TRABAJO DE FIN DE GRADO

Grado en Estadística

Título: Análisis estadístico de los gustos musicales europeos (Eurovisión)

Autor: Pol Garcia Sató

Director: Sergi Ramírez Mitjans

Departamento: Estadística i Investigación Operativa

Convocatoria: Junio 2023

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Agradecimientos

Me gustaría expresar mi más sincero agradecimiento a varias personas que han sido fundamentales en el desarrollo y finalización de este trabajo de final de grado.

En primer lugar, quiero agradecer al profesor Sergi Ramírez por su apoyo y sus orientaciones a lo largo de todo el proceso. Su profundo conocimiento en el tema y dedicación han sido de gran importancia para llevar a cabo el proyecto y alcanzar los objetivos iniciales.

También me gustaría dar las gracias mis padres que, al fin y al cabo, son los que ven y entienden de primera mano las dificultades que he tenido y que, con su respaldo y motivación, me han impulsado a superar los obstáculos.

Asimismo, quiero agradecer a mis amigos y pareja, quienes han estado a mi lado durante todo este proceso. Siempre los he tenido tanto física como telemáticamente para ayudarme a mantener el ánimo arriba y salir de la rutina para despejar la mente.

Además, me gustaría mencionar las dificultades que surgieron al crear las distintas bases de datos completas para este trabajo. A pesar de los problemas encontrados, puedo decir que, gracias a la perseverancia y al esfuerzo constante, finalmente logré confeccionar el trabajo que me propuse al comienzo.

Abstract

Castellano

Cuando se acerca el mes de mayo de cada año, siempre nos preguntamos quién va a ser el ganador de la nueva edición del festival de Eurovisión. Este evento no solo atrae la mirada de los países europeos, sino que pone el foco del talento musical por el resto del mundo.

Gracias a mi conocimiento sobre este acontecimiento anual, este proyecto tiene como objetivo poder predecir, mediante la minería de datos, cuál será el ganador, la posición y la puntuación de los representantes de cada país que participarán en la edición de Eurovisión de 2023.

Para ello se realizará un estudio de los principales festivales nacionales que más impactan en Eurovisión, como el Benidorm Fest de España y el Melodifestivalen de Suecia, para conocer los gustos musicales de estos países, y así poder inferenciar en los resultados europeos. Mediante información de las apuestas, características de las canciones en Spotify y un análisis de texto de la letra de cada una de ellas, se formarán bases de datos con la finalidad de crear modelos de predicción. Además, también se introducirá un índice (distancia) de amistad entre los países para conocer correctamente sus patrones de votaciones.

Finalmente, se logra obtener unas predicciones con bastante precisión y observamos que nuestro mejor modelo para conseguirlo es la red neuronal.

Palabras claves: Eurovisión, Cuotas, *Web-Scraping*, Spotify, *Topic-Modelling*, Indicador, Grafos, Predicción, Ganador. Clasificación AMS: 62M20, 62M45, 62M86, 62M99.

Català

Quan s'apropa el mes de maig de cada any, sempre ens preguntem qui serà el guanyador de la nova edició del festival d'Eurovisió. Aquest esdeveniment no només atreu les mirades dels països europeus, sinó que posa l'accent en el talent musical a nivell mundial.

Gràcies al meu coneixement sobre aquest esdeveniment anual, aquest projecte té com a objectiu poder predir, mitjançant la mineria de dades, qui serà el guanyador, la posició i la puntuació dels representants de cada país que participaran en l'edició d'Eurovisió del 2023.

Per aconseguir-ho, es realitzarà un estudi dels principals festivals nacionals que tenen més impacte en Eurovisió, com el Benidorm Fest d'Espanya i el Melodifestivalen de Suècia, per conèixer els gustos musicals d'aquests països i així poder influir en els resultats europeus. Mitjançant informació de les apostes, característiques de les cançons a Spotify i un anàlisi de text de la lletra de cada una d'elles, es crearan bases de dades amb la finalitat de crear models de predicció. A més, també s'introduirà un índex (distància) d'amistat entre els països per conèixer correctament els seus patrons de votacions.

Finalment, s'aconsegueixen prediccions amb una gran precisió i observem que el nostre millor model per a aconseguir-ho és la red neuronal.

Paraules clau: Eurovisió, Quotes, *Web-Scraping*, Spotify, *Topic-Modelling*, Indicador, Grafs, Predicció, Guanyador. Clasificació AMS: 62M20, 62M45, 62M86, 62M99.

English

When May approaches each year, we always wonder who will be the winner of the new edition of the Eurovision Song Contest. This event not only attracts the attention of European countries but also puts the spotlight on musical talent worldwide.

Thanks to my knowledge of this annual event, this project aims to predict, through data mining, the winner, position, and score of the representatives from each country participating in the 2023 Eurovision edition.

To achieve this, a study will be conducted on the major national festivals that have the most impact on Eurovision, such as Spain's Benidorm Fest and Sweden's Melodifestivalen, to understand the preferences of these countries and infer their influence on the European results. By utilizing information from betting, song characteristics on Spotify, and text analysis of the lyrics, databases will be formed to create prediction models. Additionally, an index (distance) of friendship between countries will also be introduced to properly understand their voting patterns.

Finally, highly accurate predictions are obtained, and we observe that our best model for this purpose is neural network.

Keywords: Eurovision, Odds, Web-Scraping, Spotify, Topic-Modelling, Indicator, Graphs, Prediction, Winner. AMS Clasification: 62M20, 62M45, 62M86, 62M99.

Lista de figuras

Lista de tablas

ÍNDICE

1. Introducción

Como modelo de competición musical a nivel internacional, Eurovisión ha generado opiniones muy diversas a lo largo de los años. Algunas personas disfrutan de la variedad musical y cultural que se presenta en el evento, mientras que otras critican la falta de calidad musical y la excesiva politización del evento.

Una de las críticas más comunes que se hacen a Eurovisión es que la calidad musical no siempre es la mejor, y que se dan más importancia a otros factores como la imagen, la coreografía y el espectáculo en sí mismo. También se ha señalado la politización del evento, en el que los países votan en función de sus alianzas políticas y no necesariamente en función de la calidad de la canción. Sin embargo, para muchas personas Eurovisión es un evento muy emocionante, que les permite descubrir nuevos artistas y géneros musicales de diferentes países. Además, la competición también ha sido un espacio para la reivindicación y la visibilidad de minorías y grupos sociales, como ocurrió con la victoria de Conchita Würst [1] en 2014.

Estos factores se verán reflejados en el proyecto como variables de predicción para la minería de datos [2]. La minería de datos es un proceso mediante el cual se extraen patrones y conocimientos útiles a partir de grandes conjuntos de datos. Es una técnica muy utilizada en la actualidad para la toma de decisiones en una gran variedad de campos, como el marketing, la medicina, la ciencia, la tecnología, entre otros.

En este proyecto de minería de datos tendremos múltiples objetivos, desde la identificación de patrones y tendencias en los datos, hasta la predicción de los resultados de los festivales. Para ello, será necesario tener un conjunto de datos adecuado y limpio, así como una metodología clara y rigurosa para el análisis y la interpretación de los resultados.

Una vez tendremos los datos adecuados, procederemos a la exploración de los mismos mediante técnicas de visualización y análisis estadístico. A continuación, se seleccionan las variables relevantes para el proyecto y se aplican técnicas de modelado y aprendizaje automático para la identificación de patrones y la predicción de resultados.

Por otro lado, usaremos el *Deep Learning*, subconjunto del *Machine Learning*, donde nos centraremos en la creación de redes neuronales artificiales profundas. Estas redes estarán diseñadas para imitar el funcionamiento del cerebro humano y podrán aprender y realizar tareas complejas como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la toma de decisiones.

Todo esto se tendrá en cuenta para realizar este proyecto y cumplir los objetivos que se presentaran a continuación.

[1] Cantante austríaca ganadora de Eurovisión en 2014 que generó polémica por la relación de la victoria con su lucha por los derechos del colectivo LGTBIQ+ y mandando un mensaje contra las leyes homófobas de Vladimir Putin tras vencer la gala.

[2] Proceso de descubrimiento y análisis de patrones y tendencias en conjuntos de datos grandes para obtener información útil.

1.1. Objetivos del proyecto

Uno de los objetivos de este trabajo, con el que se empezará a trabajar en primer lugar, será el aprendizaje autónomo y desde cero del lenguaje de programación Python. Este no ha sido usado en la carrera, por lo que, mediante cursos y ejercicios, se intentará consolidar un nivel medio para poder llevar a cabo todos los retos que se nos propongan durante la realización del proyecto.

Una vez logrado esto, continuaremos con la parte más importante del trabajo. Se usarán distintas técnicas de minería de datos para crear nuestras bases de datos principales. En términos generales, se intentará extraer información de páginas web, analizar las letras de las canciones para generar nuevas variables de interés y agrupar países por su nivel de amistad a lo largo del tiempo.

A raíz de las bases de datos, el objetivo más atractivo será predecir las puntuaciones finales de los distintos festivales nacionales que estudiaremos y el gran festival de Eurovisión. Buscaremos desarrollar un modelo predictivo que pueda identificar patrones, tendencias y características clave de la información obtenida.

1.2. Metodología del proyecto

Tal como se menciona en los objetivos, se desea implementar las técnicas de minería de datos mediante el lenguaje Python. Para cada festival, seguiremos las siguientes fases:

Recogida de datos

Este proceso es fundamental para tomar decisiones informadas y precisas en cualquier campo, ya sea en investigación de mercado, la ciencia, la política y el deporte. En este proyecto tendremos un periodo de investigación referente a toda la información de los festivales nacionales, como las preselecciones de los concursantes, los sistemas de puntuaciones y los resultados que nos marcamos como objetivo.

Extracción de las cuotas de apuestas

Mediante *Web Scraping* [3] en una página, donde se encuentra la información principal de las apuestas, extraeremos las cuotas disponibles de todos los participantes para cada festival a estudiar. También se representará la probabilidad de ganar y los votos de la encuesta para el año actual.

Extracción de parámetros de Spotify

A partir de la creación de una app en la API [4] de Spotify para Desarrolladores, almacenaremos en una base de datos los parámetros que están detrás de cada canción.

[3] Técnica de extracción de datos de sitios web de manera automatizada.

[4] Conjunto de reglas y protocolos que permiten a diferentes aplicaciones interactuar y compartir datos entre sí.

Estos medirán características como la popularidad, la longevidad o la energía de todas las canciones para así poder sacar mejores predicciones.

Topic modelling de las letras de las canciones

A través de las *lyrics* (letras) de las canciones, obtenidas mediante *Web Scraping* de la API de Genius [5], haremos una depuración para lematizar y eliminar palabras *stopwords* [6], y así poder identificar y extraer los temas principales que se tratan en ellas. El objetivo es descubrir patrones y relaciones ocultas entre palabras y canciones, y agruparlas en temas coherentes y significativos. Esto nos permitirá dividir la base de datos según patrones de comportamiento en los gustos musicales nacionales.

Generación de un indicador de "amistad" para Eurovisión

A partir del histórico de resultados del festival de Eurovisión, crearemos un indicador que medirá la amistad que hay entre los países que participan en el festival de eurovisión. Este se sacará a partir de las puntuaciones anuales que se dan entre países, y los agrupará por comunidades. Además, nos permitirá sacar una animación del grafo, donde cada nodo es un país, y observaremos los conjuntos de países más afines durante el paso del tiempo.

Uso de metodología Machine y Deep Learning [7] para generar los modelos de predicción

Para poder predecir, juntaremos las bases de datos disponibles y crearemos distintos modelos de predicción de menor a mayor complejidad, para evaluar cuál predice mejor la puntuación y la posición final de los concursantes en cada festival. Primero usaremos un árbol de decisión, seguiremos con el XGBoost y terminaremos con una red neuronal.

1.3. Herramientas utilizadas

A continuación, se detallan las herramientas que se han utilizado para la realización de este proyecto.

1.3.1. Python

Python es un lenguaje de programación de alto nivel que se caracteriza por ser fácil de aprender y utilizar. Su sintaxis es clara y sencilla, lo que facilita el desarrollo de aplicaciones en diversos ámbitos. Es utilizado en una amplia variedad de aplicaciones, como la ciencia de datos, el aprendizaje automático, la automatización de tareas, el desarrollo web y de aplicaciones móviles, entre otros.

[5] Aplicación móvil de la plataforma Genius, que proporciona letras de canciones y explicaciones de su significado.
[6] Palabras comunes que se eliminan en el procesamiento de texto debido a su falta de relevancia, como "el", "la", "y", etc.
[7] El machine learning es una disciplina de la inteligencia artificial que permite a las máquinas aprender y mejorar a partir de datos, mientras que el deep learning es una subrama del machine learning que se enfoca en redes neuronales artificiales profundas para realizar tareas complejas de aprendizaje automático.

Una de las características más destacadas de Python es su filosofía "Zen de Python", que promueve la legibilidad del código y la simplicidad. Python también es conocido por su comunidad activa y solidaria, que desarrolla y mantiene numerosas librerías y proyectos de código abierto.

Algunas de las librerías más populares de Python son *NumPy*, Pandas, *Matplotlib*, *Scikit-Learn*, cuyas usaremos en este proyecto. Estas librerías son paquetes que contienen funciones para facilitar a la hora de crear nuevos sistemas.

La realización de este proyecto me permitirá aprender de cero uno de los programas más utilizados en el mundo y aplicar las técnicas de minería de datos no aprendidas en el grado.

1.3.2. Anaconda y Spyder

Anaconda y Spyder son dos programas muy utilizados en la ciencia y el análisis de datos en Python.

Anaconda cuenta con su propio gestor de paquetes, llamado "conda", que permite instalar y gestionar paquetes de Python y otros lenguajes de programación. Una de las características más destacadas de Anaconda es que permite la creación de ambientes virtuales, lo que facilita la gestión de las dependencias y versiones de las librerías utilizadas en cada proyecto. Esto es especialmente útil cuando se trabaja en múltiples proyectos o cuando se necesita trabajar con versiones específicas de las librerías.

Por otro lado, Spyder es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) especialmente diseñado para la ciencia de datos y el análisis de datos en Python. Spyder cuenta con numerosas herramientas y características para facilitar el desarrollo y la depuración de código en Python, como el explorador de variables, el visor de objetos, la consola interactiva y el autocompletado de código.

Spyder también cuenta con integración con otras herramientas populares en la ciencia de datos, como *NumPy*, *Pandas* y *Matplotlib*, lo que facilita el trabajo con datos y el análisis de los mismos. Además, Spyder es altamente personalizable y se puede extender con numerosos *plugins* y paquetes adicionales.

1.3.3. Librerías: NumPy, Pandas, Matplotlib y Sklearn

Las librerías *NumPy*, Pandas, *Matplotlib* y *Scikit-Learn* son cuatro de las herramientas más populares en el mundo de la ciencia de datos y el análisis de datos en Python.

NumPy es una librería que proporciona un conjunto de funciones para la manipulación de *arrays* y matrices de datos numéricos. *NumPy* proporciona un rendimiento eficiente y rápido para la manipulación y el procesamiento de grandes conjuntos de datos.

Pandas proporciona estructuras de datos flexibles y eficientes para el manejo de datos, como los *DataFrames* y las Series. Estas estructuras de datos permiten el procesamiento y análisis de grandes cantidades de datos de manera eficiente y fácil de entender.

Matplotlib es una librería de visualización de datos en Python que permite la creación de gráficos y visualizaciones de datos de alta calidad. *Matplotlib* es una librería muy flexible y personalizable, que permite la creación de gráficos en 2D y 3D, y su integración con otras librerías de análisis de datos como Pandas y *NumPy*.

Finalmente, *Scikit-Learn* es una librería de aprendizaje automático en Python, que proporciona herramientas para el modelado y la predicción de datos. *Scikit-Learn* incluye numerosos algoritmos de aprendizaje automático, como la regresión lineal, la regresión logística, los árboles de decisión y las redes neuronales, entre otros.

Estas cuatro librerías han sido las más utilizadas en el código de este proyecto.

1.3.4. APIs de Spotify y Genius

Las APIs [8] (Interfaces de Programación de Aplicaciones) de Spotify y Genius son herramientas útiles para la minería de datos en la industria de la música. Estas APIs permiten a los desarrolladores acceder a datos de música y letras de canciones de manera programática, lo que permite realizar análisis y descubrir patrones interesantes en los datos.

Por ejemplo, la API de Spotify permite acceder a información detallada sobre canciones, artistas, álbumes y listas de reproducción, como información de género, popularidad, duración, energía y mucho más. Esto permite a los desarrolladores analizar y comparar diferentes géneros y artistas, así como identificar las tendencias más populares en la música.

Por otro lado, la API de Genius proporciona acceso a una gran cantidad de letras de canciones, incluyendo información sobre el autor, el género y la popularidad de cada canción. Esto permite a los desarrolladores analizar la estructura de las letras de canciones, así como identificar temas recurrentes y patrones en el contenido de las letras.

Mediante estas dos interfaces, hemos extraído parámetros y textos muy valiosos de concursantes y canciones que nos han permitido crear bases de datos muy completas para poder jugar y predecir con ellas.

1.3.5. Librerías: Nltk, Googletrans y Gensim

Las librerías *NLTK*, *Googletrans* y *Gensim* son tres herramientas muy útiles en el campo del procesamiento de lenguaje natural con Python.

NLTK (*Natural Language Toolkit*) es una librería que proporciona una amplia variedad de herramientas y recursos para el análisis de texto, como tokenización, etiquetado gramatical, análisis sintáctico y semántico, entre otros. También ofrece recursos lingüísticos, como corpus y diccionarios.

[8] Conjunto de reglas y protocolos que permiten a diferentes aplicaciones interactuar y compartir datos entre sí.

Googletrans es una librería que permite realizar traducciones de texto utilizando la API de *Google Translate*. Es muy fácil de usar y soporta una amplia variedad de idiomas.

Gensim es una librería que se enfoca en el modelado de temas y la similitud semántica. Permite crear modelos de tema a partir de un corpus de texto y luego usarlos para realizar búsquedas de similitud semántica y encontrar temas relevantes en nuevos textos.

En conjunto, estas tres librerías son una poderosa combinación para el análisis de texto y la traducción automática en Python Spyder, lo que nos ha permitido, principalmente, hacer el *Topic Modelling* (detallado posteriormente) de cada canción a estudiar.

1.3.6. Librerías: TensorFlow y Keras

Las librerías *TensorFlow* y *Keras* son herramientas poderosas para el desarrollo de modelos de aprendizaje profundo en Python. Cada una de ellas ofrece distintas características y funcionalidades para la implementación de redes neuronales.

TensorFlow es una librería de código abierto desarrollada por Google, que ofrece una gran cantidad de herramientas y recursos para la construcción de modelos de aprendizaje profundo, incluyendo redes neuronales. Ofrece tanto una API de alto nivel para modelos predefinidos, como una API de bajo nivel para una mayor flexibilidad.

Keras es una librería de alto nivel que se utiliza en conjunto con *TensorFlow* y permite la creación de modelos de aprendizaje profundo con una sintaxis simple y fácil de usar. Tiene soporte para una amplia variedad de arquitecturas, incluyendo las redes neuronales.

En resumen, cada una de estas librerías ofrece distintas ventajas y características para la implementación de redes neuronales en Python. En este proyecto se han usado ambas para realizar las predicciones de los objetivos comentados anteriormente.

2. Creación del estado del arte

El estado del arte, también conocido como estado del conocimiento, será la revisión sistemática y crítica de la literatura existente en mi área temática o disciplina, con el objetivo de identificar y comprender el conocimiento actual, las tendencias, las brechas y las oportunidades de investigación en el área de interés.

En este proyecto, empezaremos con la investigación exhaustiva de los tres eventos claves para realizar el modelo de predicción. Expondremos sus definiciones y preselecciones, y acabaremos describiendo el sistema de puntuación y los resultados finales de estas. Esto último será nuestro foco principal al que se querrá llegar una vez hecha la predicción [9].

Finalmente, hablaremos en términos generales sobre las apuestas deportivas y el funcionamiento de una de nuestras variables más importantes: las cuotas[10].

2.1. Benidorm fest

El Benidorm Fest es un festival de la canción que tiene lugar en España organizado por la empresa pública de comunicación de Radiotelevisión Española (RTVE). Este concurso determina la canción del citado país para el Festival de la Canción de Eurovisión y se celebra desde el año 2022 en la localidad homónima. El evento consta de dos semifinales y una final, siendo el ganador elegido por el público y por equipos de jueces, ambas partes con la misma influencia en el resultado final.

2.1.1. Preselección española

La canción española para Eurovisión 2023 se seleccionó a través de Benidorm Fest 2023 el 4 de febrero. Previamente hubo una gala de presentación el 29 de enero y dos semifinales, celebradas los días 31 de enero y 2 de febrero. El concurso, presentado por Mónica Naranjo, Inés Hernand y Rodrigo Vázquez, se celebrará en el Palau Municipal d'Esports l'Illa de Benidorm.

El 1 de septiembre TVE abrió la convocatoria para que artistas, autores y compositores presentaran su propuesta a través de la página web de la cadena. El reglamento establecía que al menos el 50% de los intérpretes de cada propuesta debía tener nacionalidad española o ser residentes en el país. Los cantantes podían enviar únicamente una solicitud.

La convocatoria, que finalizó el 10 de octubre, recibió 876 propuestas. Un jurado de expertos redujo el número de participantes a 18, anunciados el 25 de octubre de 2022. El 2 de noviembre, en rueda de prensa, se anunció el título de las canciones y el 18 de diciembre se publicaron los temas. El 11 de enero de 2023, a través de una rueda de prensa, se dio a conocer la distribución de propuestas en cada semifinal. El orden de actuación sería por criterio operativo y televisivo.

[9] Proceso de estimar o prever un resultado futuro basado en información y datos disponibles.
[10] Valor numérico o porcentaje que representa la proporción o porción asignada a algo, como una cantidad a pagar, una participación en un reparto o una asignación de recursos.

Los vencedores de las tres galas serían elegidos a través de un sistema mixto de votación popular (25% televoto [11] y 25% jurado demoscópico [12]) y jurado profesional (nacional e internacional - 50%) que explicaremos detalladamente en el próximo apartado. El 25 de enero RTVE anunció a los ocho miembros del jurado profesional. En este había tanto personas influyentes en al ámbito musical de carácter nacional como internacional.

2.1.2. Sistema de puntuación

Este concurso determina la canción representante del citado país para el Festival de la Canción de Eurovisión y se celebra desde el año 2022 en la localidad homónima. El certamen consta de dos semifinales [13] y una final, siendo el ganador elegido por el público y por equipos de jueces, ambas partes con la misma influencia en el resultado final.

