio %to &sfinal;
proc surveyselect data=&data rate=&fracciontrain out=sal1234
seed=&semilla;run;
ods output   SelectionSummary=modelos;
ods output    SelectedEffects=efectos;
ods output    Glmselect.SelectedModel.FitStatistics=ajuste;
proc glmselect data=sal1234  plots=all seed=&semilla;
  class &listclass;
  model &vardepen= &modelo/ selection=backward(select=&criterio
choose=&criterio) details=all stats=all;
run;
ods graphics off;
ods html close;
data union;i=5;set efectos;set ajuste point=i;run;
data _null_;semilla=&semilla;file "&directorio\cosa.txt" mod
linesize=2000;set union;put effects ;run;
%end;
proc printto ;run;
data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto @@;
if efecto ne 'Intercept' then output;
run;
proc freq data=todos;tables efecto /out=sal;run;
proc sort data=sal;by descending count;
proc print data=sal;run;

data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto $ &&;
run;
```

```
proc freq data=todos;tables efecto /out=salefec;run;
proc sort data=salefec;by descending count;
proc print data=salefec;run;
data _null_;set salefec;put efecto;run;
%mend;

%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=AIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
```

```
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=BIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=SBC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;

/*FORWARD*/
%macro
randomselect(data=,listclass=,vardepen=,modelo=,criterio=,sinicio=,sfi
nal=,fracciontrain=,directorio=&directorio);
options nocenter linesize=256;
proc printto print="&directorio\kk.txt";run;
data _null_;file "&directorio\cosa.txt" linesize=2000;run;
%do semilla=&sinicio %to &sfinal;
proc surveyselect data=&data rate=&fracciontrain out=sal1234
seed=&semilla;run;
ods output   SelectionSummary=modelos;
ods output    SelectedEffects=efectos;
ods output    Glmselect.SelectedModel.FitStatistics=ajuste;
proc glmselect data=sal1234  plots=all seed=&semilla;
  class &listclass;
  model &vardepen= &modelo/ selection=forward(select=&criterio
choose=&criterio) details=all stats=all;
run;
```

```
ods graphics off;
ods html close;
data union;i=5;set efectos;set ajuste point=i;run;
data _null_;semilla=&semilla;file "&directorio\cosa.txt" mod
linesize=2000;set union;put effects ;run;
%end;
proc printto ;run;
data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto @@;
if efecto ne 'Intercept' then output;
run;
proc freq data=todos;tables efecto /out=sal;run;
proc sort data=sal;by descending count;
proc print data=sal;run;

data todos;
infile "&directorio\cosa.txt" linesize=2000;
length efecto $ 1000;
input efecto $ &&;
run;
proc freq data=todos;tables efecto /out=salefec;run;
proc sort data=salefec;by descending count;
proc print data=salefec;run;
data _null_;set salefec;put efecto;run;
%mend;

%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
EXP_key_min_track EXP_loudness_mean_track EXP_mode_max_track
EXP_mode_track LOG_speechiness_track LOG_tempo_mean_track
LOG_tempo_min_track LOG_total_followers_artist
LOG_valence_track PWR_danceability_track PWR_key_mean_track
PWR_key_track PWR_loudness_max_track PWR_loudness_min_track
PWR_loudness_track PWR_tempo_track
PWR_time_signature_max_track PWR_time_signature_min_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_liveness_track
SQRT_total_tracks_album SQR_mode_mean_track
SQR_mode_min_track acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_mean_track key_min_track key_track
liveness_track loudness_max_track loudness_mean_track
loudness_min_track loudness_track mode_max_track mode_mean_track
mode_min_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_mean_track
time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
criterio=AIC,
sinicio=12345,
sfinal=12445,
fraccion train=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_transformadas_train,
```

```
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=BIC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;
%randomselect(data=datos.variables_transformadas_train,
listclass=REP_disc_number_track album_type explicit_track,
vardepen=popularity_track,
modelo=REP_disc_number_track album_type explicit_track
acousticness_track danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_mean_track key_min_track
key_track liveness_track
loudness_max_track loudness_mean_track loudness_min_track
loudness_track mode_max_track mode_mean_track mode_min_track
mode_track speechiness_track tempo_max_track tempo_mean_track
tempo_min_track tempo_track time_signature_max_track
time_signature_mean_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
criterio=SBC,
sinicio=12345,
sfinal=12445,
fracciontrain=0.8, directorio=C:\Users\cris\Documents\TFM);
proc print data="hola";run;
proc copy inlib=work outlib=datos ;
select salefec;
run;



/*REGRESIONES CRUZADAS*/
%macro
cruzada(archivo=,vardepen=,conti=,categor=,ngrupos=,sinicio=,sfinal=);
data final;run;
/* Bucle semillas */
%do semilla=&sinicio %to &sfinal;
      data dos;set &archivo;u=ranuni(&semilla);
      proc sort data=dos;by u;run;
      data dos;
      retain grupo 1;
      set dos nobs=nume;
      if _n_>grupo*nume/&ngrupos then grupo=grupo+1;
      run;
      data fantasma;run;
      %do exclu=1 %to &ngrupos;
```

```
            data tres;set dos;if grupo ne &exclu then vardep=&vardepen;
            proc glm data=tres noprint;/*<<<<<******SE PUEDE QUITAR EL
NOPRINT */
            %if &categor ne %then %do;class &categor;model
vardep=&conti &categor;%end;
            %else %do;model vardep=&conti;%end;
            output out=sal p=predi;run;
            data sal;set sal;resi2=(&vardepen-predi)**2;if grupo=&exclu
then output;run;
            data fantasma;set fantasma sal;run;
      %end;
      proc means data=fantasma sum noprint;var resi2;
      output out=sumaresi sum=suma mean=media;
      run;
      data sumaresi;set sumaresi;semilla=&semilla;
      data final (keep=suma media semilla);set final sumaresi;if
suma=. then delete;run;
%end;
proc print data=final;run;
%mend;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
speechiness_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final1;set final;modelo=1;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
speechiness_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final2;set final;modelo=2;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= danceability_track duration_ms_track instrumentalness_track
key_min_track loudness_min_track loudness_track
time_signature_min_track time_signature_track total_followers_artist
total_tracks_album valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final3;set final;modelo=3;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_min_track
loudness_track mode_mean_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
```

```
time_signature_max_track time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final4;set final;modelo=4;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
speechiness_track tempo_max_track tempo_mean_track tempo_min_track
tempo_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final5;set final;modelo=5;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_min_track
loudness_track mode_mean_track mode_track speechiness_track
tempo_max_track tempo_mean_track tempo_min_track tempo_track
time_signature_max_track time_signature_min_track time_signature_track
total_followers_artist total_tracks_album track_number_track
valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final6;set final;modelo=6;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
speechiness_track tempo_max_track tempo_mean_track tempo_min_track
tempo_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final7;set final;modelo=7;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_min_track
loudness_track time_signature_min_track time_signature_track
total_followers_artist total_tracks_album valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final8;set final;modelo=8;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
```

```
speechiness_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final9;set final;modelo=9;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
speechiness_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final10;set final;modelo=10;

%cruzada(archivo=datos.variables_originales_train,vardepen=popularity_
track,
conti= danceability_track duration_ms_track instrumentalness_track
key_min_track loudness_min_track loudness_track
time_signature_min_track time_signature_track total_followers_artist
total_tracks_album valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final11;set final;modelo=11;
data union;set final1 final2 final3 final4 final5 final6 final7 final8
final9 final10 final11;
proc boxplot data=union;plot media*modelo;run;
/*cruzadas transformadas*/
%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= EXP_key_min_track LOG_speechiness_track
LOG_total_followers_artist LOG_valence_track PWR_loudness_track
PWR_time_signature_max_track SQRT_instrumentalness_track
SQRT_total_tracks_album acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_min_track loudness_min_track mode_track speechiness_track
time_signature_max_track time_signature_min_track time_signature_track
total_followers_artist track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final12;set final;modelo=12;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= EXP_key_min_track LOG_speechiness_track
LOG_total_followers_artist LOG_valence_track PWR_loudness_track
PWR_time_signature_max_track SQRT_instrumentalness_track
SQRT_total_tracks_album acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_min_track loudness_min_track mode_track speechiness_track
time_signature_max_track time_signature_min_track time_signature_track
total_followers_artist track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final13;set final;modelo=13;
```

```
%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= LOG_total_followers_artist PWR_loudness_track
PWR_time_signature_max_track SQRT_instrumentalness_track
SQRT_total_tracks_album danceability_track duration_ms_track
energy_track key_min_track loudness_min_track time_signature_min_track
time_signature_track total_followers_artist valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final14;set final;modelo=14;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= EXP_key_min_track EXP_loudness_mean_track EXP_mode_track
LOG_speechiness_track LOG_tempo_mean_track LOG_tempo_min_track
LOG_total_followers_artist PWR_loudness_max_track
PWR_loudness_min_track PWR_tempo_track PWR_time_signature_max_track
SQRT_instrumentalness_track SQRT_total_tracks_album acousticness_track
danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_min_track loudness_max_track
loudness_mean_track loudness_track speechiness_track tempo_max_track
tempo_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist track_number_track
valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final15;set final;modelo=15;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= EXP_key_min_track EXP_loudness_mean_track EXP_mode_track
LOG_speechiness_track LOG_tempo_mean_track LOG_tempo_min_track
LOG_total_followers_artist PWR_loudness_max_track
PWR_loudness_min_track PWR_tempo_track PWR_time_signature_max_track
SQRT_instrumentalness_track SQRT_total_tracks_album acousticness_track
danceability_track duration_ms_track energy_track
instrumentalness_track key_max_track key_min_track loudness_max_track
loudness_mean_track loudness_track speechiness_track tempo_max_track
tempo_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist track_number_track
valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final16;set final;modelo=16;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= EXP_key_min_track EXP_mode_track LOG_speechiness_track
LOG_tempo_mean_track LOG_tempo_min_track LOG_total_followers_artist
PWR_loudness_max_track PWR_tempo_track PWR_time_signature_max_track
SQRT_energy_track SQRT_instrumentalness_track SQRT_total_tracks_album
acousticness_track danceability_track duration_ms_track
instrumentalness_track key_max_track key_min_track loudness_max_track
loudness_min_track loudness_track speechiness_track tempo_max_track
tempo_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist track_number_track
valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final17;set final;modelo=17;
```

```
%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= LOG_total_followers_artist PWR_loudness_min_track
PWR_loudness_track PWR_time_signature_max_track
SQRT_total_tracks_album danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track
time_signature_max_track time_signature_min_track
total_followers_artist valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final18;set final;modelo=18;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= EXP_key_min_track LOG_speechiness_track
LOG_total_followers_artist LOG_valence_track PWR_loudness_track
PWR_time_signature_max_track SQRT_instrumentalness_track
SQRT_total_tracks_album acousticness_track danceability_track
duration_ms_track energy_track instrumentalness_track key_max_track
key_min_track loudness_min_track mode_track speechiness_track
time_signature_max_track time_signature_min_track time_signature_track
total_followers_artist track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final19;set final;modelo=19;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= acousticness_track danceability_track duration_ms_track
energy_track instrumentalness_track key_min_track loudness_mean_track
loudness_min_track loudness_track mode_mean_track mode_track
speechiness_track time_signature_max_track time_signature_min_track
time_signature_track total_followers_artist total_tracks_album
track_number_track valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final20;set final;modelo=20;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= danceability_track duration_ms_track instrumentalness_track
key_min_track loudness_min_track loudness_track
time_signature_min_track time_signature_track total_followers_artist
total_tracks_album valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final21;set final;modelo=21;

%cruzada(archivo=datos.variables_transformadas_train,vardepen=populari
ty_track,
conti= danceability_track duration_ms_track instrumentalness_track
key_min_track loudness_min_track loudness_track mode_track
time_signature_min_track time_signature_track total_followers_artist
total_tracks_album valence_track,
categor= album_type explicit_track,
ngrupos=4,sinicio=12345,sfinal=12445);
data final22;set final;modelo=22;
data union;set final12 final13 final14 final15 final16 final17 final18
final19 final20 final21 final22;
proc boxplot data=union;plot media*modelo;run;
```

```
data union;set final1 final2 final3 final4 final5 final6 final7 final8
final9 final10 final11 final12 final13 final14 final15 final16 final17
final18 final19 final20 final21 final22;
proc boxplot data=union;plot media*modelo;run;

data union;set final12 final13 final14 final15 final16 final17 final18
final19;
proc boxplot data=union;plot media*modelo;run;

data union;set final1 final3 final4 final5 final8 final9 final12
final14 final15 final17 final18 final22;
proc boxplot data=union;plot media*modelo;run;

data final100;set final1;modelo=1;
data final101;set final3;modelo=2;
data final102;set final4;modelo=3;
data final103;set final5;modelo=4;
data final104;set final8;modelo=5;
data final105;set final9;modelo=6;
data final106;set final12;modelo=7;
data final107;set final14;modelo=8;
data final108;set final15;modelo=9;
data final109;set final17;modelo=10;
data final110;set final18;modelo=11;
data final111;set final22;modelo=12;

title 'Regresión lineal';
data union;set final100 final101 final102 final103 final104 final105
final106 final107 final108 final109 final110 final111;
proc boxplot data=union;plot media*modelo;run;
title 'Regresión lineal';
data union;set final106 final107 final108 final109 final110;
 proc boxplot data=union;plot media*modelo;run;
```

ANEXO III: Código R para modelizar

```
# **********************************
# CRUZADAS PARA ENSAMBLADO DEPENDIENTE CONTINUA
# **********************************
# VALIDACIÃ“N CRUZADA REPETIDA Y BOXPLOT para
#
# LOGISTICA
# AVNNET
# RF
# GBM
# XGBM
# SVM
# **********************************
library(plyr)
detach(package:plyr)
library(dummies)
library(MASS)
library(reshape)
library(caret)
library(dplyr)


cruzadaavnnet<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5,
  size=c(5),decay=c(0.01),repeticiones=5,itera=100,trace=FALSE)

 {

  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
```

```
{
 databis<-data[,c(vardep,listconti,listclass)]
 databis<- dummy.data.frame(databis, listclass, sep = ".")
}  else   {
 databis<-data[,c(vardep,listconti)]
}

# c)estandarizar las variables continuas

# Calculo medias y dtipica de datos y estandarizo (solo las continuas)

means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
 number=grupos,repeats=repe,
 savePredictions = "all")

# Aplico caret y construyo modelo

avnnetgrid <-  expand.grid(size=size,decay=decay,bag=FALSE)

avnnet<- train(formu,data=databis,
 method="avNNet",linout = TRUE,maxit=itera,repeats=repeticiones,
 trControl=control,tuneGrid=avnnetgrid,trace=trace)
```

```
  print(avnnet$results)

  preditest<-avnnet$pred[,c("pred","obs","Resample")]

  preditest$prueba<-strsplit(preditest$Resample,"[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba<-NULL

  preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))

return(list(medias,preditest))

 }

cruzadalin<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5)

 {
  library(caret)
  library(dplyr)
  library(dummies)

  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
```

```
} else  {
 databis<-data[,c(vardep,listconti)]
}

# c)estandarizar las variables continuas

# Calculo medias y dtipica de datos y estandarizo (solo las continuas)

means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
 number=grupos,repeats=repe,
 savePredictions = "all")

# Aplico caret y construyo modelo

 lineal<- train(formu,data=databis,
 method="lm",trControl=control)

print(lineal$results)

preditest<-lineal$pred[,c("pred","obs","Resample")]

preditest$prueba<-strsplit(preditest$Resample,"[.]")
preditest$Fold <- sapply(preditest$prueba, "[", 1)
```

```
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba<-NULL

  preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))

return(list(medias,preditest))

 }


cruzadarf<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5,
  nodesize=20,replace=TRUE,ntree=100,mtry=2,sampsize=1)

 {

  if  (sampsize==1)
  {
   sampsize=floor(nrow(data)/(grupos-1))
  }
  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
  }  else   {
   databis<-data[,c(vardep,listconti)]
  }
```

```
# c)estandarizar las variables continuas

# Calculo medias y dtipica de datos y estandarizo (solo las continuas)

means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
 number=grupos,repeats=repe,
 savePredictions = "all")

# Aplico caret y construyo modelo

 rfgrid <-expand.grid(mtry=mtry)

rf<- train(formu,data=databis,
 method="rf",trControl=control,
 tuneGrid=rfgrid,nodesize=nodesize,replace=replace,
 ntree=ntree)

print(rf$results)

preditest<-rf$pred[,c("pred","obs","Resample")]
```

```
   preditest$prueba<-strsplit(preditest$Resample,"[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba<-NULL

  preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))

return(list(medias,preditest))

}

cruzadagbm<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5,
  n.minobsinnode=20,shrinkage=0.1,n.trees=100,interaction.depth=2)

 {
  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
  }  else   {
   databis<-data[,c(vardep,listconti)]
  }

  # c)estandarizar las variables continuas

  # Calculo medias y dtipica de datos y estandarizo (solo las continuas)
```

```
means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
 number=grupos,repeats=repe,
 savePredictions = "all")

# Aplico caret y construyo modelo

# n.minobsinnode=20,shrinkage=0.1,n.trees=100,interaction.depth=2

 gbmgrid <-expand.grid(n.minobsinnode=n.minobsinnode,
  shrinkage=shrinkage,n.trees=n.trees,
  interaction.depth=interaction.depth)

gbm<- train(formu,data=databis,
 method="gbm",trControl=control,
 tuneGrid=gbmgrid,distribution="gaussian",verbose=FALSE)

print(gbm$results)

preditest<-gbm$pred[,c("pred","obs","Resample")]


  preditest$prueba<-strsplit(preditest$Resample,"[.]")
```

```
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba<-NULL

  preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))
return(list(medias,preditest))
#return(list(medias,preditest,gbm,gbmgrid))
 #return(gbm)
}


 cruzadaxgbm<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5,
  min_child_weight=20,eta=0.1,nrounds=100,max_depth=2,
  gamma=0,colsample_bytree=1,subsample=1,alpha=0,lambda=0,lambda_bias=0)
 {

  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
  }  else   {
   databis<-data[,c(vardep,listconti)]
  }

  # c)estandarizar las variables continuas
```

```
# Calculo medias y dtipica de datos y estandarizo (solo las continuas)

means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
 number=grupos,repeats=repe,
 savePredictions = "all")

# Aplico caret y construyo modelo

xgbmgrid <-expand.grid( min_child_weight=min_child_weight,
eta=eta,nrounds=nrounds,max_depth=max_depth,
gamma=gamma,colsample_bytree=colsample_bytree,subsample=subsample)

xgbm<- train(formu,data=databis,
 method="xgbTree",trControl=control,
 tuneGrid=xgbmgrid,objective = "reg:linear",verbose=FALSE,
 alpha=alpha,lambda=lambda,lambda_bias=lambda_bias)

print(xgbm$results)

preditest<-xgbm$pred[,c("pred","obs","Resample")]


  preditest$prueba<-strsplit(preditest$Resample,"[.]")
```

```
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba<-NULL

  preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))

return(list(medias,preditest))

}


 cruzadaSVM<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5, C=1)
 {

  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
  }  else   {
   databis<-data[,c(vardep,listconti)]
  }

  # c)estandarizar las variables continuas

  # Calculo medias y dtipica de datos y estandarizo (solo las continuas)
```

```
means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
 number=grupos,repeats=repe,
 savePredictions = "all")

# Aplico caret y construyo modelo

SVMgrid <-expand.grid(C=C)

SVM<- train(formu,data=databis,
 method="svmLinear",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)

print(SVM$results)

preditest<-SVM$pred[,c("pred","obs","Resample")]



preditest$prueba<-strsplit(preditest$Resample,"[.]")
preditest$Fold <- sapply(preditest$prueba, "[", 1)
preditest$Rep <- sapply(preditest$prueba, "[", 2)
preditest$prueba<-NULL

preditest$error<-(preditest$pred-preditest$obs)^2
```

```
medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))

return(list(medias,preditest))

}


 cruzadaSVMpoly<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5, C=1,degree=2,scale=1)
 {

  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
  }  else   {
   databis<-data[,c(vardep,listconti)]
  }

  # c)estandarizar las variables continuas

  # Calculo medias y dtipica de datos y estandarizo (solo las continuas)

  means <-apply(databis[,listconti],2,mean)
  sds<-sapply(databis[,listconti],sd)

  # Estandarizo solo las continuas y uno con las categoricas
```

```
 datacon<-scale(databis[,listconti], center = means, scale = sds)
 numerocont<-which(colnames(databis)%in%listconti)
 databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

 formu<-formula(paste(vardep,"~.",sep=""))

 # Preparo caret

 set.seed(sinicio)
 control<-trainControl(method = "repeatedcv",
  number=grupos,repeats=repe,
  savePredictions = "all")

 # Aplico caret y construyo modelo

 SVMgrid <-expand.grid(C=C,degree=degree,scale=scale)

 SVM<- train(formu,data=databis,
  method="svmPoly",trControl=control,
  tuneGrid=SVMgrid,verbose=FALSE)

 print(SVM$results)

 preditest<-SVM$pred[,c("pred","obs","Resample")]


 preditest$prueba<-strsplit(preditest$Resample,"[.]")
 preditest$Fold <- sapply(preditest$prueba, "[", 1)
 preditest$Rep <- sapply(preditest$prueba, "[", 2)
 preditest$prueba<-NULL

 preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
 group_by(Rep) %>%
 summarize(error=mean(error))
```

```
return(list(medias,preditest))

}


 cruzadaSVMRBF<-
 function(data=data,vardep="vardep",
  listconti="listconti",listclass="listclass",
  grupos=4,sinicio=1234,repe=5, C=1,sigma=1)
 {

  # PreparaciÃ³n del archivo

  # b)pasar las categÃ³ricas a dummies

  if  (listclass!=c(""))
  {
   databis<-data[,c(vardep,listconti,listclass)]
   databis<- dummy.data.frame(databis, listclass, sep = ".")
  }  else   {
   databis<-data[,c(vardep,listconti)]
  }

  # c)estandarizar las variables continuas

  # Calculo medias y dtipica de datos y estandarizo (solo las continuas)

  means <-apply(databis[,listconti],2,mean)
  sds<-sapply(databis[,listconti],sd)

  # Estandarizo solo las continuas y uno con las categoricas

  datacon<-scale(databis[,listconti], center = means, scale = sds)
  numerocont<-which(colnames(databis)%in%listconti)
  databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

  formu<-formula(paste(vardep,"~.",sep=""))
```

```
  # Preparo caret

  set.seed(sinicio)
  control<-trainControl(method = "repeatedcv",
   number=grupos,repeats=repe,
   savePredictions = "all")

  # Aplico caret y construyo modelo

  SVMgrid <-expand.grid(C=C,sigma=sigma)

  SVM<- train(formu,data=databis,
   method="svmRadial",trControl=control,
   tuneGrid=SVMgrid,verbose=FALSE)

  print(SVM$results)

  preditest<-SVM$pred[,c("pred","obs","Resample")]


  preditest$prueba<-strsplit(preditest$Resample,"[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba<-NULL

  preditest$error<-(preditest$pred-preditest$obs)^2

medias<-preditest %>%
  group_by(Rep) %>%
  summarize(error=mean(error))

return(list(medias,preditest))

}
```

```
library(sas7bdat)
library(dummies)
library(MASS)
library(reshape)
library(caret)
library(dplyr)
library(pROC)
library(readr)
library(haven)
library(stringr)
install.packages('readr', dependencies = TRUE)
install.packages("e1071")

file<-
read.sas7bdat("c:/Users/cris/Documents/TFM/variables_transformadas_train.sas7bdat"
)

load("c:/Users/cris/Documents/TFM/svml.RData")
load("c:/Users/cris/Documents/TFM/randomforest.RData")
#Pasamos a factor las variables categÃ³ricas
file$album_type<-as.factor(file$album_type)
file$explicit_track<-as.factor(file$explicit_track)

vardep<-"popularity_track"
conti<- c("EXP_key_min_track", "LOG_speechiness_track",
"LOG_total_followers_artist", "LOG_valence_track",
      "PWR_loudness_track", "PWR_time_signature_max_track",
"SQRT_instrumentalness_track",
      "SQRT_total_tracks_album","acousticness_track", "danceability_track",
"duration_ms_track",
      "energy_track","instrumentalness_track","key_max_track", "key_min_track",
"loudness_min_track",
      "mode_track", "speechiness_track","time_signature_max_track",
"time_signature_min_track",
      "time_signature_track", "total_followers_artist", "track_number_track",
"valence_track")
categor<- c("album_type", "explicit_track")
grupos<-4
sinicio<-12345
repe<-5
```

```
#REDES NEURONALES
listamodelos_rn<-NULL
listamodelos2_rn<-NULL
listam_rn<-NULL
tabla_modelos_rn<-data.frame()

i<-1
size_<-c(5,10,15,20)
decay_<-c(0.001,0.01,0.1)
for (s in size_){
  for (d in decay_){
    modelo<-cruzadaavnnet(data=file,vardep=vardep,listconti=conti,
                          listclass=categor,
                          grupos=grupos,sinicio=sinicio,repe=repe,
                            size=s,decay=d,repeticiones=10,itera=100)
    listamodelos_rn<-paste(c(listamodelos_rn,paste0("modelo_rn",i)), collapse =",")
    listamodelos2_rn<-paste(c(listamodelos2_rn,paste0("modelo_rn",i,"bis")),   collapse
=",")
    assign(paste0("modelo_rn",i,"bis"),modelo)
    listam_rn<-c(listam_rn,paste0("modelo",i))
    eval(parse(text=paste0("modelo_rn", i, "bis <- as.data.frame(modelo[1])")))
    eval(parse(text=paste0("modelo_rn", i, "bis$modelo <- i")))
    eval(parse(text=paste0("tabla_modelos_rn[",i,",1] <- s")))
    eval(parse(text=paste0("tabla_modelos_rn[",i,",2] <- d")))
    i<-i+1
    print(i)
    modelo<-NULL
  }
}

listam_rn
listamodelos2_rn
tabla_modelos_rn
row.names(tabla_modelos_rn)<-listam_rn
names(tabla_modelos_rn)<-c('Nodos', 'Decay')
tabla_modelos_rn
eval(parse(text=paste0("union_rn <- rbind(",listamodelos2_rn,")")))
eval(parse(text=paste0("union_rn <- rbind(",listamodelos2_rn,")")))
```

```
par(cex.axis=0.5)
boxplot(data=union_rn,error~modelo,main="MSE Redes neuronales")
union_rn$modelo <- with(union_rn,
                        reorder(modelo,error, mean))
par(cex.axis=0.5)
boxplot(data=union_rn3,error~modelo,main="MSE Redes neuronales")
#BAGGING
# La funciÃƒÂ³n cruzadarfbin permite plantear bagging
# (para bagging hay que poner mtry=numero de variables independientes)
listamodelos<-NULL
listamodelos2<-NULL
listam<-NULL
tabla_modelos<-data.frame()
i<-1
nodos <- c(10,20,30,40)
num <- c(100,200,500,1000)
for (nod in nodos){
  for (n in num){
    #i<-i+1
    print("entro")
    modelo<-cruzadarf(data=file,vardep=vardep,listconti=conti,
                      listclass=categor,
                      grupos=grupos,sinicio=sinicio,repe=repe,nodesize=nod,
                      mtry=26,ntree=n,replace=TRUE,sampsize=350)

    listamodelos<-paste(c(listamodelos,paste0("modelobagging",i)), collapse =",")
    listamodelos2<-paste(c(listamodelos2,paste0("modelobagging",i,"bis")),      collapse
=",")
    assign(paste0("modelobagging",i,"bis"),modelo)
    listam<-c(listam,paste0("modelo",i))
    #eval(parse(text=paste0("modelobagging", i, "$modelo <- i")))
    eval(parse(text=paste0("modelobagging", i, "bis$modelo <- i")))
    eval(parse(text=paste0("modelobagging", i, "bis <- as.data.frame(modelo[1])")))
    #medias3bis<-as.data.frame(medias3[1])
    #medias3bis$modelo<-"rf"
    eval(parse(text=paste0("tabla_modelos[",i,",1] <- nod")))
    eval(parse(text=paste0("tabla_modelos[",i,",2] <- n")))
    i<-i+1
```

```
    print(i)
  }
}

listam
tabla_modelos
row.names(tabla_modelos)<-listam
names(tabla_modelos)<-c('Num obs','Num trees ')
tabla_modelos

i<-1
lista_modelos<-
c("modelobagging1bis","modelobagging2bis","modelobagging3bis","modelobagging4bi
s","modelobagging5bis","modelobagging6bis","modelobagging7bis","modelobagging8bi
s","modelobagging9bis","modelobagging10bis","modelobagging11bis","modelobagging
12bis","modelobagging13bis","modelobagging14bis","modelobagging15bis","modeloba
gging16bis")
for (k in lista_modelos){
  #eval(parse(text=paste0("modelo", i, "$modelo <- k")))
  parametros<-paste0(tabla_modelos[i,1],"_",tabla_modelos[i,2])
  print(parametros)
  eval(parse(text=paste0(k,"$modelo <- parametros")))
  print(k)
  print(i)
  i<-i+1
}
listamodelos2
eval(parse(text=paste0("unionbag <- rbind(",listamodelos2,")")))
par(cex.axis=0.5)
unionbag$modelo <- with(unionbag,
                        reorder(modelo,error, mean))
boxplot(data=unionbag,error~modelo,main="MSE Bagging")
####^Random forest