Para cada uno de los eventos, se diferencian los tres criterios de votación: el jurado profesional, y el jurado de la audiencia, donde tenemos el demoscópico y el televoto. En la sección de jurados, a su vez, se distingue un apartado para cada uno de los 8 que hay.

A continuación, observamos una cartilla vacía del funcionamiento del sistema de puntuación del Benidorm Fest. Según la posición que se le es asignada a cada concursante, se les otorga una cantidad de puntos y sumando los totales de los tres criterios, obtenemos la posición final.

JURADO																	AUDIENCIA					TOTAL	
JUR 1		JUR 2		JUR 3		JUR 4		JUR 5		JUR 6		JUR 7		JUR 8		TOTAL	DEMOSCÓPICO		TELEVOTO		TOTAL		
#	PTS	#	PTS	#	PTS	#	PTS	#	PTS	#	PTS	#	PTS	#	PTS		#	PTS	#	PTS			
	0		0		0		0		0		0		0		0	0		0		0	0	0	9
	0		0		0		0		0		0		0		0	0		0		0	0	0	8
	0		0		0		0		0		0		0		0	0		0		0	0	0	7
	0		0		0		0		0		0		0		0	0		0		0	0	0	6
	0		0		0		0		0		0		0		0	0		0		0	0	0	5
	0		0		0		0		0		0		0		0	0		0		0	0	0	4
	0		0		0		0		0		0		0		0	0		0		0	0	0	3
	0		0		0		0		0		0		0		0	0		0		0	0	0	2
	0		0		0		0		0		0		0		0	0		0		0	0	0	1

Figura 2.1. Ejemplo de cartilla del sistema de puntuación del Benidorm Fest 2023

Al inicio y al final de cada tabla se irán actualizando las posiciones de los intérpretes. En las semifinales se iluminarán en verde los 4 con mayor puntuación. En el caso de la final solo se iluminará el ganador. En caso de empates, gana el que mayor puntuación haya obtenido por parte del jurado profesional, tal y como se establece en las normes oficiales.

La metodología mantiene su esencia respecto al año pasado, con un 50% del peso para el jurado (30% nacional y 20% internacional) y el 50% restante dividido a partes iguales entre la muestra demoscópica de 350 individuos y el televoto de llamadas y SMS.

[11] Sistema de votación a distancia en el que los espectadores emiten sus votos o preferencias mediante medios de comunicación, como el teléfono o internet, en programas o competiciones televisivas.
[12] Grupo de personas seleccionadas de manera representativa para evaluar y emitir opiniones sobre un tema o producto, basándose en métodos y técnicas demoscópicas para obtener una visión más amplia y representativa de la opinión pública.
[13] Etapa o ronda de una competencia en la que se seleccionan los participantes o equipos que avanzarán a la final, generalmente a partir de una serie de eliminatorias previas.

Los componentes del jurado demoscópico provendrán de todas las Comunidades Autónomas de España, siguiendo un orden proporcional al número de habitantes. Además, la muestra se dividirá en dos grupos: género (hombre o mujer) y edad (de 18 a 44 años y de más 45).

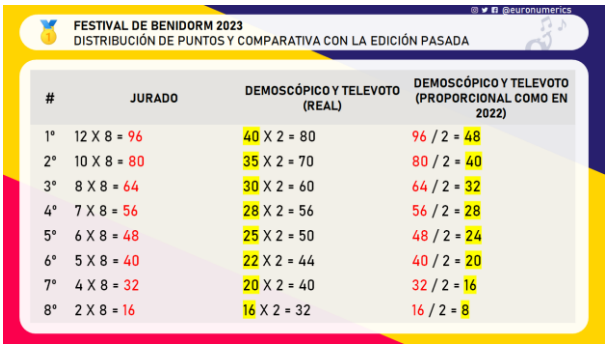
De este jurado quedarán excluidos las personas relacionadas con la industria discográfica, el sector televisivo, periodistas y profesionales de la publicidad o estudios de mercado. El cambio más importante ya se conocía de antes: en caso de empate en un puesto decisivo, decide el jurado.

Para la edición de 2022 los jurados repartían un set de puntos en función del número de participantes. Cada puntuación que otorgaban los jurados se multiplica por el número de jurados, 5, y se dividía entre dos, una mitad para el demoscópico y otra para el televoto. Por ejemplo, la máxima puntuación de un jurado son 12 puntos, multiplicado por 5 son 60. La mitad de cada uno, 30, era la máxima puntuación del demoscópico y del televoto.

Figura 2.2. Reparto de puntos entre jurados en el Benidorm Fest 2022

Sin embargo, este año no será así. Si hacemos cuentas, este año tenemos 8 jurados. Por tanto, se entregarán 8 veces 12 puntos, lo cual son 96 puntos. Si dividimos esta cantidad entre 2, el resultado es 48 puntos. Siguiendo el ejemplo del año pasado, televoto y demoscópico deberían dar un máximo de 48 puntos. En cambio, solo van a entregar 40 puntos. En un sistema que puede dar resultados muy estrechos, 16 puntos, 8 del televoto y 8 del demoscópico, son realmente decisivos.

Si nos fijamos en la puntuación más baja, los jurados darán como mínimo 2 puntos al participante que peor valoren. En mitades, televoto y demoscópico deberían dar 8 puntos cada uno y, sin embargo, según lo publicado otorgarán 16 puntos cada uno. Es decir, el doble de lo que deberían a la última posición. El desajuste ocurre tanto en la final como en semifinales. A continuación, vemos la otorgación de puntos de la final, donde tendremos un número de concursantes menor al de las semifinales, donde en el jurado no profesional (tanto demoscópico como televoto) tenemos una puntuación más baja que otorga 13 puntos.



#	JURADO	DEMOSCÓPICO Y TELEVOTO (REAL)	DEMOSCÓPICO Y TELEVOTO (PROPORCIONAL COMO EN 2022)
1º	12 X 8 = 96	40 X 2 = 80	96 / 2 = 48
2º	10 X 8 = 80	35 X 2 = 70	80 / 2 = 40
3º	8 X 8 = 64	30 X 2 = 60	64 / 2 = 32
4º	7 X 8 = 56	28 X 2 = 56	56 / 2 = 28
5º	6 X 8 = 48	25 X 2 = 50	48 / 2 = 24
6º	5 X 8 = 40	22 X 2 = 44	40 / 2 = 20
7º	4 X 8 = 32	20 X 2 = 40	32 / 2 = 16
8º	2 X 8 = 16	16 X 2 = 32	16 / 2 = 8

Figura 2.3. Comparativa entre el reparto de puntos entre las ediciones 2022 y 2023 (1)

En términos de rango, con el sistema publicado, tanto televoto como demoscópico otorgarán de 40 puntos máximo a 16 puntos mínimo, 24 puntos de diferencia entre uno y otro. Si se hubiese respetado el sistema del año pasado, la diferencia sería de los 48 de máximo a los 8 de mínimo, 40 puntos de distancia entre uno y otro.

Todo ello se traduce en que el jurado pueda dar un máximo de 96 puntos a un participante, y demoscópico más televoto solo 80 puntos, 40 y 40. En la zona baja, el jurado como poco dará 16 puntos a un participante, mientras que la mitad de la audiencia dará 26 puntos.

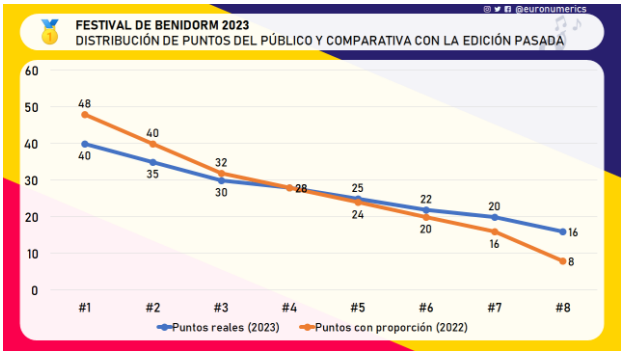


Figura 2.4. Comparativa entre el reparto de puntos entre las ediciones 2022 y 2023 (2)

En definitiva, estamos ante un sistema de puntos distinto que resta peso al criterio de la audiencia en favor del jurado.

A continuación, mostramos una posible tabla de semifinales para poder ver cómo funciona cada criterio y qué peso tienen en la puntuación final, como hemos explicado anteriormente:

JURADO																	AUDIENCIA					TOTAL	#
JUR 1		JUR 2		JUR 3		JUR 4		JUR 5		JUR 6		JUR 7		JUR 8		TOTAL	DEMOSCÓPICO		TELEVOTO		TOTAL		
#	PTS	#	PTS	#	PTS	#	PTS	#	PTS	#	PTS	#	PTS	#	PTS		#	PTS	#	PTS			
5	6	2	10	9	2	9	2	8	3	3	8	2	10	3	8	49	7	20	1	40	60	109	3
3	8	3	8	7	4	3	8	1	12	6	5	9	2	4	7	54	2	35	5	25	60	114	2
9	2	9	2	1	12	6	5	4	7	1	12	8	3	5	6	49	4	28	4	28	56	105	4
6	5	7	4	6	5	5	6	7	4	2	10	4	7	7	4	45	6	22	7	20	42	87	8
7	4	4	7	2	10	8	3	5	6	4	7	5	6	9	2	45	5	25	6	22	47	92	7
4	7	6	5	5	6	4	7	9	2	8	3	1	12	1	12	54	9	13	2	35	48	102	6
2	10	5	6	8	3	1	12	2	10	7	4	7	4	6	5	54	3	30	3	30	60	114	1
8	3	1	12	3	8	2	10	6	5	9	2	3	8	2	10	58	8	15	9	13	28	86	9
1	12	8	3	4	7	7	4	3	8	5	6	6	5	8	3	48	1	40	8	15	55	103	5

Tabla 2.1. Ejemplo de puntuaciones finales en la cartilla

Podemos ver remarcado en rojo la puntuación más alta que, por cada jurado es 12 (en total 96), y por el demoscópico y el televoto, son 40. Y en verde, como hemos dicho antes, aquellos que se clasificarían para la final ya que habrían sido los cuatro que han obtenido una puntuación más alta.

Para la final, como hemos dicho anteriormente, el sistema tiene el mismo funcionamiento, donde el concursante con mayor puntuación, gana el festival.

2.1.3. Resultados de los eventos

Benidorm Fest 2022

Posición	*Primera semifinal (31 de enero)*	*Segunda Semifinal (2 de febrero)*	*Final (4 de febrero)*
1	Chanel: 110 p	Rigoberta Bandini: 111 p	Chanel: 96 p
2	Tanxugueiras: 93 p	Rayden: 90 p	Rigoberta Bandini: 91 p
3	Blanca Paloma: 79 p	Xeinn: 81 p	Tanxugueiras: 90 p
4	Varry Brava: 74 p	Gonzalo Hermida: 76 p	Rayden: 67 p
5	Azúcar Moreno: 69 p	Marta Sango: 63 p	Blanca Paloma: 61 p
6	Unique: 87 p	Javiera Mena: 50 p	Varry Brava: 55 p
7		Sara Deop: 49 p	Xeinn: 45 p
8			Gonzalo Hermida: 35 p

Clasificados para la final Ganador/a p: puntos

Tabla 2.2. Resultados de los eventos del Benidorm Fest 2023

Benidorm Fest 2023

Posición	*Primera semifinal (31 de enero)*	*Segunda Semifinal (2 de febrero)*	*Final (4 de febrero)*
1	Agoney: 161 p	Blanca Paloma: 167 p	Blanca Paloma: 169 p
2	Alice Wonder: 119 p	Vicco: 135 p	Agoney: 145 p
3	Fusa Nocta: 118 p	Karmento: 112 p	Vicco: 129 p
4	Megara: 111 p	José Otero: 105 p	Megara: 106 p
5	Aritz Aren: 107 p	Alfred García: 102 p	Alice Wonder: 89 p
6	Sharonne: 87 p	Siderland: 88 p	Karmento: 80 p
7	Meler: 84 p	Famous: 87 p	José Otero: 75 p
8	Twin Melody: 77 p	E'Femme: 64 p	Fusa Nocta: 71 p
9	Sofía Martín: 48 p	Rakky Rypper: 52 p	

Clasificados para la final Ganador/a p: puntos

Tabla 2.3. Resultados de los eventos del Benidorm Fest 2023

Esta información la usaremos posteriormente como variables respuesta para realizar las predicciones. (Para conocer el nombre y el orden de salida de las canciones, se puede consultar el apartado de los anexos para su detalle, en la página 79)

2.2. Melodifestivalen

Melodifestivalen (literalmente, «El festival de la melodía») o Mello (conocido de esta manera en el propio país), es un festival de la canción organizado en Suecia por las empresas públicas de comunicación *Sveriges Television* (SVT) y *Sveriges Radio* (SR). Este concurso determina la canción representante del citado país para el Eurovisión y ha sido celebrado todos los años desde 1959. Desde 2000, es el programa de televisión más popular en Suecia.

2.2.1. Preselección sueca

La canción sueca para Eurovisión 2023 se seleccionará el 11 de marzo a través del Melodifestivalen 2023. Previamente, entre el 4 de febrero y 4 de marzo, habrá cinco semifinales.

La televisión pública sueca, SVT, abrió el 26 de agosto de 2022 la convocatoria para su preselección, que se alargó hasta el 16 de septiembre. El reglamento estipulaba que al menos el 50% de las canciones participantes debían estar compuestas por mujeres, y que al menos el 30% de los temas debían ser interpretados en sueco. La cadena recibió sobre 2.824 canciones. De ellas, un jurado de expertos eligió a 14, mientras que la SVT invitó, de forma directa, a otros 14 artistas y compositores a formar parte del formato. Los participantes de las dos primeras semifinales fueron anunciados el 29 de noviembre, mientras que los intérpretes y canciones restantes fueron anunciados el 30 de noviembre.

La fase televisada del Melodifestivalen dura cinco semanas y consiste de seis espectáculos en vivo: cuatro semifinales con siete canciones en cada una; una ronda de repesca [14] llamada *Andra Chansen*, con canciones que obtuvieron el tercer y cuarto lugar en las semifinales; y la gran final que la componen las mejores dos canciones de cada semifinal (un total de ocho) y las cuatro provenientes de la repesca, en total 12.

2.2.2. Sistema de puntuación

El sistema de votación utilizado en las semifinales estará formado exclusivamente por el televoto (proporción peso 1/8) y la aplicación oficial del Melodifestivalen (proporción peso 7/8), disponible solamente en Suecia. Al registrarse, el votante tendrá que detallar su edad, para que su votación se incluya en uno de los siete grupos definidos por la SVT: 3-9 años, 10-15 años, 16-29 años, 30-44 años, 45-59 años, 60-74 años y más de 75 años. Cada espectador podrá votar cinco veces. El televoto y cada uno de los grupos de edad otorgará sets de 1, 2, 4, 6, 8, 10 y 12 puntos.

- *Semifinales (giras)*

El sistema en las semifinales utiliza únicamente el televoto. Así pues, se muestra durante todas las canciones los respectivos números de teléfono para fijo y móvil y SMS.

[14] Proceso o etapa adicional en una competencia en la que los participantes o equipos que no lograron clasificar en una fase anterior tienen una oportunidad de competir nuevamente para intentar avanzar a la siguiente fase o ronda.

Entonces, mientras hay una actuación en directo, aparece un corazón palpitando que indica la cantidad de votos que recibe esa canción. A más pulsaciones por minuto más votos está recibiendo.

Cuando han sido realizadas todas las actuaciones, se hace el último repaso a las canciones con un resumen de todas ellas y se anuncian los resultados. Únicamente la más votada pasa directamente a la gran final, las otras seis canciones van a la segunda ronda de la semifinal.

Tras anunciar las seis canciones que pasan a la segunda ronda, se vuelve a hacer el resumen de las actuaciones con las líneas abiertas para volver a votar durante unos pocos minutos. Una vez finalizado el segundo tiempo de votación: primero se anuncia una finalista directa que repetirá la actuación, y después dos canciones que pasan a la gala de repesca.

Este funcionamiento también se aplicó para la edición del 2022

- *Andra Chansen (Segunda oportunidad)*

En la gala de segunda oportunidad, el sistema de votación difiere un poco respecto al año anterior, aunque el procedimiento sea casi idéntico. Para este año siguieron el mismo funcionamiento que en las semifinales habiendo solo una ronda de votación 100% del público y donde pasaban las cuatro canciones con mayor puntuación.

En 2022 se dividieron los temas en dos grupos de cuatro participantes cada uno. Dentro de cada grupo, los participantes se sometían al mismo sistema que al año actual y las dos canciones con mayor puntuación para cada grupo, eran las que pasaban a la final.

- *Final*

El sistema actual de votación es similar al usado en el Festival de Eurovisión. Una serie de jurados, antiguamente de distintas regiones de Suecia, y en la actualidad jurados de diversos países de Europa, votan de 1 a 12 puntos a sus 7 canciones favoritas. El televoto hasta 2010 tenía una influencia equivalente; 11, 22, 44, 66, 88, 110, y 132 puntos se otorgaban a las siete canciones más votadas. La canción con la mayor puntuación al final de los dos resultados era la ganadora.

Desde la edición de 2011 se cambió la forma de puntuar del televoto. Sumando todos los puntos que repartirían en total el jurado, que siguen puntuando como antes, se tiene que deben sumar 473 puntos y el repartirá de forma proporcional al número de llamadas que haya tenido cada artista. Por ejemplo, si un artista se lleva el 10% de los votos, se llevaría el 10% de esos 473 puntos, unos 47 puntos (los decimales siempre se redondean.

A diferencia del sistema anterior, todas las canciones reciben puntos acordes con el número de llamadas recibidas, y no solo las siete más votadas. Se emplean dos números telefónicos donde también se puede votar vía SMS. En caso de empate, la canción que reciba más votos del público es la ganadora.

A continuación, vemos cómo funciona el sistema de puntuación en una de las semifinales de este año:

	Intérprete/s	Canción	Ronda 1
01	Tone Sekelius	*«Rhythm Of My Show»*	¿?
02	Loulou LaMotte	*«Inga Sorger»*	¿?
03	Rejhan	*«Haunted»*	¿?
04	Elov & Beny	*«Raggen Går»*	¿?
05	Victor Crone	*«Diamonds»*	¿?
06	Eva Rydberg & Ewa Roos	*«Länge Leve Livet»*	¿?
07	Jon Henrik Fjällgren, Arc north feat. Adam Woods	*«Where Are You (Sávečan)»*	1º

Tabla 2.4. Resultado de la primera ronda de la 1ª semifinal del Melodifestivalen 2023

De la primera ronda se extrae que, la canción subrayada de amarillo, es la que pasa a la final ya que ha sido la más votada durante la actuación de esta.

	Canción	3-9	10-15	16-29	30-44	45-59	60-74	75+	Telev.	Total	Puesto
01	*«Rhythm Of My Show»*	12	12	12	12	12	12	10	12	94	1º
02	*«Inga Sorger»*	1	1	1	5	8	8	5	3	32	5º
03	*«Haunted»*	5	8	10	3	3	1	1	1	32	4º
04	*«Raggen Går»*	8	3	3	8	5	5	3	8	43	3º
05	*«Diamonds»*	10	10	8	10	10	10	12	10	80	2º
06	*«Länge Leve Livet»*	3	5	5	1	1	3	8	5	31	6º

Tabla 2.5. Resultado de la segunda ronda de la 1ª semifinal del Melodifestivalen 2023

A partir de la segunda ronda, vemos que la primera canción con más puntuación pasa también a la gran final, y las subrayadas en verde son las que pasan a la repesca/quina semifinal (*Andra Chansen*). Como hemos explicado antes, la repesca tendrá el mismo sistema, pero solo será una sola ronda con el segundo formato de las semifinales previas. Las cuatro canciones con puntuación más alta pasan a la final.

A continuación, vemos cómo funciona la final:

Artista	Canción	Jurado	Televoto	TOTAL
Loreen	*«Tattoo»*	92	85	**177**
Marcus & Martinus	*«Air»*	71	67	**138**
Smash Into Pieces	*«Six Feet Under»*	53	59	**112**

Tabla 2.6. Resultado de las 3 primeras posiciones de la final del Melodifestivalen 2023

Como se puede observar en los resultados de la final y reafirmando lo comentado anteriormente, la puntuación total se compone de los puntos del jurado y del televoto. En este caso, la mayor puntuación gana el festival.

2.2.3. Resultados de los eventos

Melodifestivalen 2022

Posición	*Primera Semifinal ** *(5 de febrero)*	*Segunda semifinal* *(12 de febrero)*	*Tercera semifinal* *(19 de febrero)*	*Cuarta semifinal* *(26 de febrero)*	*Quinta semifinal - Andra Chansen* *(5 de marzo)*
1	Cornelia Jakobs: CPR	Liamoo: CPR	Anders Bagge: CPR	Klara Hammarström: CPR	Tone Sekelius: 86 p
2	Robin Bengtsson: 12 p	Niello & Lisa Ajax: 92 p	Faith Kakembo: 79 p	Medina: 84 p	Anna Bergendahl: 77 p
3	Danne Stråhed: 10 p	Samira Manners: 69 p	Lisa Miskovsky: 75 p	Anna Bergendahl: 73 p	Cazzi Opeia: 76 p
4	Theoz: 8 p	Álvaro Estrella: 62 p	Cazzi Opeia: 67	Lillasyster: 65	Theoz: 71 p
5	Omar Rudberg: 5 p	Browsing Collection: 35 p	Linda Bengtzing: 37 p	Angelino: 37 p	Álvaro Estrella: 71 p
6	Shirley Camp: 3 p	John Lundvik: 33 p	Lancelot: 35 p	Tenori: 37 p	Lisa Miskovsky: 28 p
7	Malou Prytz: 1 p	Tone Sekelius: 21 p	Tribe Friday: 19 p	Malin Christin: 16 p	Lillasyster: 22
8					Danne Stråhed: 18 p

Clasificados para la final 1ª ronda Clasificados para la final 2ª ronda

Clasificados para la repesca Clasificados para la final p: puntos

Tabla 2.7. Resultados de las semifinales del Melodifestivalen 2022

*Cabe destacar que, en la segunda ronda, tras un fallo técnico en la aplicación de votación, no pudieron ser contabilizados los votos de los grupos de edad, definiendo los resultados a través de solamente las llamadas telefónicas. En el tratamiento de datos se plasmará el método usado para contrarrestar este hecho.

Posición	*Final (12 de marzo)*
1	Cornelia Jakobs: 146 puntos
2	Anders Bagge: 121 puntos
3	Medina: 109 puntos
4	Liamoo: 91 puntos
5	Tone Sekelius: 84 puntos
6	Klara Hammarström: 83 puntos
7	Theoz: 65 puntos
8	John Lundvik: 60 puntos
9	Cazzi Opeia: 55 puntos
10	Faith Kakembo: 51 puntos
11	Robin Bengtsson: 34 puntos
12	Anna Bergendahl: 29 puntos

Ganador/a

Tabla 2.8. Resultados de la final del Melodifestivalen 2022

Melodifestivalen 2023

Posición	*Primera semifinal (4 de febrero)*	*Segunda semifinal (11 de febrero)*	*Tercera semifinal (18 de febrero)*	*Cuarta semifinal (25 de febrero)*	*Quinta semifinal - Andra Chansen (4 de marzo)*
1	Fjällgren, Arc north ft. Adam Woods: CPR	Maria sur: CPR	Marcus & Martinus: CPR	Loreen: CPR	Nordman: 75 p
2	Tone Selkelius: 94 p	Panetoz: 84 p	Paul Rey: 85 p	Smash Into Pieces: 79 p	Theoz: 70
3	Victor Crone: 80 p	Theoz: 78 p	Nordman: 75 p	Kiana: 69 p	Kiana: 69 p
4	Elov & Beny: 43 p	Tennessee tears: 69 p	Melanie Wehbe: 57	Mariette: 56	Mariette: 63 p
5	Loulou LaMotte: 32 p	Wiktoria: 47 p	Ida-Lova: 35 p	Axel Schylström: 55 p	Tennessee tears: 55 p
6	Rejhan: 32 p	Uje Brandelius: 22 p	Laurell: 30 p	Signe & Hjördis: 27 p	Victor Crone: 28 p
7	Eva Rydberg & Ewa Roos: 31 p	Eden: 12 p	Casanovas: 29 p	Emil Henrohn: 26 p	Melanie Wehbe: 22
8					Elov & Beny: 18 p

Clasificados para la final 1ª ronda (CPR) Clasificados para la final 2ª ronda

Clasificados para la repesca Clasificados para la final p: puntos

Tabla 2.9. Resultados de las semifinales del Melodifestivalen 2023

Posición	*Final (11 de marzo)*
1	Loreen: 177 puntos
2	Marcus & Martinus: 138 puntos
3	Smash Into Pieces: 112 puntos
4	Jon Henrik Fjällgren, Arc north ft. Adam Woods: 81 puntos
5	Theoz: 78 puntos
6	Kiana: 76 puntos
7	Paul Rey: 57 puntos
8	Mariette: 51 puntos
9	Maria sur: 47 puntos
10	Panetoz: 47 puntos
11	Nordman: 44 puntos
12	Tone Selkelius: 20 puntos

Ganador/a

Tabla 2.10. Resultados de la final del Melodifestivalen 2023

Esta información la usaremos posteriormente como variables respuesta para realizar las predicciones. (Para conocer el nombre y el orden de salida de las canciones, se puede consultar el apartado de los anexos para su detalle, en la página 80)

2.3. Eurovisión

El Festival de la Canción de Eurovisión (en francés: *Concours Eurovision de la Chanson*; en inglés: *Eurovision Song Contest*) es un concurso televisivo de carácter anual, en el que participan intérpretes representantes de las televisiones (en su mayoría públicas) cuyos países son miembros de la Unión Europea de Radiofusión.

El festival ha sido transmitido desde 1956, siendo el programa de televisión más antiguo que aún se transmite en el mundo, recibiendo en 2015 el récord Guinness como la competición musical televisiva más longeva del mundo y captando entre 200 y 600 millones de audiencia a nivel internacional. El festival es históricamente conocido por ser un gran promotor de la música pop. Sin embargo, en años reciente se han presentado en el festival varios temas pertenecientes a otros géneros como dance, tango, reguetón, heavy metal, pop rap, punk, electrónica y rumba, entre otros.

2.3.1. Participación

Los países aptos para participar son los miembros activos (a diferencia de los miembros asociados) de la Unión Europea de Radiodifusión. Los miembros activos son aquellos países que estén dentro del Área de Radiodifusión Europea (que incluye países que no son europeos) o los que pertenecen al Consejo de Europa, siempre que hayan solicitado su ingreso en la UER y cumplan con todos los requisitos para permanecer como miembros activos.

La participación histórica es la siguiente:

Figura 2.5. Participaciones desde 1956 en Eurovisión

Cada país debe inscribir solo una canción para que lo represente cada año que deseen participar. Existe una regla que prohíbe que el tema inscrito haya sido previamente publicado o transmitido en público antes de cierta fecha relativa al festival en cuestión. Esto asegura que solo haya canciones nuevas y elimina cualquier tipo de ventaja a posibles canciones famosas.

2.3.2. Sistema de puntuación

Hay dos cambios importantes en la edición de Eurovisión 2023 respecto a los años previos. En primer lugar, los telespectadores serán los únicos que decidan qué países pasan de las semifinales a la Gran Final. En esta decisión, antes también influía el voto del jurado experto. Este jurado seguirá emitiendo sus votos, pero solo se utilizarán en caso de que no se registre o no sea posible un televoto válido en un país.

Eso sí, para la Gran Final las reglas no cambian: el voto del jurado experto seguirá jugando un papel importante a la hora de decantar la balanza, como había ocurrido hasta ahora. El festival defiende que sea así por cuatro razones:

- Para garantizar mejor la clasificación "cualitativa" de los participantes.
- Para continuar con la tradición de conectar en directo con los portavoces de cada país.
- Para mitigar el voto "cultural" y de "diáspora" (sin influencias de razas o etnias).
- Para "mantener la emoción" de que haya dos votaciones y que el resultado se conozca al final de la gala.

El segundo cambio es que los telespectadores de países no participantes podrán participar en las votaciones de este año. Por lo que todos ellos, vivan donde vivan en el mundo, podrán votar por sus canciones favoritas.

2.3.3. Resultados de los eventos

Eurovisión 2021

Posición	*Primera Semifinal (18 de mayo)*	*Segunda Semifinal (20 de mayo)*
1	Malta con Destiny: 325 puntos	Suiza con Gjon's Tears: 291 puntos
2	Ucrania con Go_A: 267 puntos	Islandia con Daði Freyr Pétursson: 288 puntos
3	Rusia con Manizha: 225 puntos	Bulgaria con Victoria: 250 puntos
4	Lituania con The Roop: 203 puntos	Portugal con The Black Mamba: 239 puntos
5	Israel con Eden Alene: 192 puntos	Finlandia con Blind Chanel: 234 puntos
6	Chipre con Elena Tsagrinou: 170 puntos	Grecia con Stefania: 184 puntos
7	Suecia con Tusse: 142 puntos	Moldavia con Natalia Gordienko: 179 puntos
8	Azerbaiyán con Efendi: 138 puntos	Serbia con Hurricane: 124 puntos
9	Bélgica con Hooverphonic: 117 puntos	San Marino con Senhit: 118 puntos
10	Noruega con Tix: 115 puntos	Albania con Anxhela Peristeri: 112 puntos
11	Croacia con Albina: 110 puntos	Dinamarca con Fyr og Flamme: 89 puntos
12	Romania con Roxen: 85 puntos	Austria con Vincent Bueno: 66 puntos
13	Eslovenia con Ana Soklic: 44 puntos	Estonia con Uku Suviste: 58 puntos
14	Australia con Montaigne: 28 puntos	Polonia con Rafal: 35 puntos
15	Macedonia del Norte con Vasil: 23 puntos	Chequia con Benny Cristo: 23 puntos
16	Irlanda con Lesley Roy: 20 puntos	Georgia con Tornike Kipiani: 16 puntos
17		Letonia con Samanta Tina: 14 puntos

Clasificados para la final

Tabla 2.11. Resultados de la primera y segunda semifinal de Eurovisión 2021

	Final (22 de mayo)		
Posición		**Posición**	
1	Italia con Måneskin: 524 puntos	*14*	Suecia con Tusse: 109 puntos
2	Francia con Barbara Pravi: 499 puntos	*15*	Serbia con Hurricane: 102 puntos
3	Suiza con Gjon's Tears: 432 puntos	*16*	Chipre con Elena Tsagrinou: 94 puntos
4	Islandia con Daði Freyr Pétursson: 378 puntos	*17*	Israel con Eden Alene: 93 puntos
5	Ucrania con Go_A: 364 puntos	*18*	Noruega con Tix: 75 puntos
6	Finlandia con Blind Chanel: 301 puntos	*19*	Bélgica con Hooverphonic: 74 puntos
7	Malta con Destiny: 255 puntos	*20*	Azerbaiyán con Efendi: 65 puntos
8	Lituania con The Roop: 220 puntos	*21*	Albania con Anxhela Peristeri: 57 puntos
9	Rusia con Manizha: 204 puntos	*22*	San Marino con Senhit: 50 puntos
10	Grecia con Stefania: 170 puntos	*23*	Países Bajos con Jeangu Macrooy: 11 puntos
11	Bulgaria con Victoria: 170 puntos	*24*	España con Blas Cantó: 6 puntos
12	Portugal con The Black Mamba: 153 puntos	*25*	Alemania con Jendrik: 3 puntos
13	Moldavia con Natalia Gordienko: 115 puntos	26	Reino Unido con James Newman: 0 puntos
			Ganador/a

Tabla 2.12. Resultados de la final de Eurovisión 2021

Eurovisión 2022

Posición	*Primera Semifinal (10 de mayo)*	*Segunda Semifinal (12 de mayo)*
1	Ucrania con Kalush Orchestra: 337 puntos	Suecia con Cornelia Jakobs: 396 puntos
2	Países Bajos con S10: 221 puntos	Australia con Sheldon Riley: 243 puntos
3	Grecia con Amanda Georgiadi: 211 puntos	Serbia con Konstrakta: 237 puntos
4	Portugal con Maro: 208 puntos	República Checa con We Are Domi: 227 puntos
5	Armenia con Rosa Linn: 187 puntos	Estonia con Stefan: 209 puntos
6	Noruega con Subwoolfer: 177 puntos	Polonia con Ochman: 198 puntos
7	Lituania con Monika Liu: 159 puntos	Finlandia con The Rasmus: 162 puntos
8	Moldavia con Zdob & Fratii: 154 puntos	Bélgica con Jérémie Makiese: 151 puntos
9	Suiza con Marius Bear: 118 puntos	Rumania con WRS: 118 puntos
10	Islandia con Systur: 103 puntos	Azerbaiyán con Nadir Rüstemli: 96 puntos
11	Croacia con Mia Dimšić: 75 puntos	Macedonia del Norte con Andrea: 76 puntos
12	Albania con Ronela Hajati: 58 puntos	Chipre con Andromache: 63 puntos
13	Letonia con Citi Zēni: 55 puntos	Israel con Michael Ben David: 61 puntos
14	Dinamarca con Reddi: 55 puntos	San Marino con Achille Lauro: 50 puntos
15	Austria con Lumix: 42 puntos	Malta con Emma Muscat: 47 puntos
16	Bulgaria con Intelligent Music Project: 29 puntos	Irlanda con Brooke: 47 puntos
17	Eslovenia con LPS: 15 puntos	Montenegro con Vladana: 33 puntos
18		Georgia con Circus Mircus: 22 puntos
		Clasificados para la final

Tabla 2.13. Resultados de la primera y segunda semifinal de Eurovisión 2022

Final (14 de mayo)			
Posición		**Posición**	
1	Ucrania con Kalush Orchestra: 631 puntos	*14*	Lituania con Monika Liu: 128 puntos
2	Reino Unido con Sam Ryder: 466 puntos	*15*	Australia con Sheldon Riley: 125 puntos
3	España con Chanel: 459 puntos	*16*	Azerbaiyán con Nadir Rüstemli: 106 puntos
4	Suecia con Cornelia Jakobs: 438 puntos	*17*	Suiza con Marius Bear: 78 puntos
5	Serbia con Konstrakta: 312 puntos	*18*	Rumania con WRS: 65 puntos
6	Italia con Mahmood & Blanco: 268 puntos	*19*	Bélgica con Jérémie Makiese: 64 puntos
7	Moldavia con Zdob & Fratii: 253 puntos	*20*	Armenia con Rosa Linn: 61 puntos
8	Grecia con Amanda Georgiadi: 215 puntos	*21*	Finlandia con The Rasmus: 38 puntos
9	Portugal con Maro: 207 puntos	*22*	República Checa con We Are Domi: 38 puntos
10	Noruega con Subwoolfer: 182 puntos	*23*	Islandia con Systur: 20 puntos
11	Países Bajos con S10: 171 puntos	*24*	Francia con Alvan & Ahez: 17 puntos
12	Polonia con Ochman: 151 puntos	*25*	Alemania con Malik Harris: 6 puntos
13	Estonia con Stefan: 141 puntos		

Ganador/a

Tabla 2.14. Resultados de la final de Eurovisión 2022

Eurovisión 2023

Posición	*Primera Semifinal (9 de mayo)*	*Segunda Semifinal (11 de mayo)*
1	Finlandia con Käärijä: 177 puntos	Australia con Voyager: 149 puntos
2	Suecia con Loreen: 135 puntos	Austria con Teya & Salena: 137 puntos
3	Israel con Noa Kirel: 127 puntos	Polonia con Blanka: 124 puntos
4	Chequia con Vesna: 110 puntos	Lituania con Monika: 110 puntos
5	Moldavia con Pasha Parfeny: 109 puntos	Eslovenia con Joker Out: 103 puntos
6	Noruega con Alessandra: 102 puntos	Armenia con Brunette: 99 puntos
7	Suiza con Remo Forrer: 97 puntos	Chipre con Andrew Lambrou: 94 puntos
8	Croacia con Let 3: 76 puntos	Bélgica con Gustaph: 90 puntos
9	Portugal con Mimicat: 74 puntos	Albania con Albina & Familja Kelmendi: 83 puntos
10	Serbia con Luke Black: 37 puntos	Estonia con Alika: 74 puntos
11	Letonia con Sudden Lights: 34 puntos	Islandia con Diljá: 44 puntos
12	Irlanda con Wild Youth: 10 puntos	Georgia con Iru: 33 puntos
13	Países Bajos con Mia y Dion: 7 puntos	Grecia con Victor Vernicos: 14 puntos
14	Azerbaiyán con TuralTuranX: 4 puntos	Dinamarca con Reiley: 6 puntos
15	Malta con The Busker: 3 puntos	Rumanía con Theodor Andrei: 0 puntos
16		San Marino con Piqued Jacks: 0 puntos

Clasificados para la final

Tabla 2.15. Resultados de la primera y segunda semifinal de Eurovisión 2023

Final (13 de mayo)

Posición		**Posición**	
1	Suecia con Loreen: 583 puntos	*14*	Armenia con Brunette: 122 puntos
2	Finlandia con Käärijä: 526 puntos	*15*	Austria con Teya & Salena: 120 puntos
3	Israel con Noa Kirel: 362 puntos	*16*	Francia con La Zarra: 104 puntos
4	Italia con Marco Mengoni: 350 puntos	*17*	España con Blanca Paloma: 100 puntos
5	Noruega con Alessandra: 268 puntos	*18*	Moldavia con Pasha Parfeny: 96 puntos
6	Ucrania con Tvorchi: 243 puntos	*19*	Polonia con Blanka: 93 puntos
7	Bélgica con Gustaph: 182 puntos	*20*	Suiza con Remo Forrer: 92 puntos
8	Estonia con Alika: 168 puntos	*21*	Eslovenia con Joker Out: 78 puntos
9	Australia con Voyager: 151 puntos	*22*	Albania con Albina & Familja Kelmendi: 76 puntos
10	Chequia con Vesna: 129 puntos	*23*	Portugal con Mimicat: 59 puntos
11	Lituania con Monika: 127 puntos	*24*	Serbia con Luke Black: 30 puntos
12	Chipre con Andrew Lambrou: 126 puntos	*25*	Reino Unido con Mae Muller: 24 puntos
13	Croacia con Let 3: 123 puntos	*26*	Alemania con Lord Of The Lost: 18 puntos

Ganador/a

Tabla 2.16. Resultados de la final de Eurovisión 2023

Esta información la usaremos posteriormente como variables respuesta para realizar las predicciones. (Para conocer el nombre y el orden de salida de las canciones, se puede consultar el apartado de los anexos para su detalle, en la página 82)

2.4. Apuestas deportivas

La apuesta deportiva es una modalidad de apuesta en la que se intenta predecir los resultados de una competición deportiva. La legalidad, la regulación y la aceptación general de estas apuestas deportivas varía de país a país. El país pionero en las apuestas de todo tipo, sobre todo carreras de caballos y carreras de galgos ha sido el Reino Unido. Son muy populares las apuestas de boxeo profesional en algunas ciudades de Estados Unidos.

Desde el año 2002 se está llevando a cabo toda una expansión de casas de apuestas virtuales, centradas en apuestas deportivas, que está llegando a países de todos los continentes, y abarca numerosas disciplinas deportivas, como nuestros festivales de la canción.

Aunque las apuestas no tienen por qué ser un factor determinante y definitivo para ganar Eurovisión 2023, si nos dan una pista de quién podría alzarse con el micrófono de cristal en esta 67ª edición del festival.

2.4.1. Tipos de apuestas deportivas

A continuación, se detalla la única manera de apostar que permiten las casas de apuestas y en lo que nos basaremos.

- Apuesta Simple: Es la más utilizada y clásica. Si la persona acierta el pronóstico se le devuelve el importe más los beneficios. En caso contrario la casa de apuesta se queda con todo el dinero.

A partir de esta, se definen los subtipos que ya dependen del enfoque de la apuesta que se quiera hacer:

- Apuesta a Largo Plazo: Se utiliza cuando el evento dura varios días. Los tres eventos duran como mínimo una semana.
- Apuesta en Directo: El apostador puede modificar su apuesta en el transcurso del evento. Las casas de apuestas permiten cierran las apuestas una vez se otorgan los puntos en cada evento
- Apuesta Especial: Apuesta que está relacionada con los equipos que se enfrentan, pero no con el evento en sí. Se puede apostar directamente sobre los concursantes ya sea su posición o sus puntos finales tanto en semifinales como en finales.

El siguiente tipo no está disponible para esta clase de eventos deportivos:

- Apuesta Combinada: Aquí se hacen varias apuestas para varios eventos deportivos y hay que acertarlos todos para poder ganar la apuesta. Las casas de apuestas no permiten apostar a más de una opción al tratarse de eventos deportivos especiales.

Dicho esto, en el próximo apartado, solo hablaremos de cuotas en apuestas simples.

2.4.2. Cuota

La cuota es, posiblemente, la palabra más importante en el mundo de las apuestas deportivas. Se trata del indicador que muestra, de forma numérica, en cuánto se valora nuestra apuesta y, por tanto, cuál va a ser nuestra ganancia. Todas las apuestas tienen una cuota asociada. Esa cifra es por la que nosotros apostamos y la que conseguiremos en el caso de acertar nuestro pronóstico.

En estadística, la cuota es la proporción entre dos probabilidades complementarias. La cuota a favor de un evento o proposición se calcula mediante la siguiente fórmula:

$$\frac{p}{(1-p)}$$

dónde p es la probabilidad del evento o proposición. La cuota en contra del mismo evento se calcula mediante la fórmula:

$$\frac{(1-p)}{p}$$

2.4.2.1. Tipos de cuotas

En las apuestas deportivas en el ámbito mundial, existen tres tipos de cuotas diferenciales: las decimales (o europeas), las fraccionadas (o inglesas) y las americanas. Cada una se muestra de una forma distinta a la hora de calcular las ganancias, pero nosotros describiremos solo la decimal ya que se extraerán los datos de festivales europeos.

- Cuota decimal

En España se trabaja principalmente con un tipo de cuota: es la cuota decimal, también denominada cuota europea. Cada mercado disponible tiene asociado un número entero o con decimales que es el que nos marca cuánto vamos a ganar.

La fórmula para saber la cantidad que recibiremos es multiplicar la cuota por el dinero que apostemos. Lo vemos con un ejemplo de apuesta de 10 euros:

Cuota 1,20: 1.20 x 10 = 12 euros (2 euros ganancia neta)

Cuota 2: 2 x 10 = 20 euros (10 euros ganancia neta)

Cuota 5: 5 x 10 = 50 euros (40 euros ganancia neta)

2.4.3. Eurovision World

Para extraer las cuotas decimales para cada concursante de cada evento, usaremos una de las principales fuentes de información sobre el mayor festival europeo de la canción: Eurovision *World*.

Esta es un sitio web con sede en Europa que brinda a los amantes de la música y de Eurovisión una amplia gama de información sobre el concurso. Es una fuente confiable de noticias y análisis detallados sobre el evento musical más importante de Europa. Puedes encontrar desde artículos, hechos y encuestas hasta fotografías y videos de toda la historia del festival.

A partir de esta web, se extraerán los datos de las cuotas y los votos públicos de la encuesta pública que se hace a unas semanas antes de cada evento. El procedimiento para hacerlo se explicará en el siguiente capítulo.

Figura 2.6. Logo de la página web de Eurovisión World

3. Descripción de los datos

En este proyecto no hemos querido ir a lo seguro y coger una base de datos ya formada para conseguir nuestros objetivos. Como se ha dicho en un inicio, tenemos como finalidad principal la creación de distintas bases de datos mediante técnicas de minería de datos que no se han visto en la carrera.

Para cada festival nacional y Eurovisión usaremos los siguientes métodos:

1- Un *Web Scraping* de las probabilidades, cuotas y votos de cada concursante en la web de *Eurovision World*.
2- Una extracción de los parámetros musicales de la API de Spotify para cada canción.
3- Topic Modelling de las letras de las canciones.
4- Indicador de amistad entre países solo para Eurovisión.
5- Variables respuesta o de estudio para cada festival

La finalidad de estos procesos es acabar creando, des de cero, una base de datos muy amplia con información de todo tipo, para conseguir hacer un modelo para cada evento musical y predecir [15]ganadores y puntuaciones finales.

3.1. Datos iniciales

A continuación, haremos una descripción de las diferentes bases de datos que hemos creado en el proceso de recogida para realizar los modelos de todos los festivales de estudio. Para Eurovisión tendremos otra más que también especificaremos al final del capítulo.

Los distintos tipos de variables que veremos a continuación son los siguientes:

- Numérica (N): es aquella que representa números y con ella se puede realizar operaciones aritméticas.
- Categórica (C): es aquella que permite clasificar una serie de datos por medio de valores fijos asociados a una cualidad o categoría concreta.
- *DateTime* o Fecha (F): es aquella que permite almacenar información sobre cualquier fecha y hora.
- *Dummies* (D): es aquella variable ficticia o artificial que empleamos para representar un atributo con dos o más niveles o categorías diferentes.