#RANDOM FOREST
listamodelos_r<-c("")
listamodelos_r<-NULL
listamodelos2_r<-NULL
listam_r<-NULL
```

```
tabla_modelos_r<-data.frame()
i<-1
nodos <- c(10,20,30,40)
num_var<-c(3,7,13,26)
num <- c(100,200,500,1000)
for (nod in nodos){
  for (n in num){
    for(v in num_var){
      modelo<-cruzadarf(data=file,vardep=vardep,listconti=conti,
                        listclass=categor,
                        grupos=grupos,sinicio=sinicio,repe=repe,nodesize=nod,
                          mtry=v,ntree=n,replace=TRUE)
      listamodelos_r<-paste(c(listamodelos_r,paste0("modelo_rf",i)), collapse =",")
      listamodelos2_r<-paste(c(listamodelos2_r,paste0("modelobagging",i,"bis")),
collapse =",")
      assign(paste0("modelo_rf",i,"bis"),modelo)
      listam_r<-c(listam_r,paste0("modelo",i))
      eval(parse(text=paste0("modelo_rf", i, "bis$modelo <- i")))
      eval(parse(text=paste0("modelo_rf", i, "bis <- as.data.frame(modelo[1])")))
      eval(parse(text=paste0("tabla_modelos_r[",i,",1] <- nod")))
      eval(parse(text=paste0("tabla_modelos_r[",i,",2] <- n")))
      eval(parse(text=paste0("tabla_modelos_r[",i,",3] <- v")))
      print(i)
      i<-i+1
    }
  }
}

listam_r
tabla_modelos_r
listamodelos_r
row.names(tabla_modelos_r)<-listam_r
names(tabla_modelos_r)<-c('Num obs','Num trees ','Num variables')
tabla_modelos_r
eval(parse(text=paste0("union_rf <- rbind(",listamodelos2_r,")")))
listamodelos2_r2
#Gradient Boosting
listamodelos2_gb<-NULL
```

```
listamodelos_gb<-NULL
listam_gb<-NULL
tabla_modelos_gb<-data.frame()
i<-1
n_min <- c(10,100,500,1000)
num <- c(100,200,500,1000)
sk<-c(0.001,0.05,0.1,0.3,0.05)

for (m in n_min){
  for (s in sk){
    for (n in num){
      print("Pensando...")
      modelo<-cruzadagbm(data=file,vardep=vardep,listconti=conti,
                         listclass=categor,
                         grupos=grupos,sinicio=sinicio,repe=repe,
                         n.minobsinnode=m,shrinkage=sk,n.trees=num,interaction.depth=2)
      listamodelos_gb<-paste(c(listamodelos_gb,paste0("modelo_gb",i)), collapse =",")
      listamodelos2_gb<-paste(c(listamodelos2_gb,paste0("modelo_gb",i,"bis")),
collapse =",")
      assign(paste0("modelo_gb",i,"bis"),modelo)
      listam_gb<-c(listam_gb,paste0("modelo",i))
      #eval(parse(text=paste0("modelo_gb", i, "bis <- as.data.frame(modelo[1])")))
      eval(parse(text=paste0("modelo_gb", i, "bis$modelo <- i")))
      eval(parse(text=paste0("tabla_modelos_gb[",i,",1] <- m")))
      eval(parse(text=paste0("tabla_modelos_gb[",i,",2] <- s")))
      eval(parse(text=paste0("tabla_modelos_gb[",i,",3] <- n")))
      i<-i+1
      print(i)
      modelo<-NULL

    }
  }
}
```

```
listam_gb
tabla_modelos_gb
row.names(tabla_modelos_gb)<-listam_gb
names(tabla_modelos_gb)<-c('Num_min_size','shrinkage','Nodos')
tabla_modelos_gb
eval(parse(text=paste0("union_gb <- rbind(",listamodelos2_gb,")")))
par(cex.axis=0.5)
boxplot(data=union_gb,error~modelo,main="MSE GBM")

#XGradient Boosting
listamodelos2_xgb<-NULL
listamodelos_xgb<-NULL
listam_xgb<-NULL
tabla_modelos_xgb<-data.frame()
i<-1
child <-c(5,10,20)
ETA <-c(0.1,0.05,0.03,0.01,0.001)
rondas <- c(100,500,1000,5000)
profundidad<-c(2,4,6)
for (ch in child){
  for (e in ETA){
    for (ron in rondas){
      for (pro in profundidad){
        modelo<-cruzadaxgbmbin(data=file,vardep=vardep,listconti=conti,
                               listclass=categor,
                               grupos=grupos,sinicio=sinicio,repe=repe,
                               min_child_weight=ch,eta=e,nrounds=ron,max_depth=pro,
                               gamma=0,colsample_bytree=1,subsample=1,
                               alpha=0,lambda=0,lambda_bias=0)
        listamodelos_xgb<-paste(c(listamodelos_xgb,paste0("modelo_xgb",i)),   collapse
=",")
        listamodelos2_xgb<-paste(c(listamodelos2_xgb,paste0("modelo_xgb",i,"bis")),
collapse =",")
        assign(paste0("modelo_xgb",i,"bis"),modelo)
        listam_gb<-c(listam_gb,paste0("modelo",i))
```

```
        eval(parse(text=paste0("modelo_xgb", i, "bis <- as.data.frame(modelo[1])")))
        eval(parse(text=paste0("modelo_xgb", i, "bis$modelo <- i")))
        eval(parse(text=paste0("tabla_modelos_xgb[",i,",1] <- ch")))
        eval(parse(text=paste0("tabla_modelos_xgb[",i,",2] <- e")))
        eval(parse(text=paste0("tabla_modelos_xgb[",i,",3] <- ron")))
        eval(parse(text=paste0("tabla_modelos_xgb[",i,",4] <- pro")))
        i<-i+1
        print(i)
        modelo<-NULL

      }
     }
    }
  }
  listam_xgb
  tabla_modelos_xgb
  row.names(tabla_modelos_xgb)<-listam_xgb
  names(tabla_modelos_xgb)<-c('child','ETA','rondas','profundidad')
  tabla_modelos_xgb
  eval(parse(text=paste0("union_xgb <- rbind(",listamodelos2_xgb,")")))
  par(cex.axis=0.5)
  boxplot(data=union_xgb,error~modelo,main="MSE XGBM")

#SVM
#LINEAL
listamodelos_svml<-NULL
listamodelos2_svml<-NULL
listam_svml<-NULL
tabla_modelos_svml<-data.frame()

i<-1
param_c<-c(0.01,0.1,1,10)
for (p in param_c){
  modelo<-cruzadaSVM(data=file,vardep=vardep,listconti=conti,
```

```
                    listclass=categor,
                    grupos=grupos,sinicio=sinicio,repe=repe,C=p)
  listamodelos_svml<-paste(c(listamodelos_svml,paste0("modelo_svml",i)),       collapse
=",")
  listamodelos2_svml<-paste(c(listamodelos2_svml,paste0("modelo_svml",i,"bis")),
collapse =",")
  assign(paste0("modelo_svml",i,"bis"),modelo)
  listam_svml<-c(listam_svml,paste0("modelo",i))
  eval(parse(text=paste0("modelo_svml", i, "bis <- as.data.frame(modelo[1])")))
  eval(parse(text=paste0("modelo_svml", i, "bis$modelo <- i")))
  eval(parse(text=paste0("tabla_modelos_svml[",i,",1] <- p")))
  print(i)
  i<-i+1
  modelo<-NULL
}

listam_svml
tabla_modelos_svml
row.names(tabla_modelos_svml)<-listam_svml
names(tabla_modelos_svml)<-c('Parametro C')
tabla_modelos_svml
eval(parse(text=paste0("union_svml <- rbind(",listamodelos2_svml,")")))

union_svml$modelo <- with(union_svml,
                      reorder(modelo,error, mean))
par(cex.axis=0.8)
boxplot(data=union_svml,error~modelo,main="MSE SVML")

  svmr<-NULL
  listamodelos_svmr<-NULL
  listam_svmr<-NULL
  tabla_modelos_svmr<-data.frame()



  i<-1
```

```
param_c2<-c(0.01,0.1,1,10)
sigma<-c(0.01,0.1,1,10)
for (p in param_c2){
  for (s in sigma){
    modelo<-cruzadaSVMRBF(data=file,vardep=vardep,listconti=conti,
                          listclass=categor,
                          grupos=grupos,sinicio=sinicio,repe=repe,C=p,sigma=s)

    listamodelos_svmr<-paste(c(listamodelos_svmr,paste0("modelo_svmr",i)), collapse
=",")
    listamodelos2_svmr<-paste(c(listamodelos2_svmr,paste0("modelo_svmr",i,"bis")),
collapse =",")
    assign(paste0("modelo_svmr",i,"bis"),modelo)
    listam_svmr<-c(listam_svmr,paste0("modelo",i))
    eval(parse(text=paste0("modelo_svmr", i, "bis <- as.data.frame(modelo[1])")))
    eval(parse(text=paste0("modelo_svmr", i, "bis$modelo <- i")))
    eval(parse(text=paste0("tabla_modelos_svmr[",i,",1] <- p")))
    eval(parse(text=paste0("tabla_modelos_svmr[",i,",2] <- s")))
    i<-i+1
    print(i)

  }

}
listamodelos_svmr
listam_svmr
tabla_modelos_svmr
row.names(tabla_modelos_svmr)<-listam_svmr
names(tabla_modelos_svmr)<-c('Parametro C','Scale')
tabla_modelos_svmr
eval(parse(text=paste0("union_svmr <- rbind(",listamodelos2_svmr,")")))
par(cex.axis=0.5)
boxplot(data=union_svmr,error~modelo,main="MSE SVMR")
```

```
#ENSAMBLADO
listconti<- conti
listclass<-categor
grupos<-4
sinicio<-12345
repe<-5
#ntree<-100
# APLICACIÃ“N CRUZADAS PARA ENSAMBLAR

medias1<-cruzadalin(data=file,
                    vardep=vardep,listconti=listconti,
                    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe)



medias1bis<-as.data.frame(medias1[1])
medias1bis$modelo<-"reg"
predi1<-as.data.frame(medias1[2])
predi1$reg<-predi1$pred

medias2<-cruzadaavnnet(data=file,
                       vardep=vardep,listconti=listconti,
                       listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
                       size=c(10),decay=c(0.01),repeticiones=10,itera=100)

medias2bis<-as.data.frame(medias2[1])
medias2bis$modelo<-"avnnet"
predi2<-as.data.frame(medias2[2])
predi2$avnnet<-predi2$pred

medias3<-cruzadarf(data=file,
                   vardep=vardep,listconti=listconti,
                   listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
                   mtry=13,ntree=1000,nodesize=20,replace=TRUE)
```

```
medias3bis<-as.data.frame(medias3[1])
medias3bis$modelo<-"rf"
predi3<-as.data.frame(medias3[2])
predi3$rf<-predi3$pred


medias4<-cruzadagbm(data=file,
                    vardep=vardep,listconti=listconti,
                    listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
                    n.minobsinnode=10,shrinkage=0.001,n.trees=1000,interaction.depth=2)

medias4bis<-as.data.frame(medias4[1])
medias4bis$modelo<-"gbm"
predi4<-as.data.frame(medias4[2])
predi4$gbm<-predi4$pred

medias5<-cruzadaxgbm(data=file,
                     vardep=vardep,listconti=listconti,
                     listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
                     min_child_weight=5,eta=0.01,nrounds=1000,max_depth=6,
                     gamma=0,colsample_bytree=1,subsample=1,
                     alpha=0,lambda=0,lambda_bias=0)

medias5bis<-as.data.frame(medias5[1])
medias5bis$modelo<-"xgbm"
predi5<-as.data.frame(medias5[2])
predi5$xgbm<-predi5$pred


medias6<-cruzadaSVM(data=file,
                    vardep=vardep,listconti=listconti,
                    listclass=listclass,grupos=grupos,
```

```
                    sinicio=sinicio,repe=repe,C=0.1)

  medias6bis<-as.data.frame(medias6[1])
  medias6bis$modelo<-"svmLinear"
  predi6<-as.data.frame(medias6[2])
  predi6$svmLinear<-predi6$pred



  medias7<-cruzadaSVMRBF(data=file,
                      vardep=vardep,listconti=listconti,
                      listclass=listclass,grupos=grupos,
                      sinicio=sinicio,repe=repe,
                      C=10,sigma=0.01)

  medias7bis<-as.data.frame(medias7[1])
  medias7bis$modelo<-"svmRadial"
  predi7<-as.data.frame(medias7[2])
  predi7$svmRadial<-predi7$pred



  union1<-rbind(medias1bis,medias2bis,
              medias3bis,medias4bis,medias5bis,medias6bis,
              medias7bis)
  union1<-rbind(medias3bis,medias5bis)
  par(cex.axis=0.5)
  boxplot(data=union1,error~modelo)



  # CONSTRUCCIÃ“N DE TODOS LOS ENSAMBLADOS
  # SE UTILIZARÃ  N LOS ARCHIVOS SURGIDOS DE LAS FUNCIONES LLAMADOS
predi1,...

  unipredi<-cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7)
```

```
# Esto es para eliminar columnas duplicadas
unipredi<- unipredi[, !duplicated(colnames(unipredi))]