Para poder hacer la predicción, necesitaremos la misma información de los festivales nacionales tanto para el año 2022 como para el 2023. Para Eurovisión también dispondremos de datos del 2021 para realizar una mejor predicción. Este paso es necesario ya que implementaremos redes neuronales a partir de las bases de datos. Como ya se explicará más adelante, este tipo de modelos se basan en hechos ocurridos hace tiempo, por lo que nos será de utilidad recoger también la información de 2022 para poder predecir correctamente.

[15] Estimar o anticipar un evento futuro o resultado basándose en información, análisis y modelos existentes.

Para conocer el detalle de cada proceso de creación de bases de datos que comentaremos a continuación, se puede consultar el código empleado adjunto en el capítulo de los Anexos, a partir de la página 88.

3.1.1. Casas de apuestas y cuotas

Tanto para los festivales nacionales (español y sueco) como para Eurovisión, la idea inicial era hacer un *Web Scraping* diario de las cuotas de las casas de apuestas expuestas en la web de *Eurovision World*, pero, como hemos dicho al inicio del proyecto, uno de los principales objetivos de este era aprender a programar en Python des de cero.

Debido a la inexperiencia, finalmente se extrajeron los valores de los festivales nacionales de manera manual, dejándolos anotados diariamente en un Excel para su posterior tratamiento. Para Eurovisión, ya se pudo hacer uso de la técnica comentada mediante código para obtenerlos.

Primero de todo fijábamos la página web de estudio y, con ayuda del paquete *Beautiful Soap*[16] de Python, se hacía un mapeado [17] completo de toda la información que estuviera capturada en una tabla. A partir de esta información, se realizaron tratamientos para obtener una visualización clara de los datos.

La base de datos tendrá las siguientes variables:

1. Interprete (C): Nombre del artista.
2. Canción (C): Nombre de la canción.
3. Fecha (F): Día, mes y año cuando se extrajeron los datos.
4. Probabilidad_ganar (N): Probabilidad en porcentaje de ganar el evento.
5. Casa_apuesta (N): Nombre de la casa de apuesta (como BET_365 o BETWAY).
6. Valor_casa_apuesta (N): Odds ratio de cada concursante.
7. Votos_Encuesta (N): Número de votos del público por cada concursante.

Vista de esta base de datos:

Interprete	Canción	Fecha	Probabilidad_ganar	Casa_apuesta	Valor_casa_apuesta	Votos_Encuesta
Agoney	Quiero Arder	27-ene	11%	BET_365	-	2687
Agoney	Quiero Arder	27-ene	11%	BETWAY	6	2687
Agoney	Quiero Arder	27-ene	11%	SMARKETS	4.5	2687
Agoney	Quiero Arder	27-ene	11%	BETFAIR_ EXCHANGE	4.7	2687

Tabla 3.1. Vista simplificada de la base de datos de casas de apuestas y cuotas del Benidorm Fest 2023

Para la presente base de datos obtuvimos cierta cantidad significativa de valores faltantes en cada festival para cada año estudiado.

[16] Biblioteca de Python utilizada para extraer datos de documentos HTML y XML.
[17] Proceso de asignar o relacionar elementos de un conjunto de datos de origen a elementos correspondientes en un conjunto de datos de destino, generalmente utilizando una correspondencia o función definida.

Los motivos de la aparición de estos valores son debidos a dos razones distintas:

- La casa de apuesta [18] no ha registrado una cuota para el momento en el que se han extraído los datos.
- La eliminación de un concursante comporta que ya no aparezcan las cuotas que le representan en los posteriores días o eventos (por ejemplo, si en un concursante es eliminado en la semifinal, ya no saldrán sus cuotas en adelante).

También tenemos valores extremos ya que la mayoría de cuotas se suelen mover entre los valores [1,100] pero las casas de apuestas pueden otorgar hasta cuotas de 1000.

3.1.2. Parámetros Spotify

Como se ha comentado en el apartado de la metodología del proyecto, para obtener los parámetros que caracterizan las canciones de les distintos festivales, hicimos uso de la Interfaz de Programación de Aplicaciones (API) de Spotify.

Primero se hizo una extracción de la URI [19]de cada canción, una cadena de caracteres que identifica una canción específica dentro del catálogo de Spotify. Mediante esta y nuestras credenciales de la aplicación de desarrolladores, extrajimos los parámetros que se querían estudiar.

Además de definir e indicar el tipo de variables que tendremos en esta base de datos, indicaremos el rango de valores que toman.

Primeramente, tenemos las dos variables comunes:

1. Interprete (C): Nombre del artista.
2. Canción (C): Nombre de la canción.

A partir de la tercera columna tenemos estos parámetros de Spotify:

3. Popularity (N): Indica qué tan popular es una canción basándose en sus reproducciones en la plataforma. (va de 0 a 100)
4. Danceability (N): Indica qué tan bailable es una canción basándose en diferentes elementos musicales como tempo, ritmo, *beat*, entre otros. (va de 0 a 1)
5. Energy (N): Valores altos de energía representan canciones rápidas, activas y ruidosas, de lo contrario la energía es baja. Toma en cuenta aspectos como rango dinámico, timbre, volumen percibido, entre otros. (va de 0 a 1)
6. Key (N): Clasifica el pitch o las notas de la canción. Canciones con bajo *pitch* suenan más graves, mientras que con alto *pitch* suenan más agudos. Cuando no se conoce el valor se marca -1. (va de -1 a 11)
7. Loudness (N): La presencia de ruido en decibeles (dB), la amplitud de las ondas de la canción. Este valor es el promedio de la canción. (va de -60 a 0)

[18] Entidad que acepta y gestiona apuestas sobre eventos deportivos u otros acontecimientos, estableciendo cuotas y pagando ganancias en caso de acierto.
[19] Identificación única y global que se utiliza para acceder y referenciar de manera precisa a esa canción en particular en servicios de streaming de música o plataformas digitales.

8. Mode (N): Indica la modalidad (mayor o menor) de una canción, el tipo de escala del que se deriva su contenido melódico. (va de 0 a 1)
9. Speechiness (N): Detecta la presencia de palabras. Muy altos niveles indican audiolibros, podcasts, entre otros; altos niveles pueden ser canciones del género rap, valores bajos indican poco aporte de palabras en la canción. (va de 0 a 1)
10. Acousticness (N): Es el nivel de confianza de que la canción sea acústica, que resalta más la melodía y la voz del cantante. (va de 0 a 1)
11. Instrumentalness (N): Predice si la canción no contiene vocales. Entre más instrumental sea la canción, más alto el valor de este atributo. Canciones del género de rap tienen un bajo valor. (va de 0 a 1)
12. Liveness (N): Este valor indica la presencia de audiencia. Entre más audiencia detecta, más alto su valor y mayor probabilidad es que la canción haya sido en vivo. (va de 0 a 1)
13. Valence (N): Describe la positividad de la canción. Valores altos indican mayor positividad (alegría, euforia, ánimos) y valores bajos indican más negatividad (tristeza, depresión, enojo). (va de 0 a 1)
14. Tempo (N): Los *Beats Per Minute* (BPM), la velocidad o el paso de una canción.
15. Duration_ms (N): El tiempo de duración de la canción en milisegundos.
16. Markets (C): Especificación de un mercado o región específica de donde se han extraído los parámetros comentados. Son códigos de dos letras que representan a un país o región. Por ejemplo "US" para Estados Unidos o "ES" para España.

Vista de esta base de datos (para algunos parámetros[20]):

Interprete	Canción	Popularity	Danceability	Energy	Key	Loudness	...
Agoney	Quiero Arder	54	0.696	0.854	1	-3.507	...
Alfred García	Desde Que Tú Estás	47	0.611	0.637	9	-4.409	...
Alice Wonder	Yo Quisiera	49	0.382	0.366	6	-10.679	...
Aritz	Flamenco	49	0.766	0.589	9	-5.777	...

Tabla 3.2. Vista simplificada de la base de datos de los parámetros de Spotify del Benidorm Fest 2023

En esta base de datos no tenemos valores faltantes ya que los parámetros se extraen enteros de la aplicación de desarrolladores, por lo que tenemos toda la información de cada canción de cada interprete.

Tampoco tendremos valores extremos ya que estos parámetros están definidos en un rango de valores, por lo que nunca encontraremos un valor atípico en la base.

3.1.3. Topic Modelling

El *Topic Modelling* es una técnica de procesamiento de lenguaje natural que se utiliza para identificar los temas (tópicos) principales en un conjunto de documentos.

[20] Variable o valor que se utiliza en una función, método o programa para recibir y transmitir información, permitiendo la personalización y configuración de su comportamiento.

Permite extraer patrones de contenido a partir de una colección de textos, lo que es especialmente útil en el análisis de grandes conjuntos de datos textuales.

El objetivo de la técnica es agrupar las palabras en temas específicos y, de esta manera, resumir el contenido de los documentos de forma más concisa y significativa. Por ejemplo, en un conjunto de noticias, los temas podrían ser deportes, política, economía, entre otros.

La técnica de *Topic Modelling* es un método no supervisado de aprendizaje automático, lo que significa que no necesita una etiqueta o categoría predefinida para aprender de los datos. El algoritmo se basa en encontrar patrones de coocurrencia entre las palabras en los documentos, y a partir de estos patrones, agrupar las palabras en tópicos.

Uno de los algoritmos más populares de *Topic Modelling* es *Latent Dirichlet Allocation* (LDA). Este algoritmo se basa en la suposición de que los documentos están compuestos por una combinación de temas, y cada palabra en el documento está relacionada con uno o más de estos temas. A partir de esta suposición, LDA busca encontrar los temas y las palabras más relevantes para cada uno.

Esta técnica nos servirá para agrupar las canciones en tópicos y de ello obtendremos una nueva característica para la base de datos final. Por este motivo, y al ser una base totalmente de variable categóricas, no tendremos valores faltantes ni valores extremos.

3.1.3.1. Base de datos previa

Para ello, partiremos de una base de datos sencilla que constará de las siguientes columnas:

1. Title (C): Nombre de la canción
2. Artist (C): Nombre del artista
3. Lyrics (C): Letra de la canción

Para obtener esta base de datos, usaremos la API de Genius y las *playlists* (listas de canciones) de cada festival en Spotify, para extraer de forma completa las letras de las canciones y así, posteriormente, poder manipularlas y crear los tópicos.

La base de datos previa se verá así:

Title	**Artist**	**Lyrics**
EAEA	Blanca Paloma	ya ea ya ea ole ...
Nochentera	Vicco	sábado noche tengo bebida fría en la nevera luces neón por toda la escalera ...
Quiero Arder	Agoney	atado a tu juego yo disparo primero alimento mi ego ...

Tabla 3.3. Vista de la base de datos previa al Topic Modelling del Benidorm Fest 2023

3.1.3.2. Procedimiento

Una vez tenemos la base de datos previa, para realizar el *Topic* Modelling seguiremos estos pasos:

1. Ya que el módulo “es_core_web_md” todavía no está bien integrado en el paquete Spacy de Python, usaremos la versión para hacer el proceso de lematización en inglés. Por este motivo, primero de todo traduciremos la columna de los *lyrics* de las canciones al inglés mediante el paquete de Python Translator del módulo Googletrans (definido en el apartado de herramientas utilizadas).

2. A continuación, empezaremos por limpiar los *lyrics* eliminando *stopwords* (palabras vacías que carecen de sentido cuando se escriben solas o sin la palabra clave o *keyword*) y otras palabras o expresiones que se usan en canciones que también no nos servirán (fijadas en el código como “otras_palabras”).

3. Acto seguido realizaremos el proceso de lematización, que es el proceso de reducir una palabra a su forma base o "lemas", eliminando las inflexiones de la palabra como género, número, tiempo, modo, etc. Por ejemplo, "correr", "corre", "corremos" y "corriendo" se reducirían a "correr" mediante la lematización.

4. Para construir el modelo LDA, definiremos el diccionario, que es una lista de palabras únicas que se utiliza para indexar y representar cada documento en el corpus, y este que es el conjunto de documentos que se utilizan para entrenar el modelo de *Topic Modelling*. El modelo *Latent Dirichlet Allocation* (LDA) se utiliza para descubrir los temas o tópicos ocultos en un corpus de documentos. En el modelo LDA, cada documento será una mezcla de varios temas y cada tema será una mezcla de varias palabras.

5. Una vez tenemos el modelo LDA, asignaremos un tema a cada canción en función de la similitud entre su contenido y los diferentes tópicos y se calculan las distancias entre las canciones en función de su contenido temático. Se imprime el tema más significativo para cada canción, junto con las palabras clave del tema. El número de temas óptimo para hacer el análisis lo hemos intentado medir mediante la métrica de coherencia, pero como tenemos más de un proceso aleatorio, genera un gráfico distinto cada vez que se ejecuta el código:

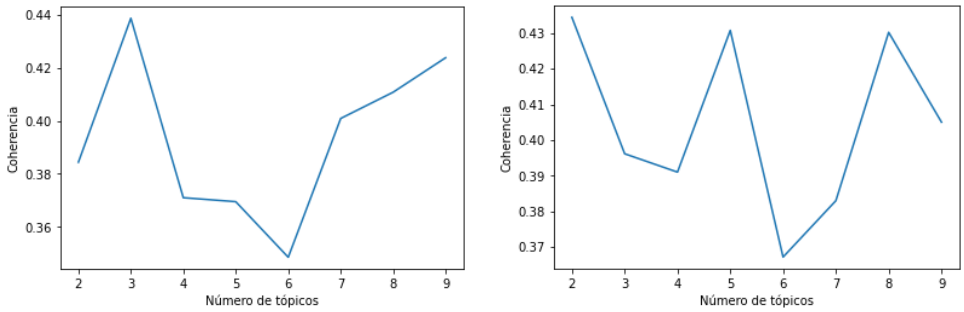


Figura 3.1. Numero de tópicos usando el nivel de coherencia

Por esta razón, fijaremos el número de temas en 7 ya que nos dará una distribución de tópicos más equitativa.

6. Finalmente, atribuiremos una palabra a cada tema que relacione a toda la nube de palabras, así sabremos qué quiere transmitirnos cada canción a partir de la técnica aplicada de *Topic Modelling*.

La base de datos final se verá así:

Title	*Artist*	*Lyrics*	*Lemas*	*Tópico*
Mi Familia	*Fusa Nocta*	*miri d your video lot hope you come one day or that we will see you little kiss family my ...*	*['video', 'lot', 'hope', 'come', 'one', 'day', 'see', 'little', 'kiss', 'family', 'family', 'family', ...*	*Nostalgia*
AIRE	*Sharonne*	*always look at excuses you don want to see it look look at your reflection who you want to see no one will stop you...*	*['always', 'look', 'excuse', 'want', 'see', 'look', 'look', 'reflection', 'want', 'see', 'one', 'stop', 'let', 'child', 'go', ...*	*Aventura*
Quiero Arder	*Agoney*	*tied to your game shoot first my ego food without error with success if you look at me and you don want to see ...*	*['tie', 'game', 'shoot', 'first', 'ego', 'food', 'without', 'error', 'success', 'look', 'want', 'see', 'force', , ...*	*Místico*

Tabla 3.4. Vista de la base de datos tras realizar ell Topic Modelling del Benidorm Fest 2023

3.1.3.3. Resultados

A partir del proceso de *Topic Modelling* hemos extraído las siguientes conclusiones de las letras de las canciones:

Para el Benidorm Fest

1- Las palabras que más se repiten son las siguientes:

2022	2023
Like (gustar): 35 veces	Look (mirar): 54 veces
Say (decir): 30 veces	Night (noche): 48 veces
Go (ir): 28 veces	Go (ir): 43 veces
Cry (llorar): 25 veces	Want (querer): 42 veces
Fly (volar): 24 veces	See (mirar): 35 veces

Tabla 3.5. Comparativa de las palabras más usadas entre las ediciones del Benidorm Fest 2022 y 2023

2- La distribución de las canciones entre los tópicos es la siguiente:

2022

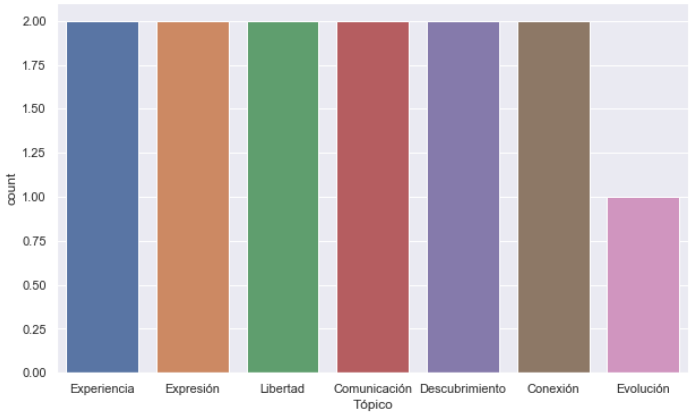


Figura 3.2. Distribución de los tópicos para el Benidorm Fest 2022

2023

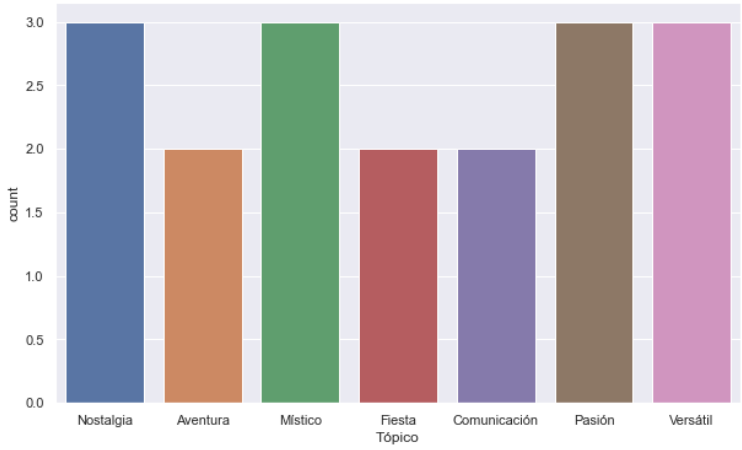


Figura 3.3. Distribución de los tópicos para el Benidorm Fest 2023

3- Agrupando las canciones en listas imaginarias a partir de sus tópicos respectivos, obtenemos la siguiente tabla:

2022

EXPERIENCIA	**EXPRESIÓN**	**LIBERTAD**	**COMUNICACIÓN**	**DESCUBRIMIENTO**	CONEXIÓN	EXPERIENCIA
SloMo - Chanel	Ay Mamá - Rigoberta Bandini	Terra - Tanxugueiras	Calle de la Llorería - Rayden	Raffaella – Varry Brava	ECO - XEINN	Mejores - Unique
Secreto De Agua - Blanca Paloma	Make You Say - Sara Deop	Culpa - Eurovision Edit - Javiera Mena	Sigues En Mi Mente - Marta Sango	Quién Lo Diría - Gonzalo Hermida	Postureo - Azucar Moreno	

Tabla 3.6. Listas de canciones según su tópico para el Benidorm Fest 2022

2023

COMUNICACIÓN	**FIESTA**	**VERSÁTIL**	**MÍSTICO**	**NOSTALGIA**	**PASIÓN**	**AVENTURA**
TRACCIÓN - Rakky Ripper	Nochentera -Vicco	UFF! - E'FEMME	Quiero Arder - Agoney	Mi Familia - Fusa Nocta	Flamenco -Aritz	AIRE - Sharonne
Arcadia - Megara	TUKI - Sofía Martín	No Nos Moverán - MELER	Que Esclati Tot - Siderland	Inviernos en Marte - José Otero	La Lola - Famous Oberogo	Yo Quisiera - Alice Wonder
		Desde Que Tú Estás - Alfred García	EAEA - Blanca Paloma	Quiero y duelo - Karmento	Sayonara -Twin Melody	

Tabla 3.7. Listas de canciones según su tópico para el Benidorm Fest 2023

Para el Melodifestivalen

1- Las palabras que más se repiten son las siguientes:

2022	2023
Like (gustar): 98 veces	Go (ir): 112 veces
Go (ir): 76 veces	Know (saber): 58 veces
Get (obtener): 63 veces	Want (desear): 51 veces
Love (querer): 58 veces	Get (obtener): 48 veces
Want (desear): 42 veces	Let (permitir): 43 veces

Tabla 3.8. Comparativa de las palabras más usadas entre las ediciones del Melodifestivalen 2022 y 2023

2- La distribución de las canciones entre los tópicos es la siguiente:

2022

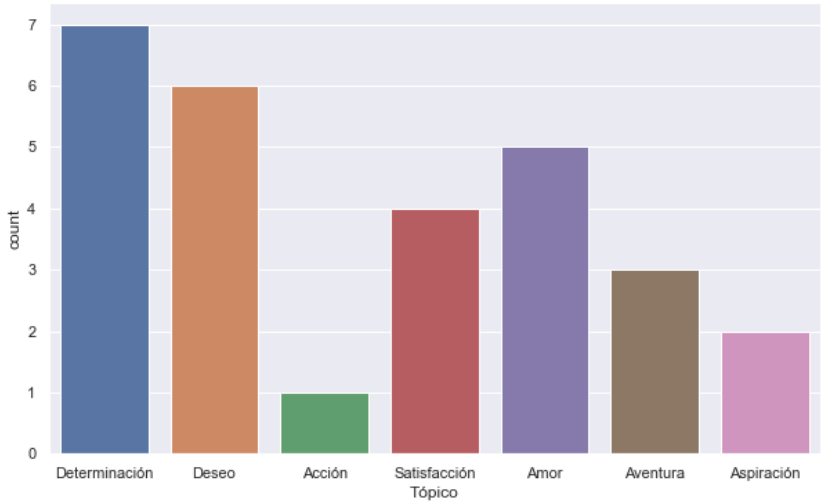


Figura 3.4. Distribución de los tópicos para el Melodifestivalen 2022

2023

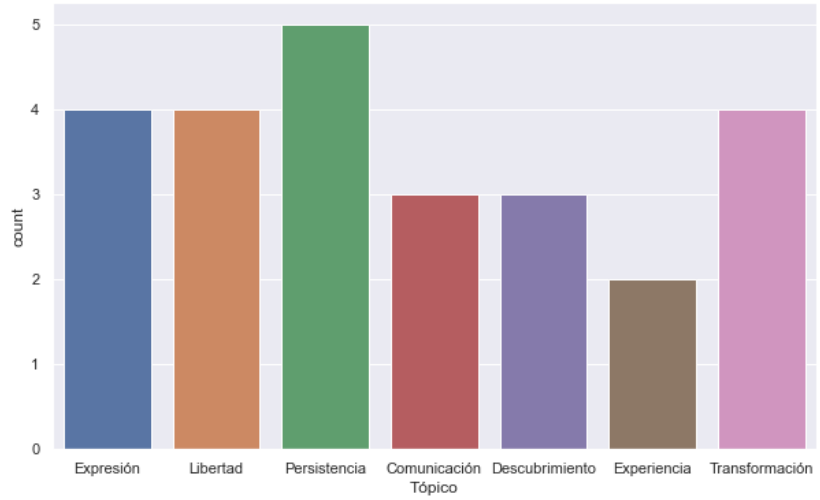


Figura 3.5. Distribución de los tópicos para el Melodifestivalen 2023

3- Agrupando las canciones en listas imaginarias a partir de sus tópicos respectivos, obtenemos la siguiente tabla:

2022

DETERMINACIÓN	DESEO	ACCIÓN	SATISFACCIÓN	AMOR	AVENTURA	ASPIRACIÓN
Bananas – Malou Prytz	Som du vill - Theoz	Moving Like That - Omar Rudberg	Innocent Love – Robin Bengtsson	Bluffin - LIAMOO	Tror du att jag bryr mig - Niello	Lyckligt slut - Lancelot
Let There Angels – Shirley Clamp	Hallabaloo - Danne Stråhed		I Can't Get Enough - Cazzi Opeia	I Want to Be Loved - Samira Manners	Face in the Crowd - Browsing Collection	La Stella – Tenori
Hold Me Closer – Cornelia Jakobs	Suave – Álvaro Estrella		Best to Come - Lisa Miskovsky	Fyrfaldigt hurra - Linda Bengtzing	Run To The Hills - Klara Hammarström	
My Way – Tone Sekelius	Änglavakt - John Lundvik			Higher Power - Anna Bergendahl		
Shut me up – Tribe Friday	Bigger Than The Universe - Anders Bagge			Till Our Days Are Over - Lillasyster		
Freedom – Faith Kakembo						
Synd om dig – Malin Christin						

Tabla 3.9. Listas de canciones según su tópico para el Melodifestivalen 2022

2023

EXPRESIÓN	LIBERTAD	PERSISTENCIA	COMUNICACIÓN	DESCUBRIMIENTO	EXPERIENCIA	TRANSFORMACIÓN
Tattoo - Loreen	Air - Marcus & Martinus	Six Feet Under - Smash Into Pieces	Mer av dig - Theoz	Where Did You Go - Kiana	Royals – Paul Rey	Never Give Up - Maria Sur
Gorgeous - Axel Schylström	Where You Are (Sávežan) - Arc North	Diamonds - Victor Crone	On My Way - Panetoz	One Day - Mariette	Så kommer känslorna tillbaka – Casanovas	Släpp alla sorger - Nordman
Låt hela stan se på - Ida-Lova	For The Show - Melanie Wehbe	RAGGEN GÅR - Elov & Beny	Rhythm Of My Show - Tone Sekelius	Now I Know - Tennessee Tears		All My Life (Where Have You Been) - Wiktoria
Comfortable - Eden	Sober - Laurell	Edelweiss - Signe & Hjördis				Mera mera mera - EMIL HENROHN
		Grytan - Uje Brandelius				

Tabla 3.10. Listas de canciones según su tópico para el Melodifestivalen 2023

Para Eurovisión

6- Las palabras que más se repiten son las siguientes:

2021	2022	2023
Go (ir): 133 veces	Go (ir): 112 veces	Go (ir): 131 veces
Get (obtener): 96 veces	Like (gustar): 84 veces	Like (gustar): 105 veces
Like (gustar): 89 veces	Take (coger): 78 veces	Get (obtener): 75 veces
Feel (sentir): 79 veces	Know (saber): 66 veces	Heart (corazón): 71 veces
Love (querer): 76 veces	Get (obtener): 64 veces	Love (querer): 68 veces

Tabla 3.11. Comparativa de las palabras más usadas entre las ediciones de Eurovisión 2021, 2022 y 2023

2. La distribución de las canciones entre los tópicos es la siguiente:

2021

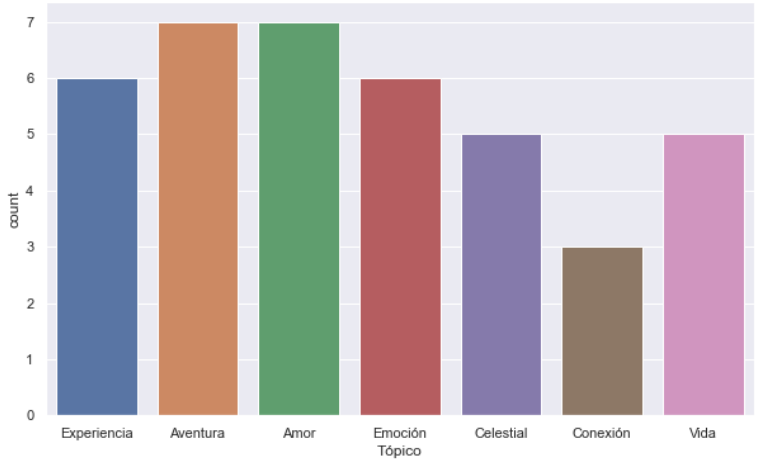


Figura 3.6. Distribución de los tópicos para Eurovisión 2021

2022

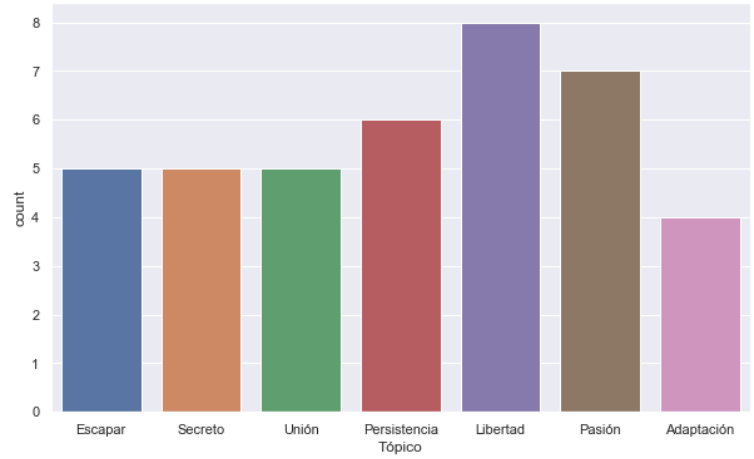


Figura 3.7. Distribución de los tópicos para Eurovisión 2022

2023

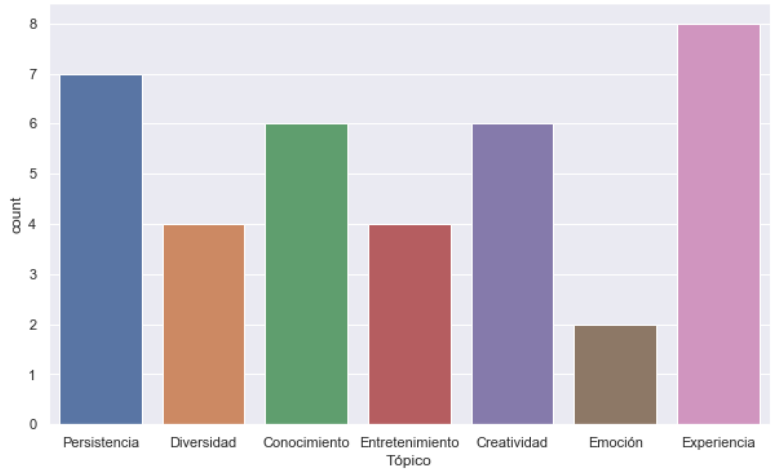


Figura 3.8. Distribución de los tópicos para Eurovisión 2023

3. Agrupando las canciones en listas imaginarias a partir de sus tópicos respectivos, obtenemos la siguiente tabla:

2021

EXPERIENCIA	**AVENTURA**	**AMOR**	**EMOCIÓN**	**CELESTIAL**	**CONEXIÓN**	**VIDA**
Zitti e buoni - Måneskin	Voilà - Barbara Pravi	Tout l'univers - Gjon's Tears	Last Dance - Stefania	Sugar - Natalia Gordienko	Voices - Tusse	Set Me Free - Eden Alene
Je me casse - Destiny	Dark Side - Blind Channel	10 Years - Daði Freyr	Birth Of A New Age - Jeangu Macrooy	Fallen Angel - TIX	Voy a quedarme - Blas Cantó	Mata Hari - Efendi
Discoteque - THE ROOP	Love Is On My Side - The Black Mamba	SHUM - Go_A	I Don't Feel Hate - Jendrik	Karma - Anxhela Peristeri	Amen - Ana Soklič	Øve Os På Hinanden - Fyr Og Flamme
Russian woman - manizha	Loco loco - Hurricane	El Diablo - Elena Tsagrinou	Tick-Tock - Albina	Embers - James Newman		Maps - Lesley Roy
Growing up is getting old - VICTORIA	The lucky one - Uku Suviste	Adrenalina - Senhit	Amen - Vincent Bueno	Omaga - Benny Cristo		The Moon Is Rising - Original - Samanta Tina
Amnesia - roxen	Technicolour - Montaigne	The Wrong Place - Hooverphonic	The Ride - Rafał Brzozowski			
	Here I Stand - Vasil Garvanliev	You - Tornike Kipiani				

Tabla 3.12. Listas de canciones según su tópico para Eurovisión 2021

2022

ESCAPAR	SECRETO	UNIÓN	PERSISTENCIA	LIBERTAD	PASIÓN	ADAPTACIÓN
Stefania (Kalush Orchestra) - KALUSH	SPACE MAN - Sam Ryder	SloMo - Chanel	Hold Me Closer - Cornelia Jakobs	River - Ochman	Sentimentai - Monika LIU	Lights Off - We Are Domi
Die Together - Amanda Tenfjord	Brividi - Mahmood	In corpore sano - Konstrakta	saudade, saudade - Various Artists	Llámame – WRS	SNAP - Rosa Linn	Fulenn (feat. Ahez) - Alvan
Boys Do Cry - Marius Bear	Hope - STEFAN	Trenulețul (cu Frații Advahov) - Zdob si Zdub	De Diepte - S10	Jezebel - The Rasmus	STRIPPER - Achille Lauro	I.M - Michael Ben David
Circles - Andrea	Not The Same - Sheldon Riley	Give That Wolf A Banana - Subwoolfer	Fade to black - Nadir Rustamli	Rockstars - Malik Harris	Lock Me In - Circus Mircus	Eat Your Salad - Citi Zēni
Guilty Pleasure - Mia Dimšić	That's Rich - Brooke	Með hækkandi sól - Various Artists	Miss You - Jérémie Makiese	Halo (feat. PIA MARIA) - LUM!X	SEKRET - Ronela Hajati	
			The Show - REDDI	I Am What I Am - Emma Muscat	Ela - Andromache	
				Breathe - Vladana	Disko - LPS	
				Intention - Intelligent Music Project		

Tabla 3.13. Listas de canciones según su tópico para Eurovisión 2022

2023

PERSISTENCIA	DIVERSIDAD	CONOCIMIENTO	ENTRETENIMIENTO	CREATIVIDAD	EMOCIÓN	EXPERIENCIA
Tattoo - Loreen	Cha Cha Cha - Käärijä	Due Vite - Marco Mengoni	Promise - Voyager	My Sister's Crown - Vesna	Stay - Monika Linkyte	Break a Broken Heart -Andrew
Unicorn - Noa Kirel	Queen of Kings - Alessandra	Bridges - ALIKA	Solo - BLANKA	EAEA - Blanca Paloma	Aijā - Sudden Lights	Future Lover - Brunette
Heart of Steel - TVORCHI	Because Of You - Gustaph	Duje - Albina Kelmendi	Watergun - Remo Forrer	Soarele si Luna - Pasha Parfeni		Évidemment - La Zarra
Mama ŠČ! - Let 3	Who the Hell Is Edgar? - TEYA	Samo mi se spava - Luke Black	Carpe Diem - Joker Out	I Wrote A Song - Mae Muller		Ai Coração - Mimicat
Echo - Iru		D.G.T (Off And On) - Theodor Andrei	Promise - Voyager	POWER - Diljá		Blood & Glitter - Lord Of The Lost
Breaking My Heart - Reiley		Tell Me More - TuralTuranX		Dance (Our Own Party) - The Busker		What They Say - Victor Vernicos
Like an Animal - Piqued Jacks						Burning Daylight - Mia Nicolai
						We Are One - Wild Youth

Tabla 3.14. Listas de canciones según su tópico para Eurovisión 2023

3.1.4. Features o características de las canciones.

Ya que los parámetros de Spotify y los tópicos son características fijas que definen cada canción, es razonable juntar ambos datos en una sola base de datos. Por lo tanto, la vista de esta será la siguiente:

Canción	*Interprete*	*Parámetros Spotify*	*Tópico*
Mi Familia	*Fusa Nocta*	*…*	*Nostalgia*
AIRE	*Sharonne*	*…*	*Aventura*
Quiero Arder	*Agoney*	*…*	*Místico*

Tabla 3.15. Vista de base de datos Features del Benidorm Fest 2023

Además, aprovecharemos esta gran base de datos para sacar una nueva variable que establezca el género de los artistas.

Utilizaremos nuevamente la técnica de *Web Scraping* para extraer y mapear la información almacenada en la Wikipedia en español, específicamente relacionada con los artistas. Para determinar el género de cada uno, nos enfocaremos en el recuadro ubicado en la parte superior derecha de la página, que contiene un breve resumen de su información personal y profesional. Nos centraremos en la frase de "Años activo o activa", donde encontraremos indicios del género basados en palabras clave específicas. A partir de estas palabras clave, asignaremos un género determinado a cada artista.

Para aquellos que no precises de una página propia en la Wikipedia en español o no tengan esa información en el recuadro, usaremos un paquete de Python llamado “Gender_guesser.detector” que, a partir del nombre del artista, intenta predecir su género. Al ser este un procedimiento aleatorio sin certeza de mostrar la realizad, haremos una comprobación final de la variable para corregir los posibles errores.

En este chequeo también introduciremos el valor “Mixto” para aquellos artistas que estén compuestos por integrantes de distinto género.

Dicho esto, la base de datos final tendrá la siguiente vista:

Canción	*Interprete*	*Parámetros Spotify*	*Tópico*	*Sexo*
Mi Familia	*Fusa Nocta*	*…*	*Nostalgia*	*Femenino*
AIRE	*Sharonne*	*…*	*Aventura*	*Femenino*
Quiero Arder	*Agoney*	*…*	*Místico*	*Masculino*

Tabla 3.16. Vista de la base de datos final de Features del Benidorm Fest 2023

Esta es el resultado de hacer la unión de dos bases de datos a partir de ciertas variables comunes. A continuación, se muestra la representación gráfica del diagrama de flujos para poder entenderlo:

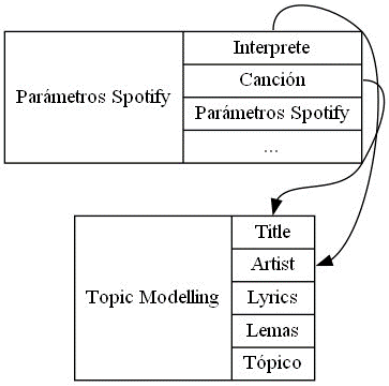


Figura 3.9. Diagrama de flujos de la base de datos final features

Las tres bases de datos expuestas serán las generales que usaremos para todos los festivales. Para Eurovisión tendremos una más que describiremos a continuación.

3.1.5. Indicador de amistad entre países

En este proyecto también hemos querido echar la vista atrás y hacer hincapié en los resultados históricos de Eurovisión durante todas las ediciones que se han disputado. Más específicamente, hemos recogido todas las puntuaciones que se han dado los países entre ellos en todas las finales de su historia (excepto la primera edición, en 1956, donde las votaciones se realizaron de forma secreta y no hay constancia de ellas), y hemos creado un indicador de "amistad" que define, como su nombre indica, los grupos de países que más afinidad tienen entre sí.

3.1.5.1. Base de datos inicial

Como hemos dicho, mediante un Excel con el desglose de las puntuaciones de cada edición del festival des del año 1957, partiremos de una base de datos que nos mostrará para cada año qué cantidad de puntos se donaron entre parejas de países.

Pais_Donante	Pais_Donado	Cantidad	Año
Russia	Armenia	24	2016
Austria	Albania	8	2012
Malta	Monaco	1	1975

Tabla 3.17. Vista de la base de datos inicial para sacar el indicador de amistad

3.1.5.2. Teoría de grafos

Para empezar, usaremos una estructura de datos para representar las relaciones entre entidades de los datos, los grafos. Estos consisten en un conjunto de nodos (también llamados vértices) y un conjunto de arista que conectan estos nodos. En este proyecto se utilizarán para representar relaciones no triviales y diversas entre los países a partir de las cantidades donadas.

En lugar de utilizar una estructura de tabla tradicional con filas y columnas, los grafos nos permitirán modelar las conexiones que hay entre el grupo de todos los países.

Cada nodo representará un país y cada arista que conecta dos nodos representará que hay amistad entre países.

Usaremos una técnica dentro de la teoría de grafos llamada asociación por comunidades. En el ámbito de la minería de datos, también es común hablar de clústeres cuando se agrupan valores según una característica. Las comunidades/clústeres se definen como un conjunto de nodos en el grafo que están más densamente conectados entre sí que con el resto de la red.

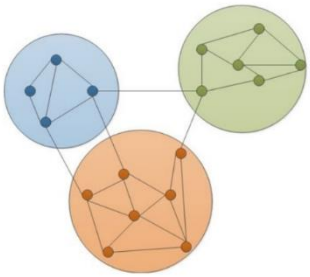


Figura 3.10. Vista general de un grafo por comunidades

3.1.5.3. Algoritmo de Louvain

Para lograr estas agrupaciones seguiremos el algoritmo de *Louvain*. Este sigue un enfoque heurístico y se basa en la maximización de la modularidad del grafo. La modularidad es una medida que evalúa la calidad de una partición de un grafo en comunidades. Mide la densidad de las aristas dentro de las comunidades en comparación con las aristas que están fuera. Un valor de modularidad cercano indica una partición de alta calidad, es decir, una división de comunidades muy densamente conectadas entre sí.

Este nos será de gran ayuda ya que es capaz de manejar grafos grandes debido a su eficiencia computacional y obtiene particiones de comunidades de alta calidad.

El algoritmo de *Louvain* consta de dos fases principales: la fase de optimización local y la fase de aglomeración.

1- Fase de optimización local:
   - En esta fase, inicialmente se asigna a cada nodo un identificador único de comunidad.
   - Luego, se itera sobre todos los nodos y se intenta mover cada uno a la comunidad vecina que maximice el incremento en la modularidad.
   - Este proceso se repite para cada nodo hasta que no se pueda encontrar una mejora significativa en la modularidad.
2- Fase de aglomeración:
   - En esta fase, se construye un nuevo grafo, donde los nodos representan las comunidades encontradas en la fase anterior.
   - Las aristas entre las comunidades se ponderan con la suma de los pesos de todas las aristas entre los nodos correspondientes en las comunidades originales.
   - Se repite el proceso de optimización local en este nuevo grafo, buscando la mejor partición de comunidades.

Este proceso de aglomeración y optimización se repite iterativamente hasta que no se pueda encontrar más mejoras significativas en la modularidad global.
Gracias a este algoritmo hemos obtenido una división por comunidades de todos los países que participaron por cada año junto al índice de modularidad de la partición. De esta manera, ya tenemos los grupos definidos por su afinidad a la hora de otorgar puntos hasta el año 2022 y, por ende, el indicador.

La vista de la nueva base de datos, para un año en concreto, sería la siguiente:

Pais	Indicador	Modularidad	Año
Croatia	2	0.0907055	1987
Cyprus	2	0.0907055	1987
Denmark	3	0.0907055	1987
Finland	0	0.0907055	1987

Tabla 3.18. Vista de la nueva base de datos tras aplicar Louvain

3.1.5.4. Predicciones de las comunidades de 2023

Desde un principio se quieren predecir los resultados de la edición de Eurovisión de 2023, entonces, ya que no tenemos las cantidades de puntos que se han otorgado este año, haremos una pequeña predicción.

Cogeremos la lista del índice de modularidad que se ha obtenido de aplicar el algoritmo de *Louvain* para cada año y predeciremos el siguiente valor que correspondería a 2023 mediante la regresión lineal del paquete *Sklearn.* Tras este proceso se ha conseguido un valor de:

Valor_siguiente_2023 = 0.143419

A partir de este valor haremos una búsqueda otra vez en la lista de índices para encontrar el valor que se asemeja más y, de esta manera, otorgar los mismos grupos de ese año. Si no aparece un país que participa en 2023, iteraremos otra vez para encontrar el índice más parecido, y así hasta tener todos los valores.

Una vez hecho esto, ya tenemos el indicador de amistad como nueva variable y podemos ver como se une al resto de variables mediante el siguiente diagrama:

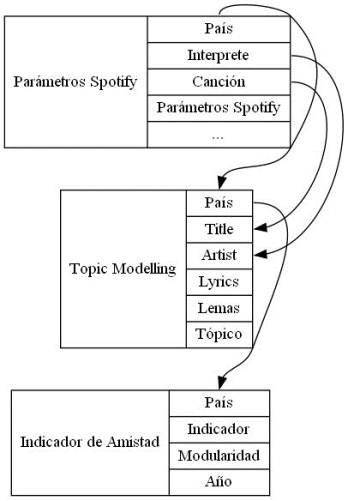


Figura 3.11. Diagrama de flujos de la base de datos final features

3.1.5.5. Animación de las comunidades por año

Finalmente, en este proyecto hemos querido implementar una vista animada de la base de datos que hemos obtenido tras obtener el indicador de amistad, que será representada en la exposición del proyecto.

Para ello seguiremos los siguientes pasos:

1- Leeremos el mapa del mundo gracias al paquete de Python *Geopandas* y obtendremos la información de las coordenadas de cada país.
2- Filtraremos por todos los países participantes en el festival de eurovisión.
3- Fijaremos la carpeta de nuestro directorio donde se crearán los *frames* (capturas).
4- Crearemos un mapa fijando parámetros de tamaño y los valores a estudiar.
5- Iteraremos por cada año y buscaremos de entre todos los países participantes, los que lo hicieron para aquella edición y los que no.
6- Uniremos ambos datos en el mapa europeo donde los que no participaron tendrán un color gris oscuro y los que sí lo hicieron se les asignará un color según su comunidad.
7- Configuramos parámetros de títulos y ejes, y guardamos en la carpeta cada mapa como un *frame* por año.
8- Una vez terminada la iteración, se creará una lista con todos los *frames* y se creará una animación con ayuda del paquete *Io*, que permite manipular datos en forma de objetos de flujo que es pueden leer o escribir y *Os* interactuar con el sistema operativo.
9- Guardaremos nuestra animación en el directorio y eliminaremos la carpeta con los *frames*.