# Construccion de ensamblados, cambiar al gusto

unipredi$regavnet<-(unipredi$reg+unipredi$avnnet)/2
unipredi$regrf<-(unipredi$reg+unipredi$rf)/2
unipredi$reggbm<-(unipredi$reg+unipredi$gbm)/2
unipredi$regxbm<-(unipredi$reg+unipredi$xgbm)/2
unipredi$regsvml<-(unipredi$reg+unipredi$svmLinear)/2
unipredi$regsvmr<-(unipredi$reg+unipredi$svmRadial)/2
unipredi$avnnetrf<-(unipredi$avnnet+unipredi$rf)/2
unipredi$avnnetgbm<-(unipredi$avnnet+unipredi$gbm)/2
unipredi$avnnetxgbm<-(unipredi$avnnet+unipredi$xgbm)/2
unipredi$avnnetsvml<-(unipredi$avnnet+unipredi$svmLinear)/2
unipredi$avnnetsvmr<-(unipredi$avnnet+unipredi$svmRadial)/2
unipredi$rfgbm<-(unipredi$rf+unipredi$gbm)/2
unipredi$rfxgbm<-(unipredi$rf+unipredi$xgbm)/2
unipredi$rfsvml<-(unipredi$rf+unipredi$svmLinear)/2
unipredi$rfsvmr<-(unipredi$rf+unipredi$svmRadial)/2
unipredi$gbmxgbm<-(unipredi$gbm+unipredi$xgbm)/2
unipredi$gbmsvml<-(unipredi$gbm+unipredi$svmLinear)/2
unipredi$gbmsvmr<-(unipredi$gbm+unipredi$svmRadial)/2
unipredi$xgbmsvml<-(unipredi$xgbm+unipredi$svmLinear)/2
unipredi$xgbmsvmr<-(unipredi$xgbm+unipredi$svmRadial)/2
unipredi$regavnnetrf<-(unipredi$reg+unipredi$avnnet+unipredi$rf)/3
unipredi$regavnnetgbm<-(unipredi$reg+unipredi$avnnet+unipredi$gbm)/3
unipredi$regavnnetxgbm<-(unipredi$reg+unipredi$avnnet+unipredi$xgbm)/3
unipredi$regavnnetsvml<-(unipredi$reg+unipredi$avnnet+unipredi$svmLinear)/3
unipredi$regavnnetsvmr<-(unipredi$reg+unipredi$avnnet+unipredi$svmRadial)/3
unipredi$regrfgbm<-(unipredi$reg+unipredi$rf+unipredi$gbm)/3
unipredi$regrfxgbm<-(unipredi$reg+unipredi$rf+unipredi$xgbm)/3
unipredi$regrfsvml<-(unipredi$reg+unipredi$rf+unipredi$svmLinear)/3
```

```
unipredi$regrfsvmr<-(unipredi$reg+unipredi$rf+unipredi$svmRadial)/3
unipredi$reggbmxgbm<-(unipredi$reg+unipredi$gbm+unipredi$xgbm)/3
unipredi$reggbmxgbm<-(unipredi$reg+unipredi$gbm+unipredi$xgbm)/3
unipredi$reggbmsvml<-(unipredi$reg+unipredi$gbm+unipredi$svmLinear)/3
unipredi$reggbmsvmr<-(unipredi$reg+unipredi$gbm+unipredi$svmRadial)/3
unipredi$regxgbmsvml<-(unipredi$reg+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$regxgbmsvmr<-(unipredi$reg+unipredi$xgbm+unipredi$svmRadial)/3
unipredi$rfgbmsvml<-(unipredi$rf+unipredi$gbm+unipredi$svmLinear)/3
unipredi$rfgbmsvmr<-(unipredi$rf+unipredi$gbm+unipredi$svmRadial)/3
unipredi$rfxgbmsvml<-(unipredi$rf+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$rfxgbmsvmr<-(unipredi$rf+unipredi$xgbm+unipredi$svmRadial)/3
unipredi$rfavnnetgbm<-(unipredi$rf+unipredi$avnnet+unipredi$gbm)/3
unipredi$rfavnnetxgbm<-(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3
unipredi$rfavnnetsvml<-(unipredi$rf+unipredi$avnnet+unipredi$svmLinear)/3
unipredi$rfavnnetsvmr<-(unipredi$rf+unipredi$avnnet+unipredi$svmRadial)/3
unipredi$avnnetgbmsvml<-(unipredi$avnnet+unipredi$gbm+unipredi$svmLinear)/3
unipredi$avnnetgbmsvmr<-(unipredi$avnnet+unipredi$gbm+unipredi$svmRadial)/3
unipredi$regrfgbmavnnet<-
(unipredi$reg+unipredi$rf+unipredi$gbm+unipredi$avnnet)/4
unipredi$regrfxgbmavnnet<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet)/4
unipredi$regrfxgbmavnnetsvml<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmLinear)/5
unipredi$regrfxgbmavnnetsvmr<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmRadial)/5
unipredi$regrfgbmavnnetsvml<-
(unipredi$reg+unipredi$rf+unipredi$gbm+unipredi$avnnet+unipredi$svmLinear)/5
unipredi$regrfgbmavnnetsvmr<-
(unipredi$reg+unipredi$rf+unipredi$gbm+unipredi$avnnet+unipredi$svmRadial)/5

dput(names(unipredi))

# Recorto los modelos de la lista de variables

listado<-c("reg", "avnnet",
```

```
          "rf", "gbm", "xgbm", "svmLinear",
"svmRadial","regavnet","regrf","reggbm","regxbm","regsvml","regsvmr","avnnetrf","avnn
etgbm","avnnetxgbm","avnnetsvml","avnnetsvmr","rfgbm","rfxgbm","rfsvml","rfsvmr",

"gbmxgbm","gbmsvml","gbmsvmr","xgbmsvml","xgbmsvmr","regavnnetrf","regavnnetgb
m","regavnnetxgbm","regavnnetsvml","regavnnetsvmr","regrfgbm","regrfxgbm","regrfsv
ml","regrfsvmr"
,"reggbmxgbm","reggbmxgbm","reggbmsvml","reggbmsvmr","regxgbmsvml","regxgbms
vmr","rfgbmsvml","rfgbmsvmr","rfxgbmsvml","rfxgbmsvmr","rfavnnetgbm","rfavnnetxgb
m","rfavnnetsvml","rfavnnetsvmr","avnnetgbmsvml","avnnetgbmsvmr","regrfgbmavnnet
","regrfxgbmavnnet","regrfxgbmavnnetsvml","regrfxgbmavnnetsvmr","regrfgbmavnnets
vml","regrfgbmavnnetsvmr"


  )




  repeticiones<-nlevels(factor(unipredi$Rep))
  unipredi$Rep<-as.factor(unipredi$Rep)
  unipredi$Rep<-as.numeric(unipredi$Rep)

  # Calculo el MSE para cada repeticion de validaciÃ³n cruzada

  medias0<-data.frame(c())
  i<-1
  for (prediccion in listado)
  {
    paso <-unipredi[,c("obs",prediccion,"Rep")]
    paso$error<-(paso[,c(prediccion)]-paso$obs)^2
    paso<-paso %>%
      group_by(Rep) %>%
      summarize(error=mean(error))
    paso$modelo<-prediccion
    medias0<-rbind(medias0,paso)
```

```
  print(i)
  i<-i+1
}
# Finalmente boxplot

par(cex.axis=0.5,las=2)
boxplot(data=medias0,outcex=0.3,error~modelo)

# PRESENTACION TABLA MEDIAS

tablamedias<-medias0 %>%
  group_by(modelo) %>%
  summarize(error=mean(error))

tablamedias<-tablamedias[order(tablamedias$error),]

# ORDENACIÃ“N DEL FACTOR MODELO POR LAS MEDIAS EN ERROR
# PARA EL GRÃ  FICO

medias0$modelo <- with(medias0,
                       reorder(modelo,error, mean))
par(cex.axis=0.5,las=2)
boxplot(data=medias0,error~modelo,col="pink")


# Se pueden escoger listas pero el factor hay que pasarlo a character
# para que no salgan en el boxplot todos los niveles del factor

listadobis<-c("regresion", "avnnet",
              "rf","gbm",  "xgbm", "svmLinear",  "svmPoly",
              "svmRadial","rfxg")

listadobis2<-c("rfxgbm","rf" ,"xgbm","rfavnnetxgbm")
```

```
medias0$modelo<-as.character(medias0$modelo)

mediasver<-medias0[medias0$modelo %in% listadobis2,]

mediasver$modelo <- with(mediasver,
                         reorder(modelo,error, mean))

par(cex.axis=0.8,las=2)
boxplot(data=mediasver,error~modelo,col="pink")

#FUNCION R2
v2<-var(rfxgbm$obs)
mse2<-sum((rfxgbm$pred - rfxgbm$obs)^2) /nrow(rfxgbm)
R2_<-1-(mse2/v2)

#TABLA IMPORTANCIA DE LAS VARIABLES DEL MODELO GANADOR RANDOM
FOREST
if  (listclass!=c(""))
{
  databis<-data[,c(vardep,listconti,listclass)]
  databis<- dummy.data.frame(databis, listclass, sep = ".")
}  else   {
  databis<-data[,c(vardep,listconti)]
}



# Calculo medias y dtipica de datos y estandarizo (solo las continuas)

means <-apply(databis[,listconti],2,mean)
sds<-sapply(databis[,listconti],sd)

# Estandarizo solo las continuas y uno con las categoricas

datacon<-scale(databis[,listconti], center = means, scale = sds)
```

```
numerocont<-which(colnames(databis)%in%listconti)
databis<-cbind(datacon,databis[,-numerocont,drop=FALSE ])

formu<-formula(paste(vardep,"~.",sep=""))

# Preparo caret

set.seed(sinicio)
control<-trainControl(method = "repeatedcv",
                      number=grupos,repeats=repe,
                      savePredictions = "all")

# Aplico caret y construyo modelo

rfgrid <-expand.grid(mtry=13)

rf<- train(formu,data=databis,
           method="rf",trControl=control,
           tuneGrid=rfgrid,nodesize=20,replace=TRUE,
           ntree=1000)



vars <-varImp(rf,scale=TRUE, total=30)
df<-as.data.frame(vars)
print(vars)
hola<-nrow(varImp(rf,scale=TRUE)$importance)
print(hola)
print(varImp(rf,scale=TRUE)$importance)
varImp(rf)$importance %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  arrange(Overall) %>%
  mutate(rowname = forcats::fct_inorder(rowname )) %>%
  ggplot()+
```