Como no podemos representar el video en directo, primero de todo representaremos dos imágenes que nos muestran una pequeña parte de esta animación referentes a dos años aleatorios, y así se puede entender el resultado comentado.

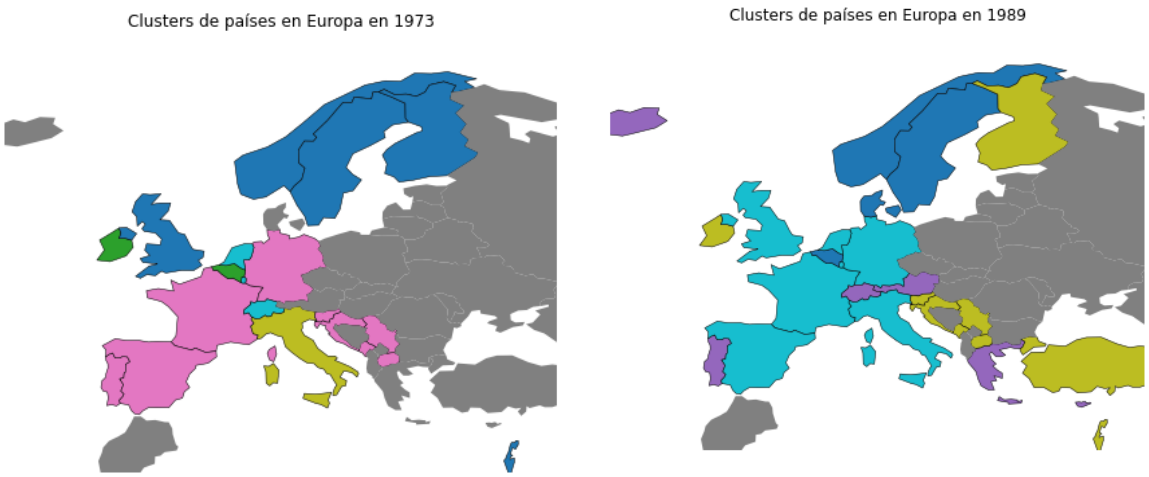


Figura 3.12. Muestra de la animación para los años 1973 y 1989

Como vemos, se pueden distinguir varios grupos de países para cada gráfico, pero nuestro objetivo es ver la tendencia total. Para ello formaremos un mapa general de las comunidades entre países. En vez de aplicar el algoritmo a todos los datos sin considerar el año, ya que este puede tener tendencia a generar grupos más grandes de lo esperado, usaremos la información de las dos bases de datos descritas.

Con la ayuda del módulo de Python *combinations* de *itertools*, sacaremos todas las combinaciones de parejas únicas de países que existen entre las variables del país donante y el país donado de la base de datos inicial. A partir de esta información, buscaremos el número de coincidencias donde cada pareja de países coincide en la misma comunidad a lo largo de los años.

La vista de esta información, ordenando la base por coincidencias será la siguiente:

Pais1	**Pais2**	**Coincidencias**
Croatia	Slovenia	44
Croatia	Serbia	42
Serbia	Slovenia	42
Montenegro	Serbia	40

Tabla 3.19. Vista de las coincidencias por parejas de países

Con tan solo una muestra de las cuatro primeras filas de esta base de datos, podemos ver que ya se pronuncia un grupo distintivo.

Usaremos el número total de coincidencias para ver patrones de afinidad entre países, y creamos un mapa donde todos los países que han participado, al menos una vez, en el festival de Eurovisión, se distribuyen por colores que representan cada comunidad.

El resultado es el siguiente:

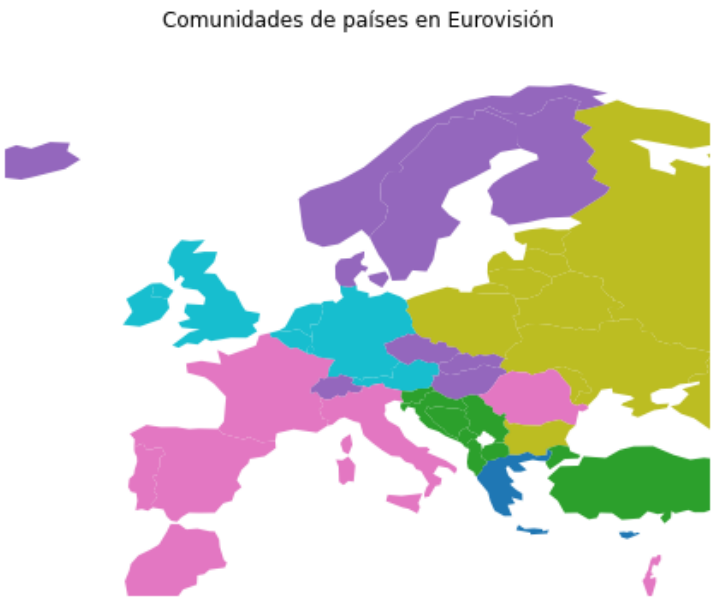


Figura 3.13. Separación por comunidades histórica de Eurovisión

Visto el mapa, podemos declarar con solvencia lo que hemos dicho en el capítulo de la introducción. Eurovisión es un concurso cuyo propósito es elegir la mejor canción de entre todos los participantes, pero anualmente es criticado por las puntuaciones que se otorgan entre países. Esto se confirma con la figura expuesta, donde la mayoría de países tienen más afinidad con los que les rodean, provocando que las puntuaciones pierdan valor.

3.1.6. Variable respuesta

La variable respuesta, también conocida como variable objetivo o variable dependiente, es un componente crucial en un proyecto de predicción basado en bases de datos. Se refiere a la variable que deseas predecir o estimar utilizando las demás variables o características disponibles en tus datos.

En nuestro caso, y como hemos comentado en el capítulo anterior donde exponíamos todos los festivales que estudiaremos, se busca predecir principalmente el ganador de cada uno de ellos. Para ello dispondremos de la información de los resultados de los años anteriores y el actual, teniendo como variables la posición y los puntos totales.

Aquí tendremos una pequeña distinción entre los festivales nacionales y Eurovisión, ya que este último cuenta, des del año 1996, con un pequeño grupo (Big-5) de países que pasan directamente a la final. Reino Unido, Alemania, Francia, Italia y España son los países que destinan más dinero a la UER (Unión Europea de Radiodifusión) y se les otorga el beneficio de no disputar las semifinales y tener su presencia anualmente en la gran final.

Debido a esto, la variable de los puntos pierde todo sentido ya que, si se quiere calcular los puntos de todos los países que participan en cada edición teniendo en cuenta las puntuaciones de las semifinales, habrá una descompensación con los integrantes del Big-5.

Ya que tenemos dos variables respuestas posibles, jugaremos con los puntos en los festivales nacionales y con la posición final en Eurovisión.

La vista de esta base de datos es la siguiente:

Canción	*Interprete*	*Puntos*	*Posición*
SloMo	*Chanel*	*189*	*7*
Postureo	*Azúcar Moreno*	*87*	*13*
Mejores	*Unique*	*306*	*2*
Make You Say	*Sara Deop*	*48*	*18*

Tabla 3.20. Vista de la variable respuesta para Benidorm Fest 2022

4. Preprocesamiento de los datos

Una vez que se ha completado el proceso de comprensión, es necesario preparar los datos para su posterior análisis. La preparación de los datos implica una serie de pasos, como la limpieza de datos, identificar los valores faltantes, los valores extremos y otra información relevante. También implica buscar posibles relaciones entre las variables, lo que puede ayudar a definir las hipótesis para el análisis posterior.

Finalmente, el último punto que veremos será la transformación de los datos, con la codificación de variables categóricas y la estandarización de los datos.

4.1. Preparación de los datos

La limpieza de datos es una etapa fundamental en el preprocesamiento de datos, que consiste en identificar y corregir o eliminar errores, inconsistencias y valores atípicos en los conjuntos de datos. La limpieza de datos es crucial para garantizar la calidad y confiabilidad de los datos antes de realizar análisis o aplicar algoritmos de aprendizaje automático.

Cada una de las bases de datos descritas han sido creadas manualmente sin que los datos hayan sido extraídos de una fuente de información como Kaggle. A priori, podemos decir que todos los datos serán importantes a la hora de decidir, pero hemos creído necesario eliminar las variables codificadas de los mercados de Spotify ya que para la mayoría de datos se repiten y esto provocarían datos duplicados que perjudicarían a la predicción.

4.1.1. Exploración de "missings" o valores faltantes

Los "*missing values*" son posiciones donde debería de haber valores, pero no aportan ninguna información. El tratamiento de los "*missings*" o valores faltantes ha sido un procedimiento necesario en nuestro proyecto ya que, reafirmando lo que hemos dicho en el apartado anterior, estos valores abundan tienen un peso importante en la base de datos de las casas de apuestas.

A continuación, vemos los porcentajes de valores faltantes en cada una de las apuestas de cada festival para cada año de estudio:

Festival	Porcentaje de *missings*
Benidorm Fest 2022	9,13%
Benidorm Fest 2023	2,53%
Melodifestivalen 2022	20,14%
Melodifestivalen 2023	8,94%
Eurovisión 2021	0,66%
Eurovisión 2022	2,21%
Eurovisión 2023	5,31%
Media	7%

Tabla 4.1. Porcentaje de missings por festival

Es importante identificar los *missing values* ya que el paquete de Python Sklearn no los acepta como datos cuando se crean los modelos de predicción, en el caso que se utilice un *missing value* aparece un error y no se crea ningún modelo.

Además, en los procesos de transformación de datos que veremos en el siguiente apartado, si las columnas de las variables contienen valores no numéricos (en este caso, un valor faltante se leería como “Nan”), no se podrán ejecutar y también saltarán errores.

Como hemos dicho en el apartado anterior, cuando realizamos el mapeo para extraer las cuotas de la página web de *Eurovisión World*, obtuvimos dos casuísticas distintas que provocaban la aparición de valores faltantes.

4.1.1.1. Concursante sin cuota sin motivo

El motivo por el cual hay valores faltantes aleatorios en nuestra base de datos tiene fácil explicación. Las cuotas en las casas de apuestas funcionan según las apuestas que hagan los jugadores; estas varían más o menos a partir del dinero que se haya apostado a cada concursante. Las fechas muy anteriores a cada evento tienen muy poca relevancia, ya que los jugadores disponen de muy poca información sobre los artistas que actuará, qué canciones interpretaran y cómo será la actuación sobre el escenario.

El conocimiento de estos factores es muy importante a la hora de decidir sobre quién apostar ya que, de lo contrario, solo se podría especular y se iría a ciegas.

Por este motivo, muchos de los jugadores de apuestas deciden esperar a las semanas claves para depositar su dinero, hecho que provoca que la variabilidad de las cuotas no se vea muy afectada y que las casas de apuestas decidan eliminar algunas de ellas.

Cabe destacar, que la gran mayoría de estos valores se encuentran en los concursantes que, a priori, tienen una probabilidad muy baja de ganar cada evento y, por lo tanto, las casas de apuestas quitan sus cuotas ya que no se usan.

Dicho esto, el tratamiento que usaremos para rellenar esta falta de información es el siguiente:

1- Nos aseguramos que la variable está en formato numérico. Si no es así, no se podría realizar el proceso.
2- Agrupamos nuestra base de datos según la fecha y después el artista. De esta manera dispondremos de un grupo de valores de forma seguida de un artista para cada fecha.
3- Aplicamos una técnica de rellenado de Python que saca la media de valores de cada artista para una fecha y lo asigna a los valores faltantes de ese grupo.

“*transform(lambda x: x.fillna(x.mean())*“

Con menos frecuencia, pero también significativos, encontramos valores faltantes para días completos o para concursantes en más de una fecha.
Para este tipo de casos usamos la interpolación, que es semejante a la media, pero calcula en base al número de valores faltantes en un espacio y los rellena proporcionalmente según al último valor y al próximo más cercano con información.

“*interpolate(method='linear', inplace=True)*”

4.1.1.2. Concursante sin cuota con motivo

Como su nombre indica, para estos valores faltantes sí tenemos la certeza de los motivos por el cual no obtenemos cuotas. Como pasa en la mayoría de concursos con fases previas eliminatorias, aquellos participantes que no logran la cantidad de puntos suficientes para clasificarse a la final, dejan de luchar por el título. Por consiguiente, las cuotas a partir de esa fecha dejan de registrarse ya que carecería de sentido alguno.

A causa de esto, a partir de la fecha en que un concursante queda eliminado en cada festival, obtenemos todo valores faltantes hasta el día de la final. Estos son más delicados que los primeramente descritos ya que no hemos encontrado un método fijo establecido que nos asigne valores con sentido. Es imposible imaginar la curva de valores real que podría tomar si no hubiesen sido eliminados.

Por este motivo, hemos usado otra técnica de rellenado en Python que se llama *Forward Fill* (Rellenado hacia delante), que simplemente replica el último valor con información de cada participante para cada casa de apuesta. "*ffill()*"

4.1.2. Análisis descriptivo de los datos previo

El análisis descriptivo de datos es una etapa fundamental en la exploración y comprensión de conjuntos de datos. Consiste en utilizar técnicas y métodos estadísticos para resumir, organizar y visualizar los datos con el objetivo de obtener una descripción clara y concisa de sus características principales.

Tras haber tratado los valores faltantes, empezaremos con el análisis univariante para comprender la distribución y las propiedades de las variables, y después aplicaremos el bivariante que nos ayudará a explorar las relaciones entre variables, detectar patrones y tendencias, y tomar decisiones informadas basadas en datos sólidos.

Ya que tenemos bases de datos con la misma información para cada festival a estudiar, expondremos el análisis solo para uno de ellos, ya que proporcionará el camino a seguir para los tres. El estudiado en nuestro caso será la edición de Eurovisión de 2023, empezaremos con un análisis de las variables individualmente y luego observaremos las posibles relaciones que hay entre ellas.

4.1.2.1. Análisis univariante

El objetivo principal del análisis univariante es comprender y describir las propiedades y patrones de una variable específica. Esto implica examinar su distribución, medidas de tendencia central (como la media, mediana y moda) y medidas de dispersión (como la desviación estándar y el rango).

Durante el análisis univariante, se aplicarán diversas técnicas y herramientas estadísticas para resumir y visualizar la información contenida en la variable estudiada. Estas técnicas incluirán histogramas, gráficos de barras, diagramas de caja y bigotes, y análisis de frecuencia.

Variables categóricas

Hay tres variables categóricas comunes entre las bases de datos de apuestas y *features*, que son las que nos describen el nombre del interprete, su canción y su país. Para estas no aplicaremos el análisis, ya que son fijas y sus valores tienen las mismas frecuencias en cada caso. Para las apuestas también tenemos otra variable de este tipo que es la que marca la casa de apuesta.

Aunque la fecha también se repite para todos los valores que se recogieron en el momento de la extracción, es importante recalcar que el rango de estas varía en cada festival y en cada año, ya que los eventos tuvieron lugar en momentos distintos. Para Eurovisión 2023, el primer día fue el 17 de marzo de 2023 y el último fue el día de la final del concurso, el 13 mayo del mismo año.

En este apartado las únicas variables que podemos describir son el Tópico y el Sexo de la base de datos *features*. La distribución de los tópicos ya ha sido mostrada en los resultados del Topic Modelling, por lo que expondremos solamente el género de los artistas:

- Sexo:

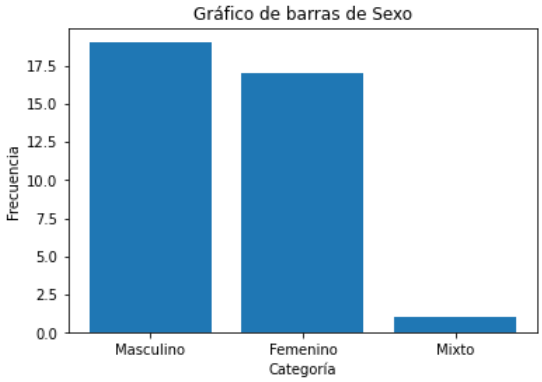


Figura 4.1. Gráfico de barras de la variable Sexo para Eurovisión 2023

Variables numéricas

Para la base de datos de features, los parámetros de Spotify son los únicos que contienen valores numéricos. No es razonable describir estas variables ya que, en la descripción de las mismas en el apartado de los datos iniciales, ya hemos definido los límites bien marcados que pueden tomar sus valores, por lo que consideramos que siempre obtendremos valores razonables sean altos o bajos.

El contrario pasa en las apuestas, como hemos dicho en el apartado anterior, estas variables no tienen un rango marcado donde se encuentran todos los valores, por lo que tenemos que estudiarlas más a fondo para tener una vista clara de los procesos que deberemos seguir.

- Valor_casa_apuesta (cuota):

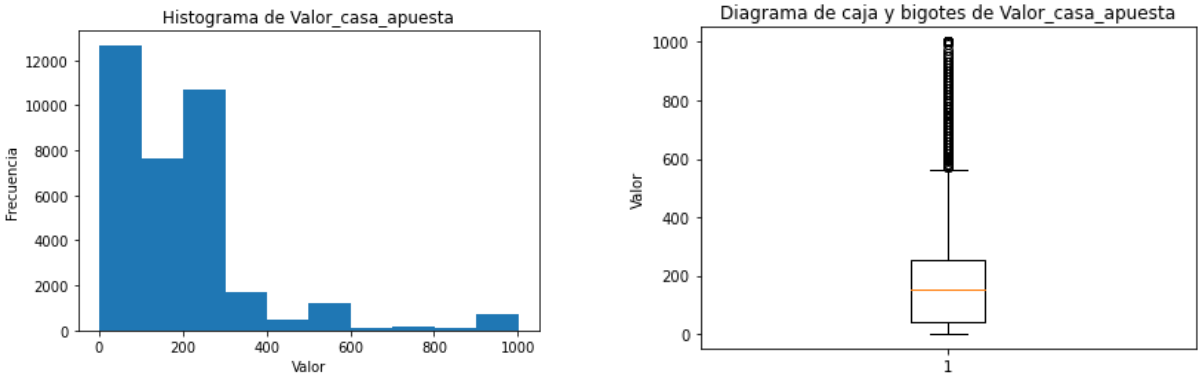


Figura 4.2. Histograma y diagrama de caja y bigotes de la variable cuota para Eurovisión 2023

Resumen	
cuenta	*32190.000000*
media	*154.569921*
std	*120.869602*
min	*1.000000*
25%	*42.000000*
50%	*151.000000*
75%	*250.000000*
max	*550.000000*

Tabla 4.2. Resumen de la variable cuota para Eurovisión 2023

Se puede observar claramente que tenemos una gran cantidad de valores extremos a partir del valor de cuota de 500.

- Probabilidad_ganar (en tanto por uno):

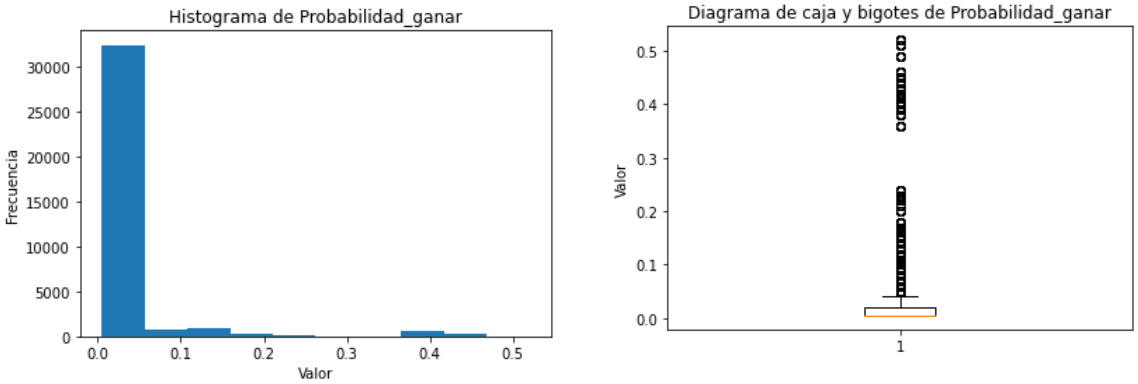


Figura 4.3. Histograma y diagrama de caja y bigotes de la variable probabilidad para Eurovisión 2023

Resumen	
cuenta	*32190.000000*
media	*0.028539*
std	*0.071231*
min	*0.005000*
25%	*0.005000*
50%	*0.005000*
75%	*0.020000*
max	*0.520000*

Tabla 4.3. Resumen de la variable probabilidad para Eurovisión 2023

Para este caso vemos que la gran mayoría de datos están comprendidos entre 0 y 0.1, por lo que los valores extremos provocan que el diagrama de cajas se vea muy afectado.

- Votos_Encuesta

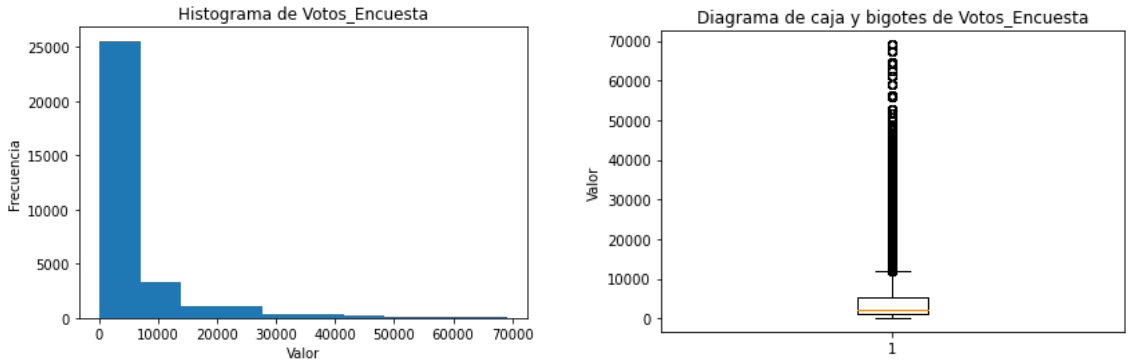


Figura 4.4. Histograma y diagrama de caja y bigotes de la variable votos para Eurovisión 2023

Resumen	
cuenta	*32190.000000*
media	*5706.152377*
std	*9028.505234*
min	*141.000000*
25%	*1151.000000*
50%	*2125.250000*
75%	*5491.000000*
max	*69095.000000*

Tabla 4.4. Resumen de la variable votos para Eurovisión 2023

Observamos el mismo patrón que en la variable anterior, donde la media está muy alejada de algunos valores muy altos.

Antes de tomar decisiones finales para las variables tenemos que observar el análisis bivariante, ya que nos proporcionará información que validará o no las acciones a tomar.