```
geom_col(aes(x = rowname, y = Overall))+
coord_flip()+
theme_bw()
```

ANEXO IV: Gráficos Capítulo 5

Debido a que algunos gráficos de los modelos calculados en el Capítulo 5 tenían muchas combinaciones. En este Anexo vamos a incluir todos los modelos antes de elegir los 20 mejores como se hizo en ese capítulo.

IV.1. Random Forest

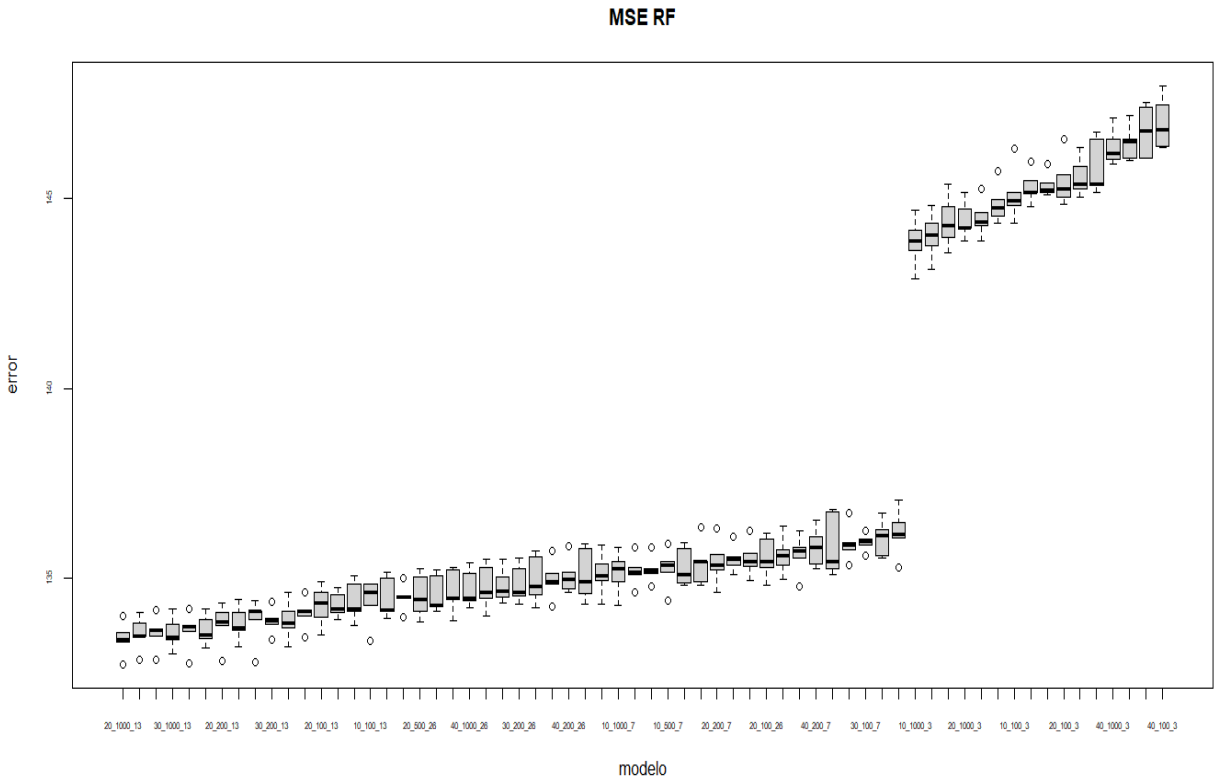

Figura IV.1: Fráfico con todos los modelos de Random Forest. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

IV.2. GRADIENT BOOSTING

Se iba a representar en un mismo gráfico las 100 combinaciones de modelos. Pero hemos obtenido este resultado:

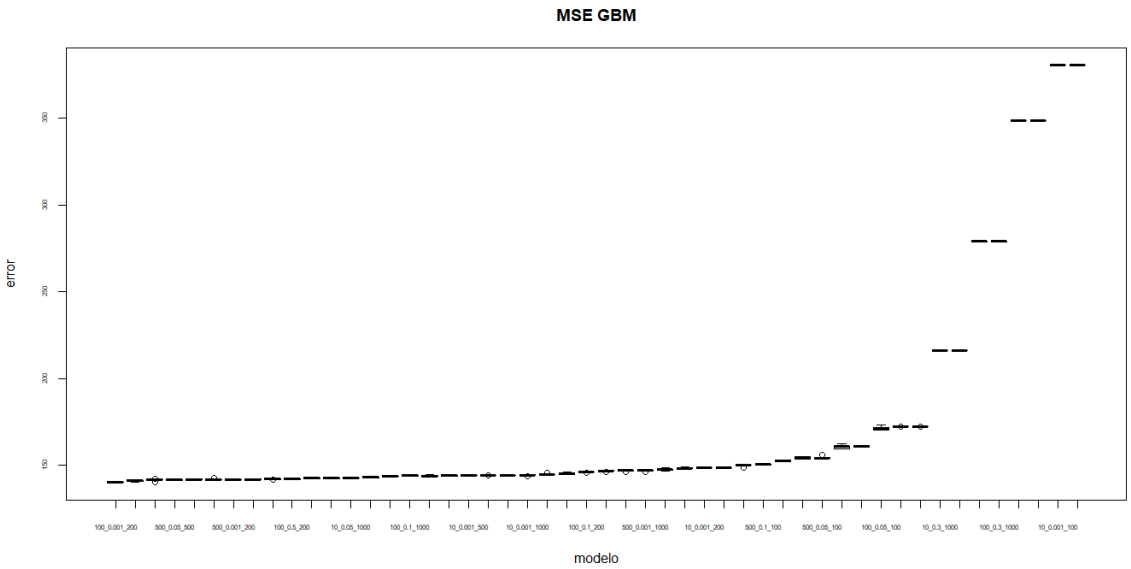

Figura IV.2: Gráfico con todos los modelos de Gradient Boosting. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

Como no se pueden ver con claridad los modelos, vamos hacer 4 subgrupos de 25 modelos:

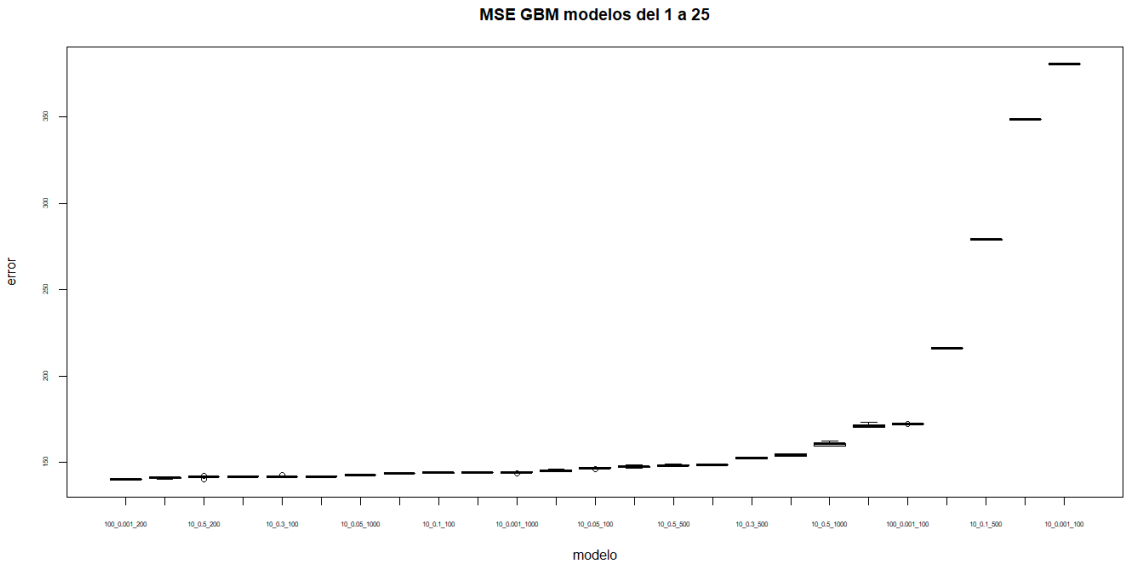

Figura IV.3: Gráfico con los modelos de 1 a 25 de Gradient Boosting. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

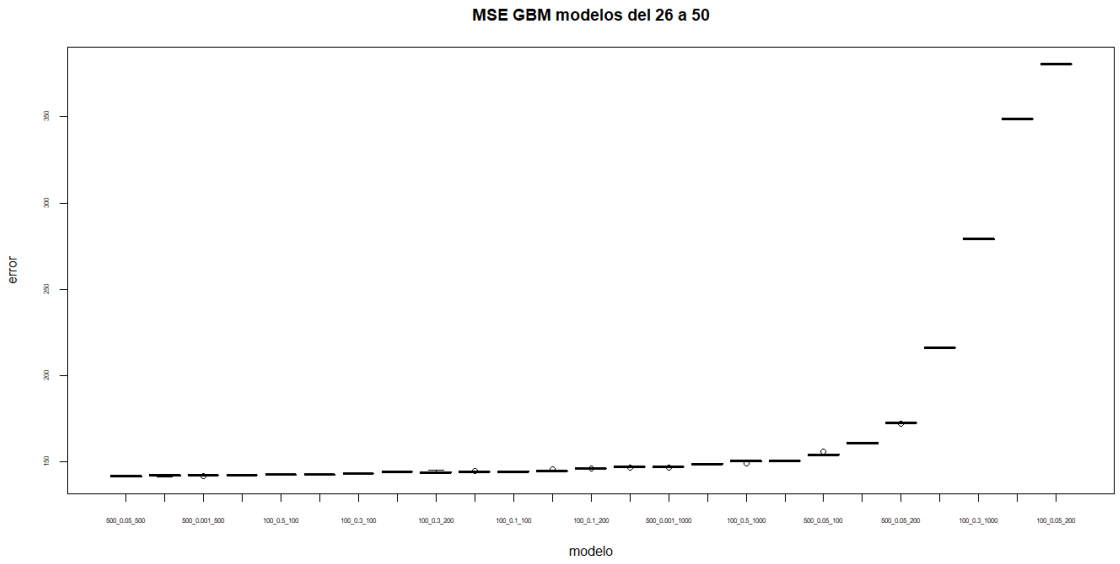

Figura IV.4: Gráfico con los modelos de 26 a 50 de Gradient Boosting. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

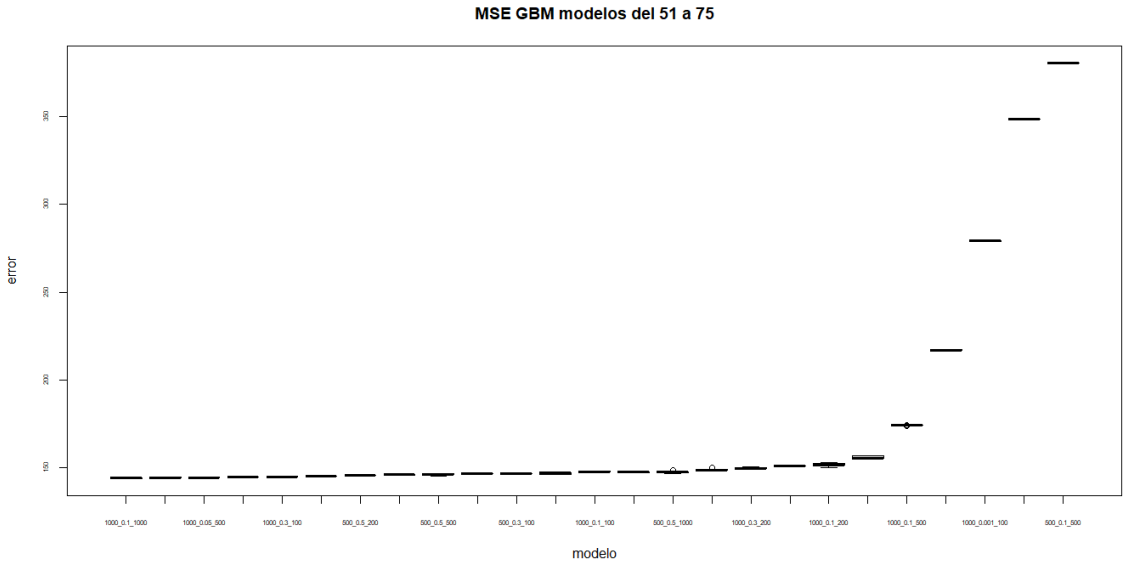

Figura IV.5: Gráfico con los modelos de 51 a 75 de Gradient Boosting. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

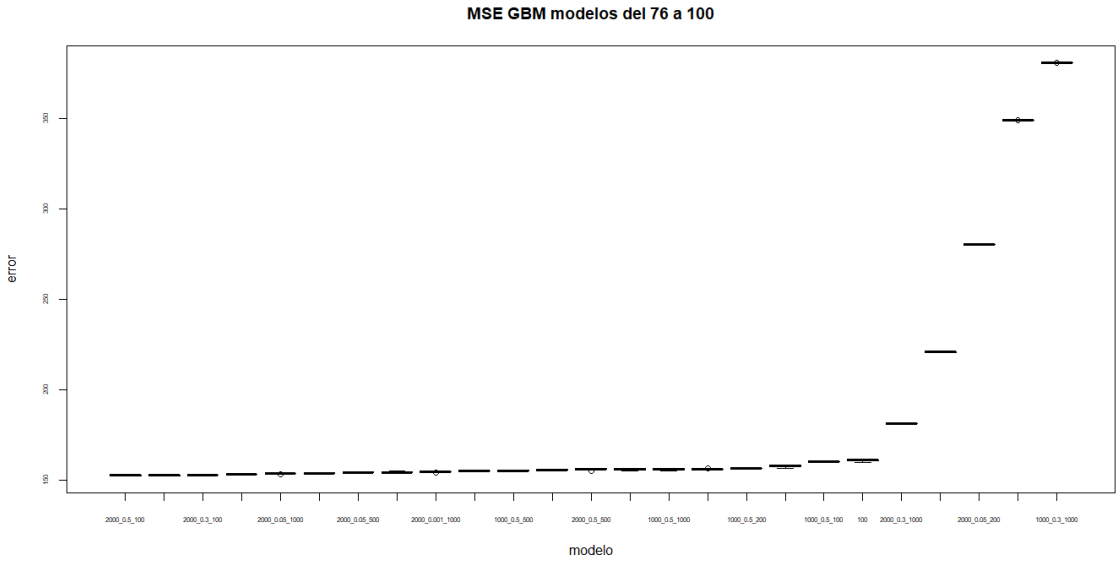

Figura IV.6: Gráfico con los modelos de 76 a 100 de Gradient Boosting. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

IV.3. XGBOOST

Se iba a representar en un mismo gráfico las 180 combinaciones de modelos, como se ha hecho en este mismo anexo con Random Forest. Pero hemos obtenido este resultado:

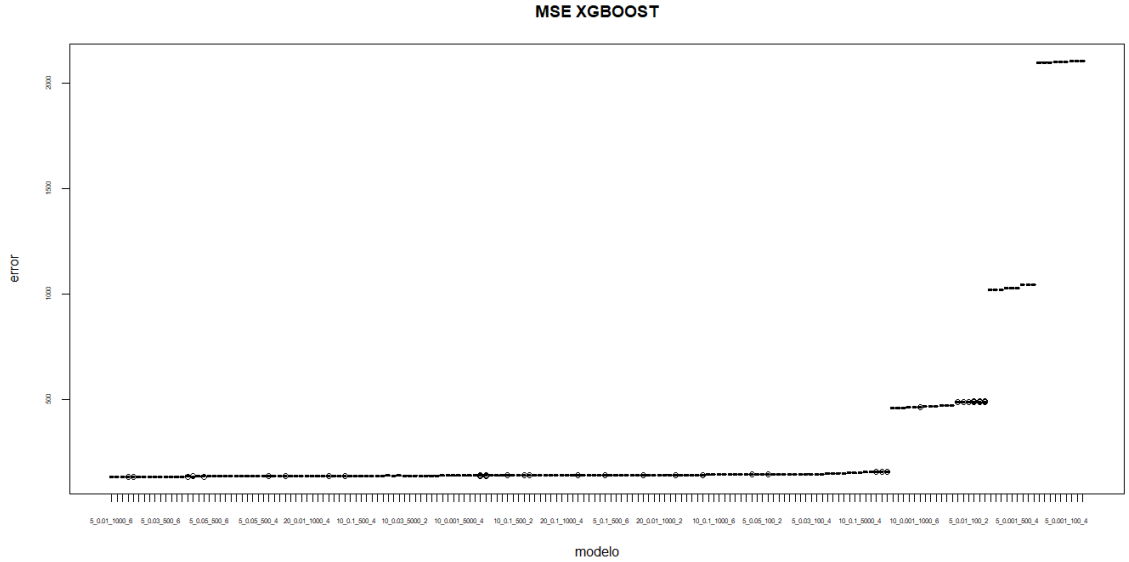

Figura IV.7: Gráfico con los modelos e Xgboost. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

No se distinguen los modelos, por lo que se va a dividir en cinco grupos de 36 modelos para representarlo. Las siguientes gráficas recogen todos los modelos divididos en grupos.

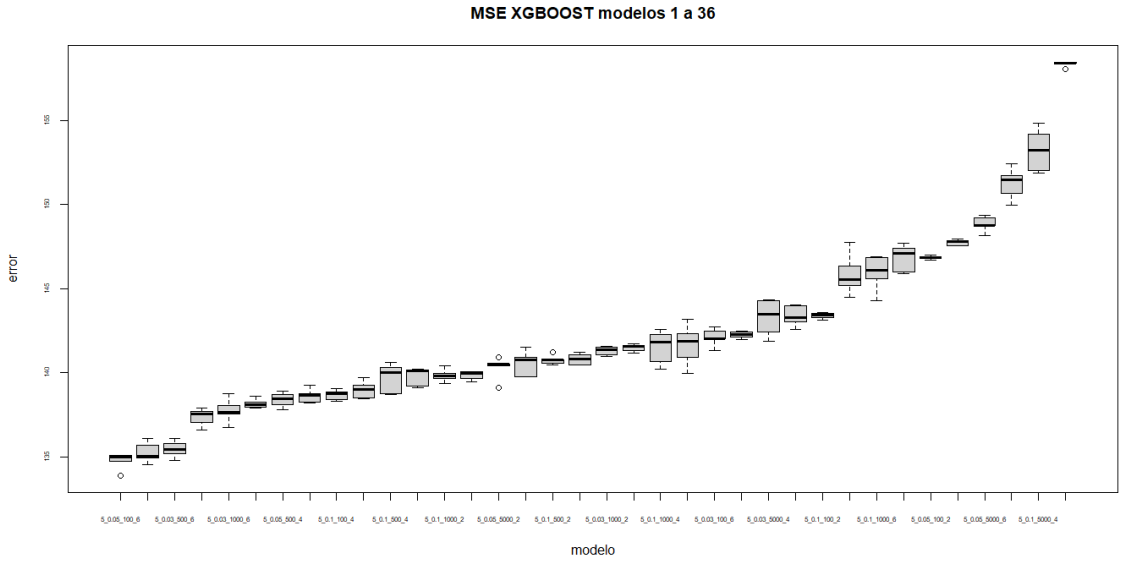

Figura IV.8: Gráfico con los modelos de 1 a 36 de Xgboost. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

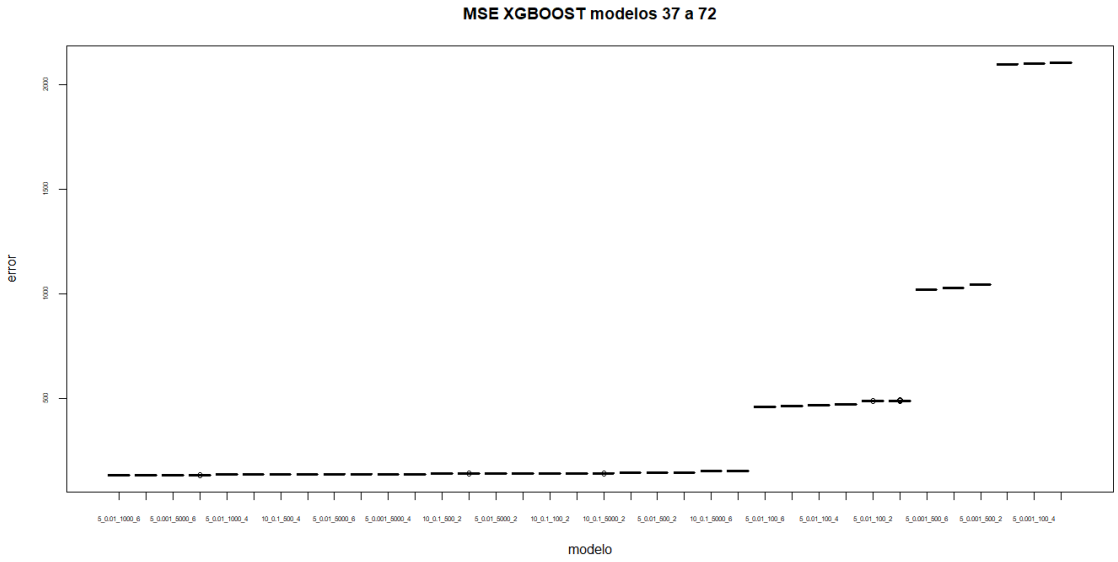

Figura IV.9: Gráfico con los modelos de 37 a 72 de Xgboost. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

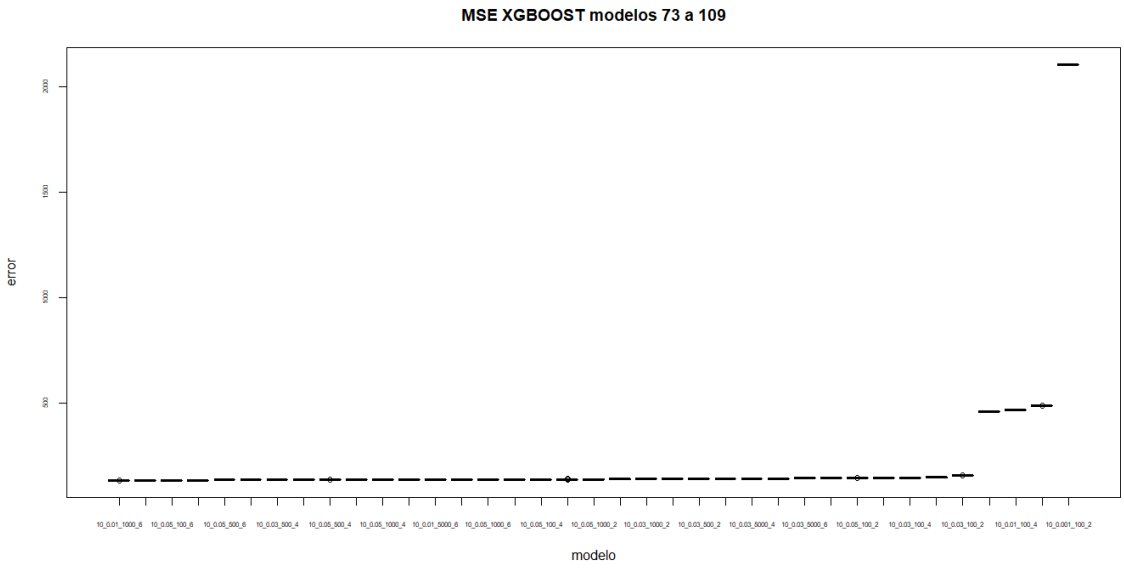

Figura IV.10: Gráfico con los modelos de 73 a 100 de Xgboost. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

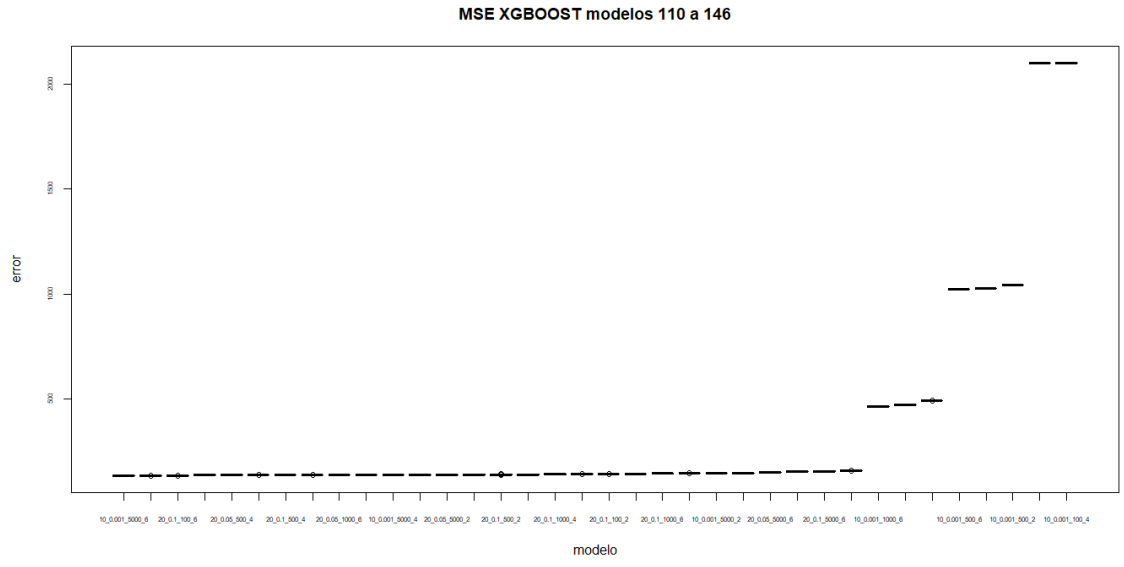

Figura IV.11: Gráfico con los modelos de 101 a 146 de Xgboost. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

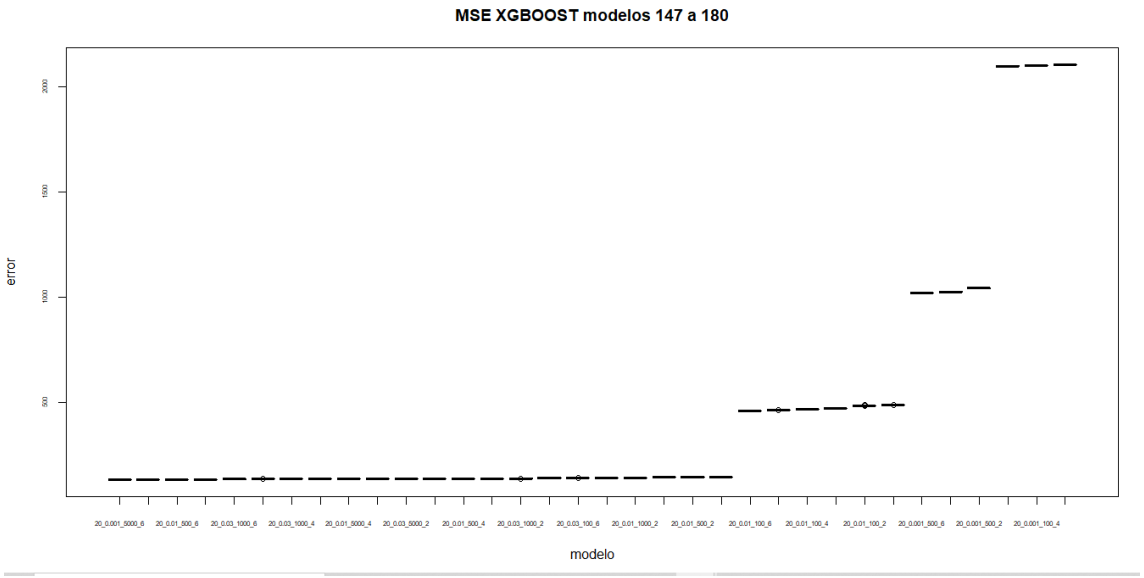

Figura IV.12: Gráfico con los modelos de 147 a 180 de Xgboost. El eje de abscisas representa los modelos y el eje de ordenadas representa su MSE

ANEXO III: Código Python para BBDD