4.1.2.2. Análisis bivariante

Como hemos dicho anteriormente, este análisis examina las relaciones entre dos variables en un conjunto de datos. Para las variables categóricas de las apuestas no lo aplicaremos ya que existen grupos de país, interprete, canción, casa de apuesta y fecha, que se repiten a lo largo de la base de datos. Por esta razón solo lo aplicaremos para las numéricas:

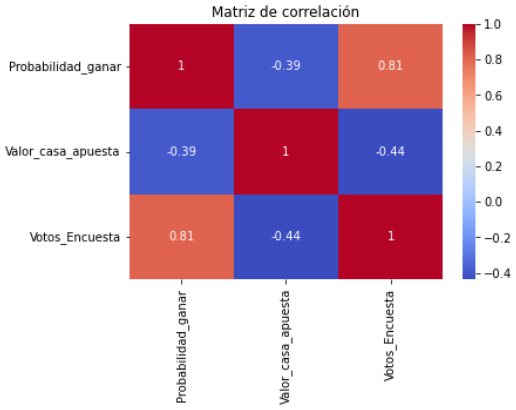


Figura 4.5. Matriz de correlación entre las variables de la base de datos de apuestas de Eurovisión 2023

En el gráfico de correlaciones vemos claramente que las variables Votos_Encuesta y Probabilidad_ganar están altamente correlacionadas. Esto es debido a que la encuesta estaba dispuesta en la misma página web de donde extrajimos la información, por lo que es fácil declarar que la audiencia se veía fuertemente sugestionada a votar esos artistas con mayor probabilidad a ganar. Por lo que podríamos considerarlo como un grupo, cuando uno sube el otro también, y viceversa.

Si ahora comparamos este grupo con el Valor_casa_apuesta, observamos que tienen una alta correlación negativa. El motivo es el mismo que antes, pero con una especificación. Cuantos más votos o probabilidad haya para un artista, la cuota será menor, debido a su funcionamiento que hemos explicado en el capítulo anterior.

Ambas situaciones las podemos ver mejor reflejadas en el siguiente gráfico:

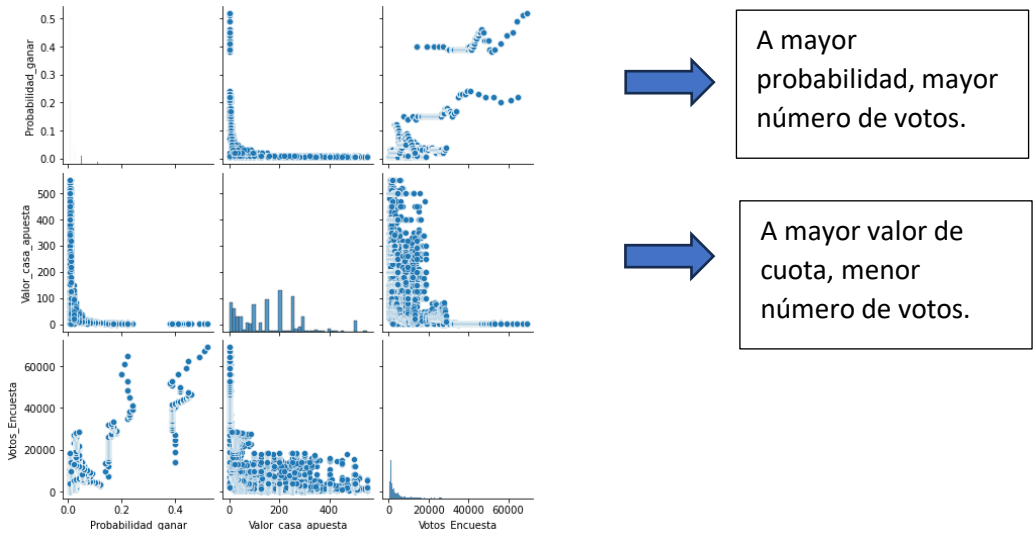


Figura 4.6. Gráfico de emparejamiento de las variables de la base de datos de apuestas de Eurovisión 2023

Dicho esto, podríamos considerar las tres variables como un mismo grupo que varía de la misma manera. Por lo tanto, solo nos quedaremos con las cuotas en esta base de datos, ya que nos proporcionan una información más detallada.

Para la base de datos *features* haremos también el mismo proceso de antes. Seleccionamos las variables numéricas de la base de datos, que corresponden a todos los parámetros de Spotify para cada canción, y estudiamos la matriz de correlación para hallar patrones entre ellos.

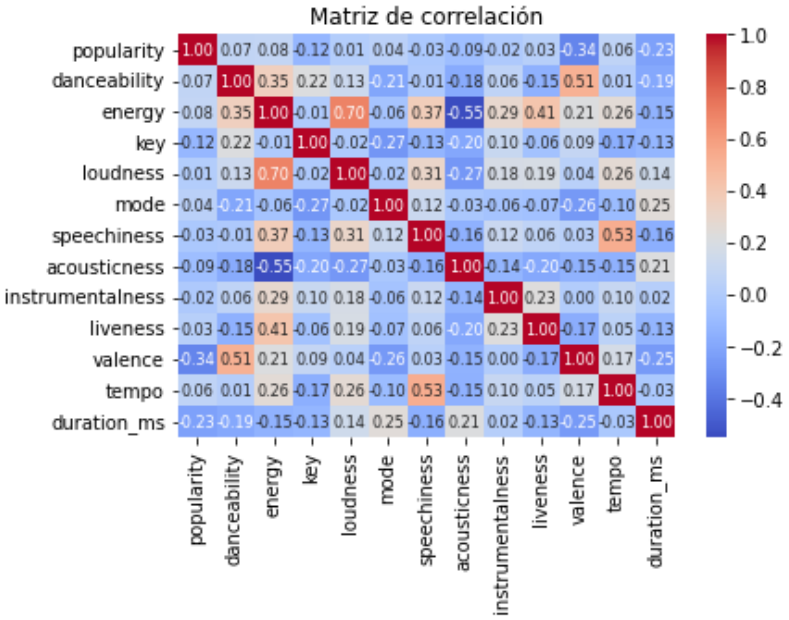


Figura 4.7. Matriz de correlación entre las variables de la base de datos de features de Eurovisión 2023

Observando la matriz, vemos dos relaciones de variables que destacan por la correlación entre ellas, tanto positiva como negativamente. Estudiaremos cada una de ellas:

- Para las variables "*Loudness*" y "*Energy*" vemos un valor positivo de 0,7. Esta situación es comprensible ya que cuantos más decibelios tenga una canción, más volumen tiene, por lo que, en general, son corresponde a una canción rápida y ruidosa, por ende, con más energía.
- Para las variables "*Acousticness*" y "*Energy*" vemos un valor negativo de -0,55. Esta situación también es comprensible ya que cuanto más acústica sea una canción, menos factores externos ensucian la voz del cantante, por lo que menos ruido provoca.

Ya que ambas relaciones contienen una variable en común, la energía, eliminaremos esta para evitar que las correlaciones afecten a nuestro estudio.

4.1.3. Exploración de “outliers” o valores extremos

Una vez hemos detectado las variables que contienen valores atípicos o muy alejados de la media, procederemos a hacer su tratamiento.

Los valores *outliers* o extremos son valores que se encuentran significativamente alejados de la mayoría de los otros valores en un conjunto de datos. Estos valores se consideran atípicos y pueden ser el resultado de errores de medición, variabilidad natural o eventos inusuales. La presencia de *outliers* puede afectar el análisis estadístico y es importante identificarlos para comprender mejor la distribución de los datos.

También reafirmando lo comentado en la descripción de las bases de datos, estos valores solamente los encontramos otra vez en la base de datos de las apuestas y, más específicamente, en la columna de la cuota (Valor_casa_apuesta).

Las casas de apuestas establecen las cuotas utilizando diversos factores, como estadísticas históricas, rendimiento de los participantes, opiniones de expertos y la demanda del mercado. Cuanto más baja sea una cuota, más alta la probabilidad de que ocurra un evento, y viceversa. Los apostantes pueden guiarse por estos valores, pero no siempre reflejan la realidad, ya que las casas de apuestas también incluyen un margen de beneficio para garantizar su rentabilidad. Dicho esto, para los concursantes con bajas posibilidades de ganar, las cuotas suelen ir incrementando gradualmente durante el paso del tiempo, provocando que tengamos valores extremos. Estos pueden llegar a tomar valores de hasta 1000, siendo el rango a tener en cuenta de 1 a 100.

Para tratarlos fijaremos una función en Python llamada “tratar_outliers” que usará el método de la media *trimming* (o recortada). En lugar de eliminar o reemplazar directamente los valores extremos, la media *trimming* busca una medida de tendencia central más robusta al recortar un porcentaje de los valores extremos en ambos extremos de la distribución. En nuestro caso solamente reemplazaremos aquellos valores que estén por encima del límite superior, ya que las cuotas bajas son de mucha importancia.

4.1.4. Análisis descriptivo de los datos final

Finalmente, habiendo tratado los valores extremos de la variable de la cuota de la base de datos de las apuestas, obtenemos el siguiente descriptivo individual:

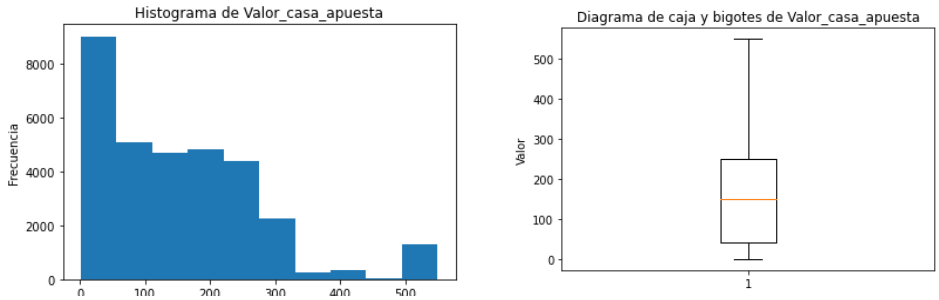


Figura 4.8. Nuevo histograma y diagrama de caja y bigotes de la variable cuotas para Eurovisión 2023

Resumen	
cuenta	*32190.000000*
media	*5706.152377*
std	*9028.505234*
min	*141.000000*
25%	*1151.000000*
50%	*2125.250000*
75%	*5491.000000*
max	*69095.000000*

Tabla 4.5. Nuevo resumen de la variable cuotas para Eurovisión 2023

Podemos ver como los valores ya se han regularizado y no se muestran extremos.

4.2. Transformación de datos

La transformación de datos es un proceso fundamental en el análisis de datos y se refiere a aplicar cambios o manipulaciones a los datos originales con el objetivo de mejorar su calidad, prepararlos para un análisis específico o ajustar su distribución. La transformación de datos puede ser necesaria debido a diversas razones, como corregir asimetrías, normalizar variables, reducir la escala de los datos o hacer que los datos cumplan con ciertas suposiciones estadísticas.

Ya que nuestro propósito será predecir mediante las bases de datos, seguiremos los siguientes pasos en este orden:

1- Unión de la información: concatenaremos tanto apuestas como *features* de todos los años en una sola base de datos para así poder hacer los tratamientos una sola vez.

2- Creación de *delays*: Para las apuestas reharemos la variable fechas de manera que se calculará la diferencia de días entre cada valor de la variable y la última de ellas. Con esto tendremos nuevas columnas que nos indicarán cada *delay (retardo)* y tendremos una vista más clara de la evolución de los datos.

Canción	*Interprete*	*Casa_apuesta*	*Año*	*0 delays*	*8 delays*	*…*
Desde que tú estás	*Alfred García*	*BETFAIR_EXCHANGE*	*2023*	*7*	*15*	*…*
Desde que tú estás	*Alfred García*	*BETWAY*	*2023*	*41*	*15*	*…*
Desde que tú estás	*Alfred García*	*BET_365*	*2023*	*101*	*15*	*…*
Desde que tú estás	*Alfred García*	*SMARKETS*	*2023*	*5*	*15*	*…*

Tabla 4.6. Vista de la base de datos tras crear los delays

3- Codificación*:* Para las variables categóricas de las *features*, usaremos la técnica *One-hot Encoding* para convertirlas a numéricas. Crearemos columnas adicionales en el conjunto, donde cada columna representará una categoría distinta de la variable categórica original. En nuestro caso lo aplicaremos al sexo de los artistas, donde la columna “Masculino” tendrá el valor de 1 si a ese artista le corresponde ese género y 0 si no es así.

4- El mismo funcionamiento para el resto de categorías y para las de la variable Tópico que, dependiendo del festival y el año de la base de datos, tendremos unas equivalencias distintas.
5- Estandarización: Para evitar que las variables con valores muy alejados respecto al resto se sobrepongan, aplicaremos estandarización a las variables input que usaremos para crear los modelos. Se ajustarán los datos para que tengan una media de 0 y una desviación estándar de uno.
6- Unión final: Después de estos dos tratamientos, recogeremos toda la información disponible y la agruparemos en una sola base de datos para el festival en estudio.

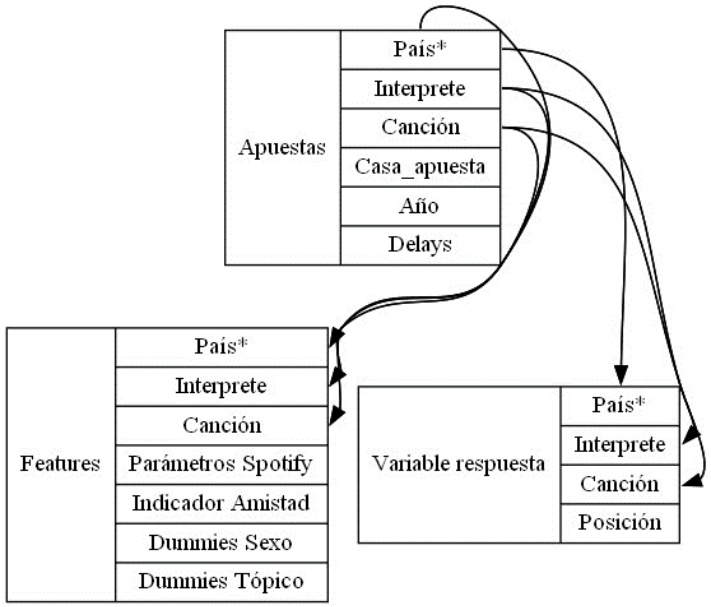


Figura 4.9. Diagrama de flujos de la base de datos final para cada festival

** Solo para el caso de Eurovisión*

Estos pasos de preprocesamiento son importantes para preparar los datos antes de aplicar algoritmos de aprendizaje automático. La codificación en *dummies* nos permite trabajar con variables categóricas de manera adecuada, la estandarización nos ayudará a crear modelos con datos mejor balanceados y la implementación de *delays* nos ayudará a comprender mejor la evolución de los datos. Ahora estamos listos para continuar con el análisis y construcción de modelos utilizando los datos preprocesados de manera correcta.

5. Modelaje y evaluación

Una vez realizado el preprocesamiento de los datos, podemos dar paso al modelaje y la evaluación. En un proyecto de análisis de datos o aprendizaje automático, son dos etapas esenciales que ayudan a construir y validar modelos predictivos.

El modelaje implica la construcción de un modelo predictivo utilizando algoritmos de aprendizaje automático o técnicas estadísticas. Esto implica seleccionar un algoritmo apropiado que se ajuste a los datos y al problema que se está abordando. Para nuestro proyecto usaremos tres tipos de modelos distintos para evaluar cuál es el que proporciona mejor predicción de los resultados de cada festival.

Implementaremos cada vez un modelo más complejo para ver cuál se adecua mejor a los datos, empezando por un árbol de decisión que se usa para tomar decisiones basadas en reglas lógicas; a continuación, utilizaremos el *XGBoost*, que combina múltiples árboles para construir un modelo más robusto; y finalizaremos con una red neuronal, que permitirá capturar patrones y dependencias temporales en los datos para hacer predicciones futuras. Durante esta etapa, se ajustan los parámetros del modelo y se entrena utilizando los datos de entrenamiento.

Una vez que se ha construido el modelo, es necesario evaluar su rendimiento. La evaluación del modelo implica medir su capacidad para hacer predicciones precisas sobre datos no vistos previamente. Esto se hace utilizando métricas de evaluación apropiadas según el tipo de problema. En este proyecto usaremos las siguientes de manera general:

- MAE (Error Absoluto Medio): Es el promedio de las diferencias absolutas entre los valores reales y los valores predichos. Cuanto menor sea este valor, mejor será el rendimiento del modelo en términos de precisión.
- MSE (Error Cuadrático Medio): Es el promedio de las diferencias al cuadrado entre los valores reales y los valores predichos. El MSE tiende a penalizar más los errores grandes que el MAE. Cuanto menor sea el valor, mejor será el rendimiento.
- RMSE (Raíz del error cuadrático medio): Es la raíz cuadrada del MSE y proporciona una medida del error promedio en la misma unidad que los valores de salida. Es una métrica ampliamente utilizada para evaluar el rendimiento de los modelos de regresión. Al igual que todos, cuanto menor sea el valor de RMSE, mejor.
- Valor R^2: También conocido como coeficiente de determinación, evalúa qué tan bien se ajusta el modelo a los datos observados. Su rango va de 0 a 1 y cuanto más cerca esté el valor a 1, mejor se explica la variabilidad en los datos y los valores predichos coinciden más con los observados.
- Promedio MAE: Medida general del error promedio del modelo en el conjunto de datos evaluado que se utiliza para comparar diferentes modelos o enfoques.

Por falta de tiempo a la hora de realizar las predicciones, usaremos solamente una casa de apuestas, diferente en cada festival, para formar los modelos. Para cada uno seleccionaremos la que creemos que nos dará unos datos con información más verídica sobre la realidad y no contenga muchos datos tratados. Tendremos "SMARKETS" para los festivales nacionales y "BET365" para Eurovisión.

Cabe remarcar de nuevo que la variable respuesta de los festivales nacionales y de Eurovisión serán distintas por razones ya comentadas. Para los primeros serán los puntos totales conseguidos y para el segundo será la posición final.

El primer paso antes de realizar cualquiera de los modelos descritos será separar nuestra base de datos completa entre las variables independientes (las numéricas y las *dummies*), cuyas estudiaremos para conocer su peso para predecir, y la variable respuesta. Además, dentro de cada separación haremos otra partición para obtener el conjunto de entrenamiento y el conjunto de prueba. El primero se utilizará para entrenar cada modelo y el segundo para evaluar el rendimiento final de este. En nuestro caso tendrán unos porcentajes de 67% y 33%, respectivamente. Una vez hecho esto, ya podemos implementar los diferentes modelos a nuestros festivales.

Como apunte fuera de esta explicación, se realizaron todos los primeros dos modelos con los datos estandarizados, pero, al obtener unos coeficientes de determinación de los ajustes muy bajos y unos errores de predicción muy grandes, se decidió proceder sin ese paso.

5.1. Árbol de decisión

Los árboles de decisión son modelos de aprendizaje automático que toman decisiones en base a condiciones y reglas. Se representan en forma de estructuras en forma de árbol, donde cada nodo representa una característica y cada rama representa una posible decisión o resultado. Los árboles de decisión son fáciles de entender e interpretar, pueden manejar datos numéricos y categóricos, y pueden utilizarse tanto para problemas de clasificación como de regresión, que sería nuestro caso.

Para cada caso empezaremos obteniendo la matriz de importancia relativa de diferentes características o variables en el modelo. Usaremos la técnica basada en los árboles de decisión. Calcularemos la importancia utilizando métricas como la ganancia de información, la reducción de impureza o la ganancia de Gini. Estas medirán como cada característica contribuye a la capacidad de dividir los datos y mejorar la precisión de las predicciones.

Una vez hecho esto, procederemos a configurar nuestro modelo para formar el árbol de decisión. Para ello, primero encontraremos aquellos parámetros, de entre unos fijados, que tienen mejor puntuación en la creación del modelo, y después fijaremos una semilla.

Finalmente implementaremos estos valores al modelo, para que este sea entrenado con nuestros conjuntos de entrenamiento y obtengamos un árbol de decisión. Cada nodo en el gráfico representará una decisión tomada por este. Los nodos internos muestran las

condiciones para tomar diferentes caminos en el árbol, mientras que las hojas representan las clases o valores de predicción finales.

Para visualizar como de buenas son las predicciones, primero sacaremos las métricas de evaluación de modelo comentadas anteriormente y luego realizaremos un gráfico de dispersión entre el valor real y el predicho, y otro que nos representará la distribución de los errores.

5.1.1. Benidorm Fest

Antes de hacer el árbol, sacamos la matriz de importancias para comprender el modelo:

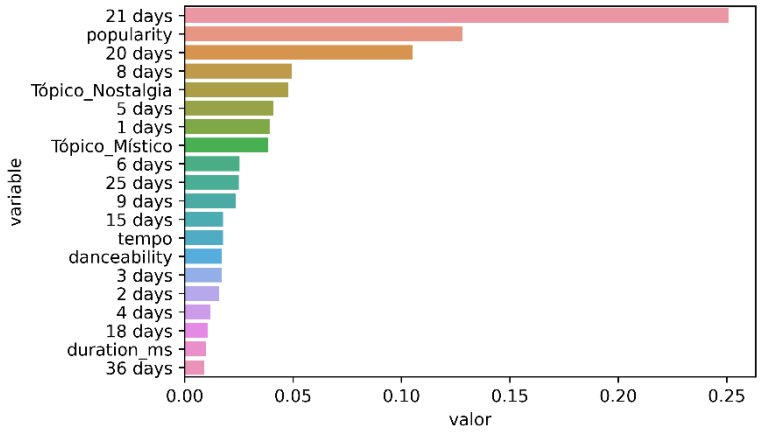


Figura 5.1. Matriz de importancias para el árbol de decisión del Benidorm Fest

Una lectura general que podemos hacer de esta figura es que, los valores de las cuotas a tres semanas de la final tienen más relevancia que las que están cerca de esta. También podemos ver que el parámetro que marca la popularidad de una canción en Spotify también tiene mucho peso, hecho que nos podíamos imaginar antes de modelizar.

Con esta información ya podemos pasar a la generación del árbol de decisión juntos a nuestros parámetros que maximizan la puntuación del modelo.