```
def total_features_track(items,str_track):
    count=1
    print(items.shape)
    lista_track=items.iloc[:,4] #cogemos el objeto track donde
vienen los datos de la cancion
    print(len(lista_track))
    features= pd.DataFrame()
    for i in lista_track:
        if (i==None): continue
        aux=json_to_dict_to_df(i)
        id_track=aux.iloc[0,10]
        if (id_track==None): continue
        try:
            track=spotify.track(id_track)
            track_df= json_to_dict_to_df(track)
        except:
            print("Cancion no encontrada")
            continue
        try:
            track1b_df=get_audio_analysis_track(id_track)
        except:
            print("Cancion no encontrada")
            continue
        try:
            track2_df=get_audio_features_track(id_track)
        except:
            print("Cancion 2 no encontrada")
            continue
        album=aux.iloc[0,0]
        try:
            album2_df = get_info_album(album,str_track)
        except:
            print("Album no encontrado")
            continue
        artist=aux.iloc[0,1]
        try:
            artist3_df = get_info_artist(artist)
        except:
            print("Artista no encontrado")
            continue

        tracks = pd.concat([track_df,track1b_df,track2_df],
axis=1, sort=False)
        print(type(track_df))
        print(type(track1b_df))
        print(type(track2_df))
        print(type(tracks))
        print(type(album2_df))
```

```
        print(type(artist3_df))
        tracks = pd.concat([tracks,album2_df,artist3_df],
axis=1, sort=False)
        tracks = tracks.rename(columns=lambda x:
str(x)+'_track')
        features = features.append(tracks, ignore_index=True,
sort=False)
        print(count)
        count=count+1
        print(len(lista_track))
    return features

def get_audio_analysis_track(id_track):
    track1=spotify.audio_analysis(id_track)
    track1_df= json_to_dict_to_df(track1)
    track1a_df=some_caracter(track1_df)
    track1b_df=pd.concat([track1_df,track1a_df], axis=1,
sort=False)
    return track1b_df

def get_audio_features_track(id_track):
    track2=spotify.audio_features(id_track)
    track2_dfa= json_to_dict_to_df(track2)
    muestra=track2_dfa.iloc[0,0]
    track2_df=json_to_dict_to_df(muestra)
    return track2_df

def get_info_artist(artist):
    artist_df= json_to_dict_to_df(artist)
    artist2=artist_df.iloc[0,0]
    artist2_df= json_to_dict_to_df(artist2)
    id_artist=artist2_df.iloc[0,2]
    info_artist=spotify.artist(id_artist)
    info_artist_df= json_to_dict_to_df(info_artist)
    followers=info_artist_df.iloc[0,1]
    followers_df=json_to_dict_to_df(followers)
    followers_df = followers_df.rename(columns=lambda x:
str(x)+'_followers')
    artist3_df =
pd.concat([artist2_df,info_artist_df,followers_df], axis=1,
sort=False)
    artist3_df = artist3_df.rename(columns=lambda x:
str(x)+str('_artist'))
    return artist3_df

def get_info_album(album,type_object_music):
    album_df= json_to_dict_to_df(album)
    if(type_object_music=="track of playlist"):
        id_album=album_df.iloc[0,5]
    else:
        id_album=album_df.iloc[0,4]
    info_album=spotify.album(id_album)
    info_album_df= json_to_dict_to_df(info_album)
    album2_df = pd.concat([album_df,info_album_df], axis=1,
sort=False)
    album2_df = album2_df.rename(columns=lambda x: x+'_album')
```

```
    return album2_df



def json_to_dict_to_df(dataframe):
    jtopy=json.dumps(dataframe) #json.dumps take a dictionary as
input and returns a string as output.
    dict_json=json.loads(jtopy)
    df=pd.DataFrame([dict_json])
    return df

def some_caracter(dataframe):
    sections=dataframe.iloc[0,3]
    #print(len(sections))
    key_l=[]
    loudness_l=[]
    mode_l=[]
    tempo_l=[]
    time_signature_l=[]
    for i in sections:
        df=json_to_dict_to_df(i)
        key=df.iloc[0,2]
        key_l.append(key)
        loudness=df.iloc[0,4]
        loudness_l.append(loudness)
        #print(loudness_l)
        mode=df.iloc[0,5]
        mode_l.append(mode)
        tempo=df.iloc[0,8]
        tempo_l.append(tempo)
        time_signature=df.iloc[0,10]
        time_signature_l.append(time_signature)
    k_l=ratio(key_l)
    #print("imprimimos loudness")
    #print(loudness_l)
    l_l=ratio(loudness_l)
    m_l=ratio(mode_l)
    t_l=ratio(tempo_l)
    ts_l=ratio(time_signature_l)
    lista_final=k_l+l_l+m_l+t_l+ts_l

    df_final = pd.DataFrame(np.array(lista_final).reshape(1,15),
                            columns =
('key_min','key_max','key_mean','loudness_min','loudness_max','l
oudness_mean',

'mode_min','mode_max','mode_mean','tempo_min','tempo_max','tempo
_mean',

'time_signature_min','time_signature_max','time_signature_mean')
)
    #print(df_final)
    return df_final

def ratio(lista):
    #print(lista)
```

```
    minimo=min(lista)
    maximo=max(lista)
    media= statistics.mean(lista)
    l=[minimo,maximo,media]
    #print(l)
    return l

def total_features_track_from_artist(items,str_track):
    count=0
    lista_track=items.iloc[:,8]
    aux=items
    album_s=items.iloc[:,0]
    #artist_s=items.iloc[:,1]
    features= pd.DataFrame()
    for i in lista_track:
        if (i==None): continue
        id_track=i
        try:
            track=spotify.track(id_track)
            track_df= json_to_dict_to_df(track)
        except:
            print("ID no encontrado")
            continue
        try:
            track1b_df=get_audio_analysis_track(id_track)
        except:
            print("Cancion no encontrada")
            continue
        try:
            track2_df=get_audio_features_track(id_track)
        except:
            print("Cancion 2 no encontrada")
            continue
        album=items.iloc[0,0]
        try:
            #album_df= json_to_dict_to_df(album)
            #id_album=album_df.iloc[0,4]
            album2_df = get_info_album(album,str_track)
            #print(id_album)
        except:
            print("Album no encontrado")
            continue
        #album=album_s[count]
        #print("hola2")
        #album_df= json_to_dict_to_df(album)
        #id_album=album_df.iloc[0,4]
        #try:
        #    info_album=spotify.album(id_album)
        #    info_album_df= json_to_dict_to_df(info_album)
        #    album2_df = pd.concat([album_df,info_album_df],
axis=1, sort=False)
        #    album2_df = album2_df.rename(columns=lambda x:
x+'_album')
        #except:
        #    print("Album no encontrado")
        #    count=count+1
```

```
        #    continue

        artist=items.iloc[0,1]
        try:
            artist3_df = get_info_artist(artist)
        except:
            print("Artista no encontrado")
            continue

        tracks = pd.concat([track_df,track1b_df,track2_df],
axis=1, sort=False)
        tracks = pd.concat([tracks,album2_df,artist3_df],
axis=1, sort=False)
        tracks = tracks.rename(columns=lambda x:
str(x)+'_track')
        features = features.append(tracks, ignore_index=True,
sort=False)
        #print(count)
        count=count+1
        #print(len(lista_track))
    return features
import sys
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import json
import ast
import pandas as pd
import statistics
import numpy as np
cid ="e5edecb240e447a79f9fc1f03ca04af7"
secret = "f73e0d0c935d4febb70d51456d8f393a"
spotify =
spotipy.Spotify(client_credentials_manager=SpotifyClientCredenti
als(client_id=cid, client_secret=secret, requests_timeout =
100))
print(spotify.requests_timeout)
spotify.requests_timeout=100
print(spotify)
country_list=[ "AD", "AR", "AT", "AU", "BE", "BG", "BO",
              "BR", "CH", "CL", "CO", "CR", "CY", "CZ",
              "DK", "DO", "EC", "EE", "ES", "FI", "FR", "GB",
"GR", "GT", "HK",
              "HN", "HU", "IE", "IS", "IT", "LI", "LT", "LU",
"LV", "MC", "MT", "MY",
              "NI", "NL", "NO", "NZ", "PA", "PE", "PH", "PL",
"PT", "PY", "RO", "SE", "SG",
              "SI", "SK", "SV", "TR", "TW", "UY"]
#Aqui habria que hacer el bucle de los países ARA LAS CANCIONES
DE LAS LISTAS MÁS IMPORTANTES OR PAIS
contador=0
lista_pais=pd.DataFrame()
for c in country_list:
    print(c)
    #if(contador==2):
    #    break
    results2 = spotify.featured_playlists(country=c)
```

```
    #Creamos un dataframe con las playlists
    hola=results2['playlists']
    df_playlists=json_to_dict_to_df(hola)
    df_playlists = df_playlists.rename(columns=lambda x:
str(x)+'_playlist')
    #print(df_playlists)
    #print(hola)
    listas=df_playlists['items_playlist']
    def_playlists= pd.DataFrame()
    #df_playlists['country']= 'ES'#al final
    for lista in listas:
        aux_playlists= json_to_dict_to_df(lista)
        print(aux_playlists)
        aux_playlists=aux_playlists.iloc[0,0]
        def_playlists = def_playlists.append(aux_playlists,
ignore_index=True)
        print(def_playlists)

    def_playlists = def_playlists.rename(columns=lambda x:
str(x)+'_playlist_')
    df_playlists = pd.concat([df_playlists,def_playlists],
axis=1)
    #print(df_playlists.columns)

    ids=df_playlists['id_playlist_']
    tracks_playlists= pd.DataFrame()
    #l=1
    for id_lista in ids: #vamos a coger el id de las listas para
buscar sus canciones
        print(id_lista)
        tracks_of_playlist =
spotify.playlist_tracks(playlist_id=id_lista)
        if (tracks_of_playlist==None): continue
        aux_playlists=json_to_dict_to_df(tracks_of_playlist)
        #jtopy2=json.dumps(tracks_of_playlist) #json.dumps take
a dictionary as input and returns a string as output.
        #dict_json2=json.loads(jtopy2)
        #aux_playlists= pd.DataFrame([dict_json2])#tiene que ser
dict_json2
        items=aux_playlists.iloc[0,1]
        print("IMPRIMIR EL ITEM")
        #print(items)
        #items_df=json_to_dict_to_df(items)
        jtopy3=json.dumps(items) #json.dumps take a dictionary
as input and returns a string as output.
        dict_json3=json.loads(jtopy3)
        items_df= pd.DataFrame(dict_json3)
        listatrack= items_df.iloc[:,4]
        #listatrack.tolist()
        print("IMPRIMO EL TRACK")
        print(len(listatrack))
        features=total_features_track(items_df,"track of
playlist")
        #features=total_features_track(listatrack)
        total_df= pd.concat([aux_playlists,features], sort=
False, axis=1)
```

```
        lista_name=list(aux_playlists)
        col=0
        for i in lista_name:
            total_df[i]=pd.Series(total_df.iloc[0,col])
            col=col+1
        tracks_playlists = tracks_playlists.append(total_df,
ignore_index=True, sort=False)
        #tracks_playlists =
tracks_playlists.append(aux_playlists, ignore_index=True,
sort=False)
        #if(l==2):
        #        break
        #l=l+1

    tracks_playlists.columns
    tracks_playlists['country']= c#al final
    lista_pais = lista_pais.append(tracks_playlists,
ignore_index=True)
    contador=contador+1

lista_pais.columns
#lista_pais['country']
file=lista_pais.to_csv (r'C:\home\playlist_paises_2.csv',
index=False,header=1)

#CGOEMOS LAS CANCIONES DEL ARTISTA.
import sys
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import json
import ast
import pandas as pd
import statistics
import numpy as np
cid ="e5edecb240e447a79f9fc1f03ca04af7"
secret = "f73e0#d0c935d4febb70d51456d8f393a"
spotify =
spotipy.Spotify(client_credentials_manager=SpotifyClientCredenti
als(client_id=cid, client_secret=secret, requests_timeout =
100))
print(spotify.requests_timeout)
spotify.requests_timeout=100
print(spotify)
country_list=[ "AD", "AR", "AT", "AU", "BE", "BG", "BO",
              "BR", "CH", "CL", "CO", "CR", "CY", "CZ",
              "DK", "DO", "EC", "EE", "ES", "FI", "FR", "GB",
"GR", "GT", "HK",
              "HN", "HU", "IE", "IS", "IT", "LI", "LT", "LU",
"LV", "MC", "MT", "MY",
              "NI", "NL", "NO", "NZ", "PA", "PE", "PH", "PL",
"PT", "PY", "RO", "SE", "SG",
              "SI", "SK", "SV", "TR", "TW", "UY"]

file_country = pd.read_csv("C:\home\playlist_paises_2.csv")#aqui
tienes que poner tu ruta
#file_country.columns
```

```
id_track=file_country['id_track']
#id_track_sin_drop= id_track.drop_duplicates()
#names_artista=file_country['name_artist_track']
#ids_artista=file_country['id_artist_track']
#names_artista_sin_drop=names_artista.drop_duplicates()
#ids_artista_sin_drop=ids_artista.drop_duplicates()
#fil3e=ids_artista_sin_drop.to_csv (r'ids_artistas.csv',
index=False,header=1)

ids2 = pd.read_csv("ids_artistas.csv")
ids_list=ids2.iloc[:,0]
yds_lis=ids_list.drop_duplicates()
ids_list2=pd.Series.tolist(ids_list)
#yds_lis=ids_list2.drop_duplicates()
#ids_list = ids2.values.tolist()
#ids2=pd.Dataframe.tolist(ids_artista_sin_drop)
cont=0
#ids3=ids_list2[501:1015]
#len(ids3)
#print(len(ids2))
artistas_songs3=pd.DataFrame()
for i in ids_list2:
#for i in ids3:
    print(cont)
    if (cont==2): break
    if (i==None): continue
    try:
        a=spotify.artist_top_tracks(i)
    except:
        print("Artista no encontrado")
        continue

    b=json_to_dict_to_df(a)
    artist_t=b.iloc[0,0]
    if (artist_t==[]): continue
    artist_a=pd.DataFrame(artist_t)
    #f=total_features_track2(artist_a)
    f=total_features_track_from_artist(artist_a,"hola")
    #print(id_album)
    artistas_songs3 = artistas_songs3.append(f,
ignore_index=True, sort=False)
    cont=cont+1

file2=artistas_songs.to_csv (r'segundo_501_a_1015.csv.csv',
index=False,header=1)

f1 = pd.read_csv('primer_0_a_500.csv')
f2 = pd.read_csv('obs_500.csv')
f3 = pd.read_csv('holasegundo_500_a_1015.csv')
f4 = pd.read_csv('adiostercero_1015_a_1515.csv')
total=pd.DataFrame()
total  = total.append(f1, ignore_index=True, sort=False)
total  = total.append(f2, ignore_index=True, sort=False)
total  = total.append(f3, ignore_index=True, sort=False)
total  = total.append(f4, ignore_index=True, sort=False)
```

```
file_total=total.to_csv (r'total_artistas.csv',
index=False,header=1)
t=pd.read_csv('total_artistas.csv')
t['Unnamed: 0']
list_columns=t.columns
print(type(list_columns))
print(list_columns.tolist())
d=list_columns.tolist()
c=d.to_csv (r'columns_total_artistas.csv', header=1)
paises=total.copy()
paises2.drop(['href', 'items', 'limit', 'next', 'offset',
'previous', 'total'], axis='columns', inplace=True)
paises.columns
p=paises.drop_duplicates(subset='id_track',keep='first')
ids=p['id_track']
ids2=pd.DataFrame()
ids2=ids2.append(ids, ignore_index=True, sort=False)
ids2=ids2.append(id_track, ignore_index=True, sort=False)
total_ids_df= pd.concat([ids,id_track], sort= False, axis=0)
p1=ids2.drop_duplicates(keep='first')
hola=t['id_track']
hola[0]
adios=t['loudness_max_track']
adios[0]
3NDgV66iYYySIgo5RSvQ5d
[-46.572, -5.403, -9.625533333333333]

id_track=file_country['id_track']
ids2 = pd.read_csv("ids_artistas.csv")

t=pd.read_csv('total_artistas.csv')
dfeliminado=t.copy()
dfeliminado.drop([ 'Unnamed: 0', 'album_track', 'artists_track',
'available_markets_track', 'external_ids_track',
 'external_urls_track', 'href_track',  'is_local_track',
'name_track', 'preview_url_track', 'uri_track',
 'track_track',  'bars_track', 'beats_track', 'sections_track',
'segments_track', 'tatums_track', 'type_track.1',
 'id_track.1', 'uri_track.1', 'track_href_track',
'analysis_url_track', 'duration_ms_track.1',
'artists_album_track',
 'external_urls_album_track',  'href_album_track',
'id_album_track', 'images_album_track', 'name_album_track',
'release_date_album_track',
 'release_date_precision_album_track',  'uri_album_track',
'album_type_album_track.1',  'artists_album_track.1',
'available_markets_album_track',
 'copyrights_album_track', 'external_ids_album_track',
'external_urls_album_track.1', 'genres_album_track',
'href_album_track.1', 'id_album_track.1',
 'images_album_track.1', 'name_album_track.1',
'popularity_album_track', 'release_date_album_track.1',
'release_date_precision_album_track.1',
'total_tracks_album_track.1',
```

```
 'tracks_album_track', 'type_album_track.1',
'uri_album_track.1', 'external_urls_artist_track',
'href_artist_track', 'id_artist_track', 'name_artist_track',
 'type_artist_track', 'uri_artist_track',
'external_urls_artist_track.1', 'followers_artist_track',
'href_artist_track.1', 'id_artist_track.1',
'images_artist_track',
 'name_artist_track.1', 'popularity_artist_track',
'type_artist_track.1', 'uri_artist_track.1',
 'href_followers_artist_track'], axis='columns', inplace=True)

paises_eliminado=file_country.copy()
paises_eliminado.drop([ 'href', 'items', 'limit', 'next',
'offset', 'previous', 'total', 'album_track', 'artists_track',
'available_markets_track', 'external_ids_track',
 'external_urls_track', 'href_track', 'is_local_track',
'name_track', 'preview_url_track', 'uri_track',
 'track_track',  'bars_track', 'beats_track', 'sections_track',
'segments_track', 'tatums_track', 'type_track.1',
 'id_track.1', 'uri_track.1', 'track_href_track',
'analysis_url_track', 'duration_ms_track.1',
'artists_album_track',
 'external_urls_album_track',  'href_album_track',
'id_album_track', 'images_album_track', 'name_album_track',
'release_date_album_track',
 'release_date_precision_album_track',  'uri_album_track',
'album_type_album_track.1',  'artists_album_track.1',
'available_markets_album_track',
 'copyrights_album_track', 'external_ids_album_track',
'external_urls_album_track.1', 'genres_album_track',
'href_album_track.1', 'id_album_track.1',
 'images_album_track.1', 'name_album_track.1',
'popularity_album_track', 'release_date_album_track.1',
'release_date_precision_album_track.1',
'total_tracks_album_track.1',
 'tracks_album_track', 'type_album_track.1',
'uri_album_track.1', 'external_urls_artist_track',
'href_artist_track', 'id_artist_track', 'name_artist_track',
 'type_artist_track', 'uri_artist_track',
'external_urls_artist_track.1', 'followers_artist_track',
'href_artist_track.1', 'id_artist_track.1',
'images_artist_track',
 'name_artist_track.1', 'popularity_artist_track',
'type_artist_track.1', 'uri_artist_track.1',
 'href_followers_artist_track','country',
'available_markets_album_track.1'], axis='columns',
inplace=True)
col=paises_eliminado.columns
type(col)
new_order = ['id_track','disc_number_track',
'duration_ms_track', 'explicit_track',
       'popularity_track', 'track_number_track', 'type_track',
'meta_track',
       'key_min_track', 'key_max_track', 'key_mean_track',
       'loudness_min_track', 'loudness_max_track',
'loudness_mean_track',
```

```
       'mode_min_track', 'mode_max_track', 'mode_mean_track',
       'tempo_min_track', 'tempo_max_track', 'tempo_mean_track',
       'time_signature_min_track', 'time_signature_max_track',
       'time_signature_mean_track', 'acousticness_track',
'danceability_track',
       'energy_track', 'instrumentalness_track', 'key_track',
'liveness_track',
       'loudness_track', 'mode_track', 'speechiness_track',
'tempo_track',
       'time_signature_track', 'valence_track',
'album_type_album_track',
       'total_tracks_album_track', 'type_album_track',
'label_album_track',
       'genres_artist_track', 'total_followers_artist_track']
df2=dfeliminado[['id_track','disc_number_track',
'duration_ms_track', 'explicit_track',
       'popularity_track', 'track_number_track', 'type_track',
'meta_track',
       'key_min_track', 'key_max_track', 'key_mean_track',
       'loudness_min_track', 'loudness_max_track',
'loudness_mean_track',
       'mode_min_track', 'mode_max_track', 'mode_mean_track',
       'tempo_min_track', 'tempo_max_track', 'tempo_mean_track',
       'time_signature_min_track', 'time_signature_max_track',
       'time_signature_mean_track', 'acousticness_track',
'danceability_track',
       'energy_track', 'instrumentalness_track', 'key_track',
'liveness_track',
       'loudness_track', 'mode_track', 'speechiness_track',
'tempo_track',
       'time_signature_track', 'valence_track',
'album_type_album_track',
       'total_tracks_album_track', 'type_album_track',
'label_album_track',
       'genres_artist_track', 'total_followers_artist_track']]
df3=paises_eliminado[['id_track','disc_number_track',
'duration_ms_track', 'explicit_track',
       'popularity_track', 'track_number_track', 'type_track',
'meta_track',
       'key_min_track', 'key_max_track', 'key_mean_track',
       'loudness_min_track', 'loudness_max_track',
'loudness_mean_track',
       'mode_min_track', 'mode_max_track', 'mode_mean_track',
       'tempo_min_track', 'tempo_max_track', 'tempo_mean_track',
       'time_signature_min_track', 'time_signature_max_track',
       'time_signature_mean_track', 'acousticness_track',
'danceability_track',
       'energy_track', 'instrumentalness_track', 'key_track',
'liveness_track',
       'loudness_track', 'mode_track', 'speechiness_track',
'tempo_track',
       'time_signature_track', 'valence_track',
'album_type_album_track',
       'total_tracks_album_track', 'type_album_track',
'label_album_track',
       'genres_artist_track', 'total_followers_artist_track']]
```

```
paises_eliminado[paises_eliminado.columns[new_order]]
if(list(df2.columns)==list(df3.columns)):
    print("son iguales")
dfeliminado.columns

df4=pd.DataFrame()
df4=df4.append(df3, ignore_index=True, sort=False)
df4=df4.append(df2, ignore_index=True, sort=False)
df5=df4.drop_duplicates(subset='id_track',keep='first')
file_total=df5.to_csv (r'final.csv', index=False,header=1)
final=pd.read_csv('final.csv')
ids_list=final['id_track']
yds_lis=ids_list.drop_duplicates()
ids_list2=pd.Series.tolist(ids_list)
#yds_lis=ids_list2.drop_duplicates()
#ids_list = ids2.values.tolist()
#ids2=pd.Dataframe.tolist(ids_artista_sin_drop)
cont=0
ids3=ids_list2[13000:14094]
#len(ids3)
#print(len(ids_list2))
artistas_songs3=pd.DataFrame()
for i in ids3:
#for i in ids3:

    #print(cont)
    #if (cont==2): break
    if (i==None): continue
    #try:
    a=get_audio_analysis_track(i)
    a['id_track']=i
    #except:
    #    print("Error en la cancion")
    #    cont=cont+1
     #   continue

    #b=json_to_dict_to_df(a)
    #artist_t=b.iloc[0,0]
    #if (artist_t==[]): continue
    #artist_a=pd.DataFrame(artist_t)
    #f=total_features_track2(artist_a)
    #f=total_features_track_from_artist(artist_a,"hola")
    #print(id_album)
    artistas_songs3 = artistas_songs3.append(a,
ignore_index=True, sort=False)
    div=cont%1000
    if(div==0):
        print(cont)
    cont=cont+1

featl=artistas_songs3.to_csv (r'total_loudness14.csv',
index=False,header=1)

l1 = pd.read_csv('total_loudness1.csv')
l2 = pd.read_csv('total_loudness2.csv')
l3 = pd.read_csv('total_loudness3.csv')
```

```
l4 = pd.read_csv('total_loudness4.csv')
l5 = pd.read_csv('total_loudness5.csv')
l6 = pd.read_csv('total_loudness6.csv')
l7 = pd.read_csv('total_loudness7.csv')
l8 = pd.read_csv('total_loudness8.csv')
l9 = pd.read_csv('total_loudness9.csv')
l10 = pd.read_csv('total_loudness10.csv')
l11 = pd.read_csv('total_loudness11.csv')
l12 = pd.read_csv('total_loudness12.csv')
l13 = pd.read_csv('total_loudness13.csv')
l14 = pd.read_csv('total_loudness14.csv')

total_final=pd.DataFrame()
total_final  = total_final.append(l1, ignore_index=True,
sort=False)
total_final  = total_final.append(l2, ignore_index=True,
sort=False)
total_final  = total_final.append(l3, ignore_index=True,
sort=False)
total_final  = total_final.append(l4, ignore_index=True,
sort=False)
total_final  = total_final.append(l5, ignore_index=True,
sort=False)
total_final  = total_final.append(l6, ignore_index=True,
sort=False)
total_final  = total_final.append(l7, ignore_index=True,
sort=False)
total_final  = total_final.append(l8, ignore_index=True,
sort=False)
total_final  = total_final.append(l9, ignore_index=True,
sort=False)
total_final  = total_final.append(l10, ignore_index=True,
sort=False)
total_final  = total_final.append(l11, ignore_index=True,
sort=False)
total_final  = total_final.append(l12, ignore_index=True,
sort=False)
total_final  = total_final.append(l13, ignore_index=True,
sort=False)
total_final  = total_final.append(l14, ignore_index=True,
sort=False)

copy_total= total_final.copy()

featl1=total_final.to_csv (r'mitadfinal.csv',
index=False,header=1)
total_final.columns
total2=total_final.copy()
total_final.drop([ 'bars', 'beats', 'meta', 'sections',
'segments', 'tatums', 'track'], axis='columns', inplace=True)
final2=final.copy()
final.columns
final2.drop([ 'key_min_track', 'key_max_track',
'key_mean_track',
       'loudness_min_track', 'loudness_max_track',
'loudness_mean_track',
```

```
       'mode_min_track', 'mode_max_track', 'mode_mean_track',
       'tempo_min_track', 'tempo_max_track', 'tempo_mean_track',
       'time_signature_min_track', 'time_signature_max_track',
       'time_signature_mean_track'], axis='columns',
inplace=True)

final_def= pd.concat([final2,total_final], sort= False, axis=1)
final_def.columns
final3=final_def.iloc[:,0:41]
final3.columns
f=final3.drop_duplicates(subset='id_track',keep='first')
fichero_final=final3.to_csv (r'final_definitivo.csv',
index=False,header=1)
fichero = pd.read_csv('final_definitivo.csv')
fichero1=fichero.copy()
print(len(fichero.columns))
print(fichero['type_album_track'])
if(list(fichero['type_album_track'])==(list(fichero['type_track'
])))):
    print("son iguales")

f1=fichero['type_album_track']
f=f1.tolist()
f2=fichero['type_track']
f_=f2.tolist()
f3=fichero['type_track']
_f_=f3.tolist()

f4=fichero['meta_track']
f=f4.tolist()


fichero1.drop(['meta_track','type_album_track','type_track' ],
axis='columns', inplace=True)
print(len(fichero1.columns))
print(fichero1.columns)
fichero1.columns=['id_track', 'disc_number_track',
'duration_ms_track', 'explicit_track',
       'popularity_track', 'track_number_track',
'acousticness_track',
       'danceability_track', 'energy_track',
'instrumentalness_track',
       'key_track', 'liveness_track', 'loudness_track',
'mode_track',
       'speechiness_track', 'tempo_track',
'time_signature_track',
       'valence_track', 'album_type', 'total_tracks_album',
       'label_album', 'genres_artist',
       'total_followers_artist', 'key_min_track',
'key_max_track', 'key_mean_track',
       'loudness_min_track', 'loudness_max_track',
'loudness_mean_track', 'mode_min_track', 'mode_max_track',
       'mode_mean_track', 'tempo_min_track', 'tempo_max_track',
'tempo_mean_track',
       'time_signature_min_track', 'time_signature_max_track',
'time_signature_mean_track']
```

```
fichero_final_=fichero1.to_csv (r'final_definitivo_.csv',
index=False,header=1)
```

BIBLIOGRAFÍA

[1]. *Spotify*. (s. f.). Recuperado 23 de septiembre de 2020, de https://www.spotify.com/es/premium/

[2]. García, F. M. (s. f.). *Beneficios del Big Data cuando la música es el hilo conductor*. Hipertextual. Recuperado 23 de septiembre de 2020, de https://hipertextual.com/presentado-por/banco-sabadell/beneficios-del-big-data-musica

[3]. lainformacion.com. (2018, mayo 11). *Predicen el devenir de la economía analizando qué se escucha en Spotify*. La Información. https://www.lainformacion.com/management/predicen-el-devenir-de-la-economia-analizando-que-se-escucha-en-spotify/6347917/

[4]. *Evolución de los reproductores de música, desde el fonógrafo hasta los MP4*. (2019, abril 22). *Depau Sistemas*. https://blog.depau.es/evolucion-de-los-reproductores-de-musica-desde-el-fonografo-hasta-los-mp4/

[5]. Redacción BBC News Mundo. (2017, noviembre 21). *8 maneras de reproducir música desde la creación del fonógrafo hace 140 años*. https://www.bbc.com/mundo/noticias-42060516

[6]. López, J. M. (s. f.). Un repaso a la historia del registro musical desde su origen hasta hoy. Hipertextual. Recuperado 23 de septiembre de 2020, de https://hipertextual.com/presentado-por/fundacion-telefonica/registro-musical-grabacion

[7]. La música: Crecimiento y evolución desde el vinilo hasta el *streaming*. (2020, mayo 13). *PromocionMusical.es*. https://promocionmusical.es/la-musica-crecimiento-y-evolucion-desde-el-vinilo-hasta-el-streaming/

[8]. CWS. (s. f.). *La Historia del streaming.* Recuperado 23 de septiembre de 2020, de https://conceptoweb-studio.com/index.php/blog-streaming/item/27-la-historia-del-streaming

[9]. García, J. (2020, febrero 5). *Quién está ganando la guerra del streaming de música*. Xataka. https://www.xataka.com/empresas-y-economia/quien-esta-ganando-guerra-streaming-musica

[10]. *Música en streaming: Ingresos a nivel mundial 2010-2019*. (s. f.). Statista. Recuperado 23 de septiembre de 2020, de https://es.statista.com/estadisticas/600954/ingresos-mundiales-de-suscripcion-a-servicios-de-reproduccion-musical-online/

[11]. *Historia de Spotify: Nacimiento y evolución - Marketing4eCommerce*. (2019, septiembre 16). Marketing 4 Ecommerce - Tu revista de marketing online para e-commerce. https://marketing4ecommerce.net/historia-de-spotify-del-lider-de-la-musica-en-streaming/

[12]. *Infografía: Spotify alcanza los 130 millones de suscriptores de pago*. (s. f.). Statista Infografías. Recuperado 23 de septiembre de 2020, de https://es.statista.com/grafico/19793/usuarios-activos-y-de-pago-de-spotify/

[13]. *Cómo funciona el Big Data en la industria musical*. (2015, febrero 6). El blog de T-Systems Iberia. https://www.t-systemsblog.es/como-funciona-el-big-data-en-la-industria-musical/

[14]. H, U. F. | M. Ú. actualización:19-08-2017 | 02:32 H.-08-2017 | 02:32. (2017, agosto 19). *La fórmula matemática de la canción del verano*. La Razón. https://www.larazon.es/cultura/la-formula-matematica-de-la-cancion-del-verano-IN15813946/

[15]. El fenómeno de Despacito, un éxito planeado a detalle. (2017, julio 28). *Revista Merca2.0*. https://www.merca20.com/el-fenomeno-de-despacito-un-exito-planeado-detalle/

[16]. Calviño, A. (2019). *Apuntes de la asignatura Técnicas y metodología de la minería de datos (SEMMA)*.

[17]. *Introducción a la Regresión Lineal Múltiple*. (s. f.). Recuperado 23 de septiembre de 2020, de https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple

[18]. *Portela, J. (2019). Apuntes de la asignatura Machine Learning.*

[19]. *Comunicación de las neuronas*. (s. f.). sinapsis. Recuperado 23 de septiembre de 2020, de https://valentinap64.wixsite.com/sinapsis/single-post/2016/01/09/Comunicaci%C3%B3n-de-las-neuronas

[20]. *Redes Neuronales: Una visión superficial—Fernando Sancho Caparrini*. (s. f.). Recuperado 23 de septiembre de 2020, de http://www.cs.us.es/~fsancho/?e=72

[21]. *Árboles de predicción: Random forest, gradient boosting y C5.0*. (s. f.). Recuperado 23 de septiembre de 2020, de https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting

[22]. johanna.orellana@ucuenca.edu.ec, J. O. A.-. (s. f.). *Arboles de decision y Random Forest*. Recuperado 23 de septiembre de 2020, de https://bookdown.org/content/2031/arboles-de-decision-parte-i.html

[23]. *Problemas Búsqueda y Planificación—Fernando Sancho Caparrini*. (s. f.). Recuperado 23 de septiembre de 2020, de http://www.cs.us.es/~fsancho/?e=33

[24]. Brownlee, J. (2016, abril 21). Bagging and Random Forest Ensemble Algorithms for Machine Learning. *Machine Learning Mastery*. https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/

[25]. *¿Cuál es la diferencia entre los métodos de bagging y los de boosting?* (2020, febrero 24). MachineLearningParaTodos.com. https://machinelearningparatodos.com/cual-es-la-diferencia-entre-los-metodos-de-bagging-y-los-de-boosting/

[26]. Maldonado, S., & Weber, R. (2012, septiembre). Modelos de Selección de Atributos para Support Vector Machines. *Revista Ingeniería de Sistemas*, *XXVI*. Septiembre 2012

[27]. *What is the main difference between a SVM and SVR? - Quora*. (s. f.). Recuperado 23 de septiembre de 2020, de https://www.quora.com/What-is-the-main-difference-between-a-SVM-and-SVR

[28]. Jacob Avila. (2018, octubre). Support Vector Regression (SVR). *JacobSoft*. https://www.jacobsoft.com.mx/es_mx/support-vector-regression/

[29]. *Support Vector Regression Made Easy(with Python Code)*. (s. f. Www.Aionlinecourse.Com. Recuperado 23 de septiembre de 2020, de https://www.aionlinecourse.com/tutorial/machine-learning/support-vector-regression

[30]. *Welcome to Spotipy! —Spotipy 2.0 documentation*. (s. f.). Recuperado 23 de septiembre de 2020, de https://spotipy.readthedocs.io/en/2.14.0/#

[31]. *Web API | Spotify for Developers*. (s. f.). Recuperado 23 de septiembre de 2020, de https://developer.spotify.com/documentation/web-api/

[32]. *Get a Track | Spotify for Developers*. (s. f.). Recuperado 23 de septiembre de 2020, de https://developer.spotify.com/documentation/web-api/reference/tracks/get-track/