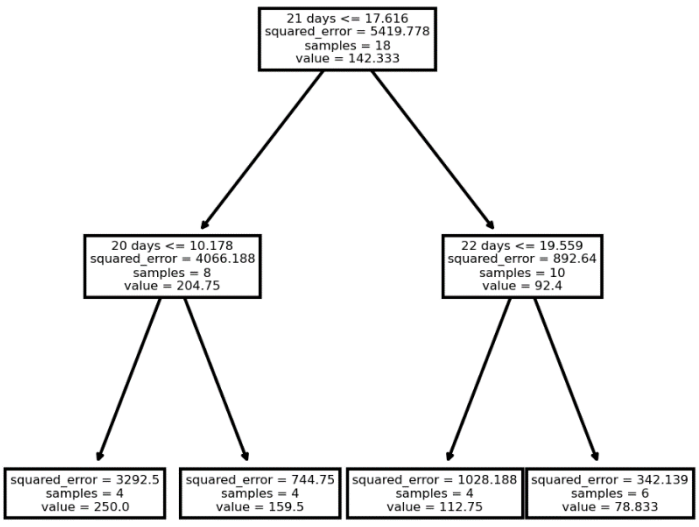


Figura 5.2. Árbol de decisión para el Benidorm Fest

En esta representación observamos de nuevo las características con más importancia y la variabilidad del valor predicho de los puntos totales dependiendo del camino que tomes.

De este modelo, hemos obtenido las siguientes métricas:

Conjunto	Entrenamiento	Prueba
MAE	29.1111	34.7166
MSE	1239.6990	1623.9208
RMSE	35.2093	40.2979
R2 Value	**0.7712**	

Tabla 5.1. Métricas tras el árbol de decisión en el Benidorm Fest

A partir de esta información podemos decir que el modelo está bien ajustado a los datos ya que tenemos un coeficiente de determinación cerca del 1, por lo que se explica un 77,12% de la variabilidad de los datos.

Para verificar estos datos, graficaremos los valores de las predicciones y sus errores:

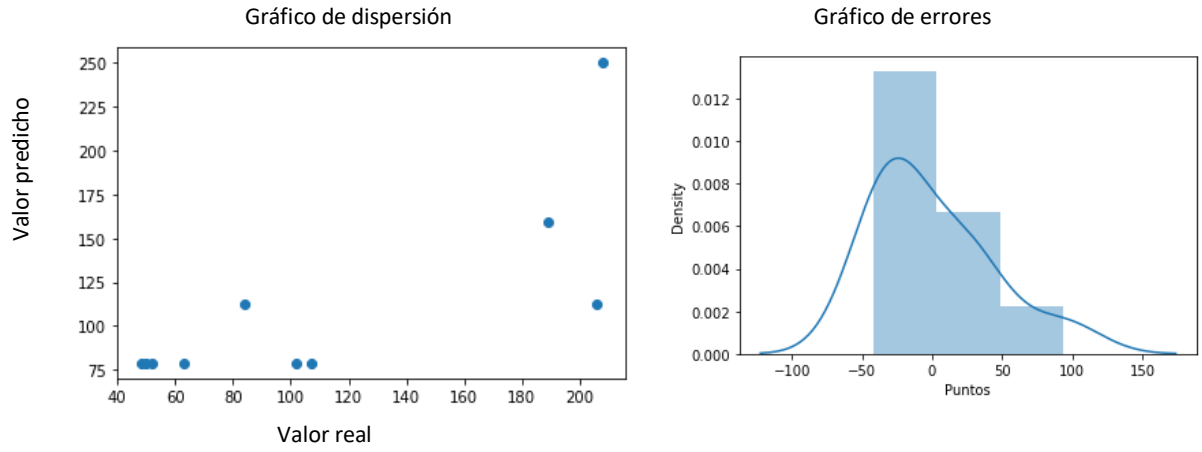


Figura 5.3. Gráficos para la visualización de las predicciones del árbol de decisión del Benidorm Fest

Para obtener la mejor predicción, deberíamos observar una línea diagonal en el gráfico de dispersión y una distribución de los errores simétrica y alrededor del valor 0. Podemos ver una tendencia de los puntos a acercarse a la diagonal, por lo que tenemos valores predichos razonables; también detectamos sesgo negativo en los errores ya que la distribución se desplaza hacia la izquierda, por lo que el modelo tiende a sobreestimar los valores reales.

En resumen, podemos declarar que el modelo no es del todo bueno, aunque ajuste bien los datos.

5.1.2. Melodifestivalen

De la misma manera que antes, observamos primero la matriz de importancias:

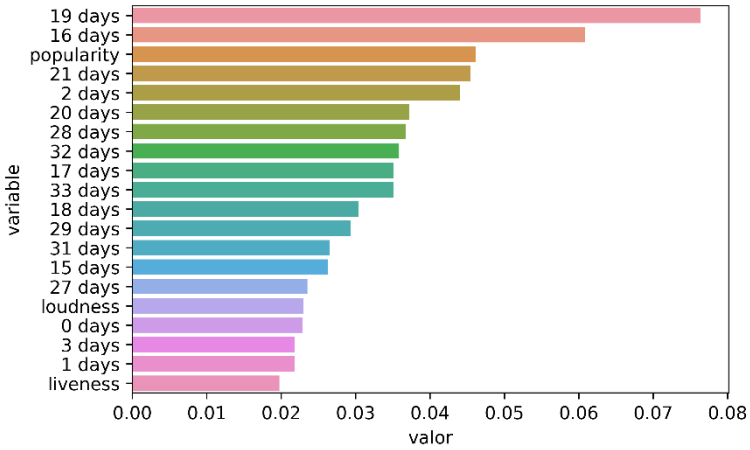


Figura 5.4. Matriz de importancias para el árbol de decisión del Melodifestivalen

Observamos el mismo patrón que en el Benidorm Fest, donde las cuotas de hace más de dos semanas y el parámetro de la popularidad tienen más peso. Ahora pasamos a observar el árbol de decisión creado.

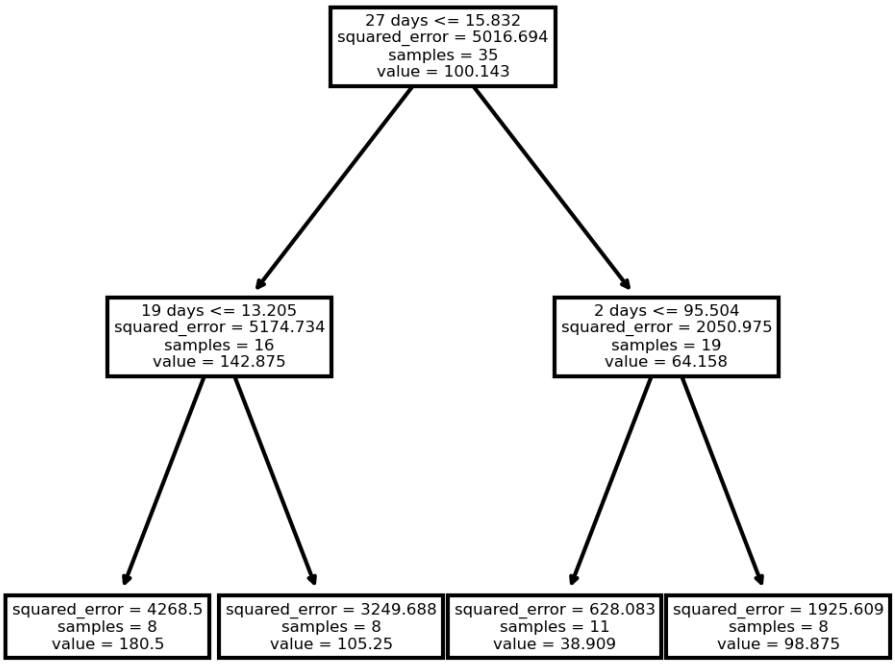


Figura 5.5. Árbol de decisión para el Melodifestivalen

Vemos de nuevo que las dos variables que se muestran en las hojas del árbol son de las que más peso tenían en la figura de antes.

De este árbol, hemos obtenido las siguientes métricas:

Conjunto	Entrenamiento	Prueba
MAE	40.7298	57.6539
MSE	2660.4408	5148.3984
RMSE	51.5795	71.7523
R2 Value	**0.4696**	

Tabla 5.2. Métricas tras el árbol de decisión en el Melodifestivalen

A partir de esta información podemos decir que el modelo no está muy bien ajustado a los datos ya que tenemos un coeficiente de determinación cerca del 0.5, por lo que se explica un solo un 47% de la variabilidad de los datos.

Volveremos a graficar las predicciones para sacar conclusiones:

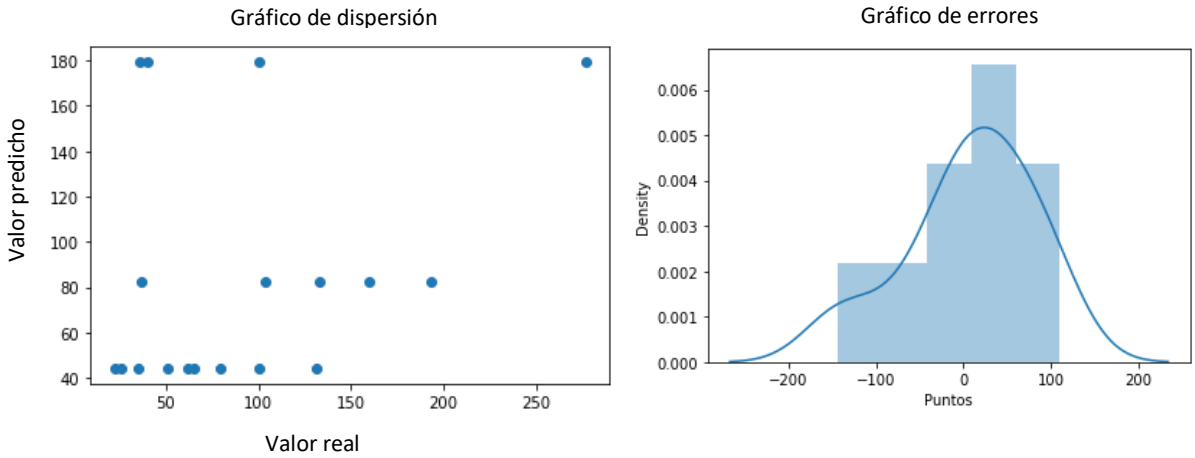


Figura 5.6. Gráficos para la visualización de las predicciones del árbol de decisión del Melodifestivalen

Podemos confirmar que obtenemos unas predicciones bastante flojas ya que la dispersión entre los valores es aleatoria y los errores tienen cola a la izquierda, por lo que vemos un caso de sobreestimación de los datos.

5.1.3. Eurovisión

Los resultados para el festival de Eurovisión son los siguientes:

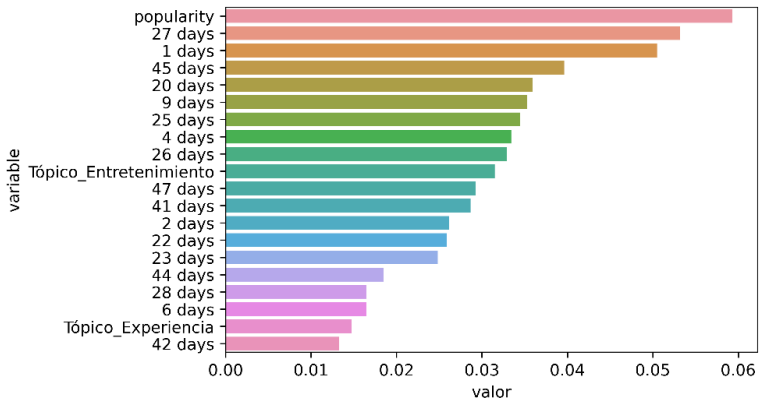


Figura 5.7. Matriz de importancias para el árbol de decisión de Eurovisión

Volvemos a obtener resultados parecidos a los festivales anteriores, con las cuotas antiguas y el parámetro de popularidad con mayor peso. Pero cabe destacar también que tenemos los valores de cuotas del último día de recogida, situación a tener en cuenta.

El árbol de decisión en este caso es el siguiente:

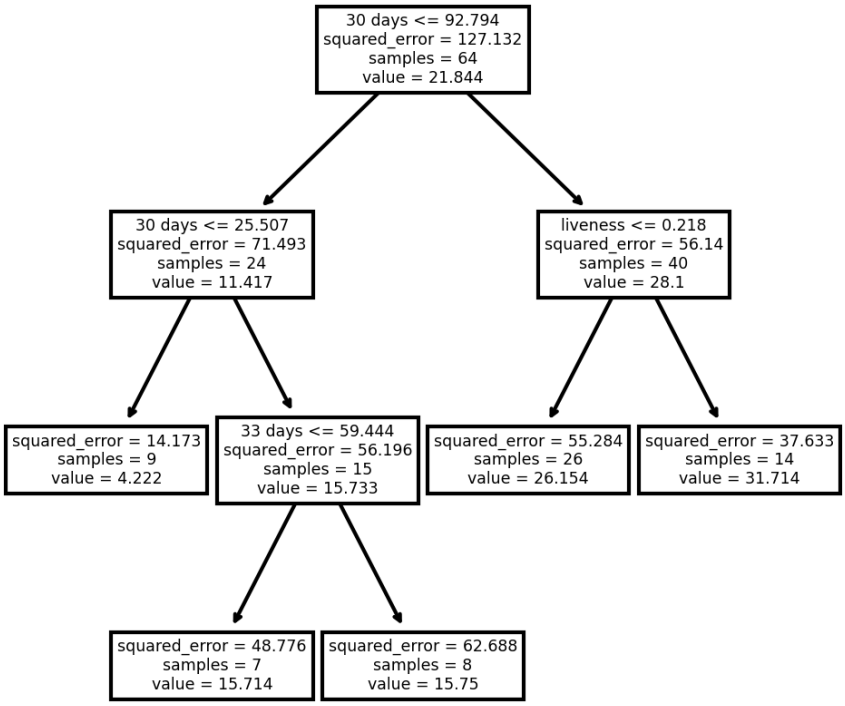


Figura 5.8. Árbol de decisión para Eurovisión

Podemos ver repetidas las variables más importantes en el árbol y, a su vez, los valores predichos.

Antes de ver las métricas, cabe destacar que los valores predichos que veremos a contnuación son de la variable posición, en vez de los puntos, de ahí que los ejes y resultados sean menores:

Conjunto	Entrenamiento	Prueba
MAE	5.2232	6.4137
MSE	45.8550	61.9672
RMSE	6.7716	7.8719
R2 Value	**0.6393**	

Tabla 5.3. Métricas tras el árbol de decisión en Eurovisión

En este caso podemos decir que el modelo está aceptablemente ajustando explicando la variabilidad de un 64% de los datos. También observamos que los errores también son más bajos que los que obteníamos en los festivales nacionales.

Para comprobar que el modelo se ajusta bien observamos de nuevo los gráficos:

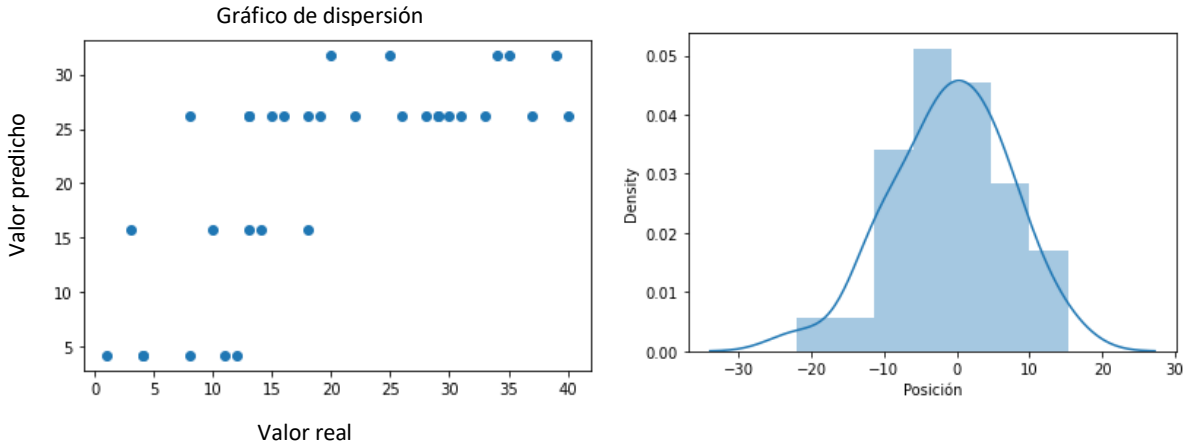


Figura 5.9. Gráficos para la visualización de las predicciones del árbol de decisión de Eurovisión

Observamos que los datos tienden a concentrarse alrededor de la diagonal comentada y los errores presentan una distribución bastante simétrica sin muchas anomalías destacables, por lo que podríamos decir que es un modelo decente a la hora de predecir.

5.2. XGBoost

XGBoost, abreviatura de *Extreme Gradient Boosting* (Refuerzo Extremo de Gradientes), es un algoritmo de aprendizaje automático ampliamente utilizado y potente. Se basa en la técnica de impulso de árboles de decisión y también destaca por su eficacia y rendimiento en problemas de clasificación y regresión. XGBoost utiliza un enfoque de ensamblaje de modelos que combina múltiples árboles de decisión débiles para crear un modelo más fuerte y preciso.

En este modelo fijaremos los mismos parámetros para todos los festivales y, mediante los conjuntos de entrenamiento de las variables independientes y dependiente, se creará el modelo y su predicción. Finalmente se evaluará de la misma manera que antes con la ayuda de métricas y gráficos.

5.2.1. Benidorm Fest

Antes de hacer el árbol múltiple, sacamos la matriz de importancias para comprender el modelo:

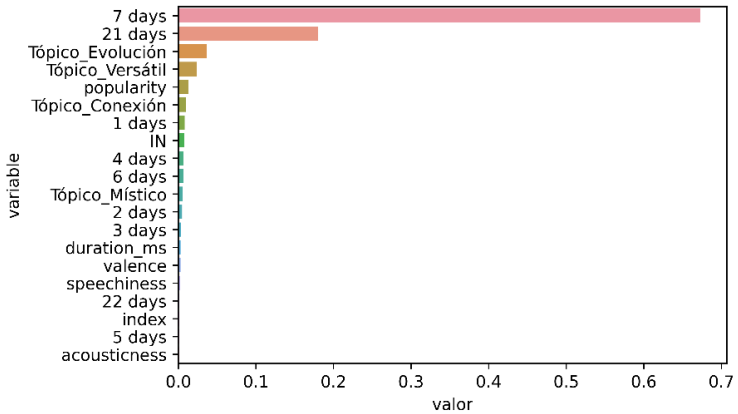


Figura 5.10. Matriz de importancias para el XGBoost del Benidorm Fest

En este caso, observamos de nuevo que las cuotas antiguas vuelven a tener peso, pero ahora son las de hace una semana las que obtienen mayor importancia. También encontramos varias categorías de la variable Tópico y otra vez el parámetro de popularidad de Spotify.

Al ser un árbol múltiple, las dimensiones que tendría su representación serían tan grandes que solo se verían borrones negros. Por esta razón mostraremos solo el primer árbol:

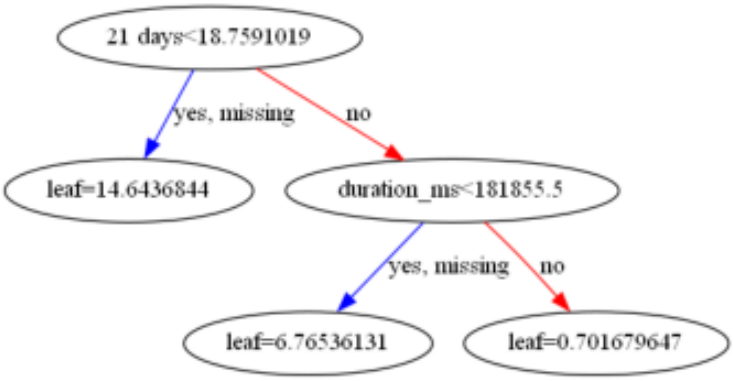


Figura 5.11. Muestra de árbol de decisión del XGBoost para el Benidorm Fest

Vemos la misma tendencia a representar las variables con más importancia en el modelo y el valor de la predicción en *leaf.*

Conjunto	**Entrenamiento**	**Prueba**
MAE	0.00032	46.8787
MSE	1.6299	2987.5573
RMSE	0.0004	54.6585

Otras	
R2 Value	0.9999
Average MAE	41.298 (29.493)

Tabla 5.4. Métricas tras el XGBoost en el Benidorm Fest

Como ya podemos ver a primera vista, tenemos un coeficiente de determinación muy alto para una precisión baja. También observamos que los errores para el conjunto de entrenamiento son prácticamente nulos. Esto es debido a una de las cualidades de este tipo de modelo, el sobreajuste que hace de estos datos y, por ende, la no generalización en nuevos datos. Además, vemos en el MAE promedio, entre paréntesis, que se pierden casi 30 puntos de media en la predicción.

Para ver el efecto que tiene esto en las predicciones, mostramos los siguientes gráficos:

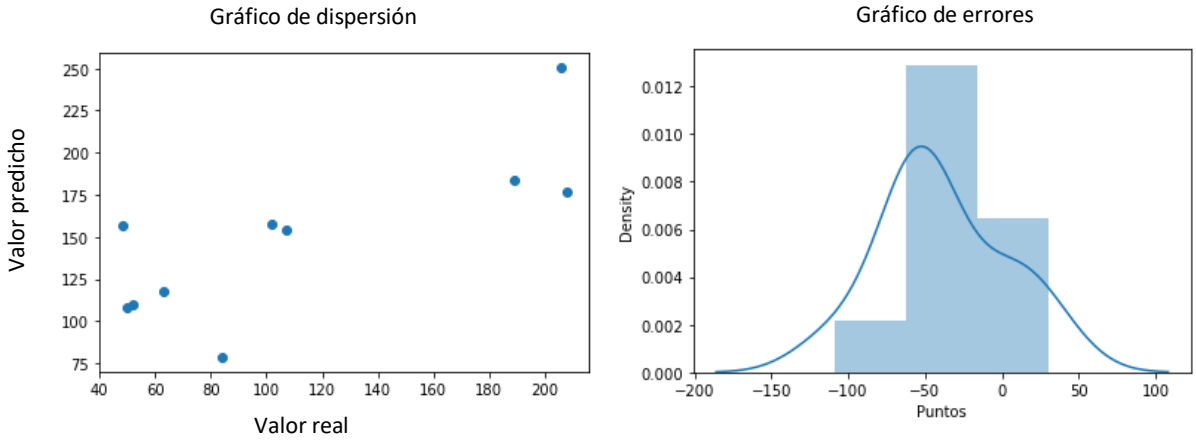


Figura 5.12. Gráficos para la visualización de las predicciones del XGBoost del Benidorm Fest

Vemos que la dispersión de los puntos entre valores reales y predichos es un poco mejor que la que precisábamos en el árbol de decisión, pero como hemos dicho, hay mucho sobreajuste por lo que la distribución de los errores está más desplazada a la izquierda, provocando más sesgo en los resultados por sobreestimación.

5.2.2. Melodifestivalen

Volvemos a observar la matriz de importancias para este modelo:

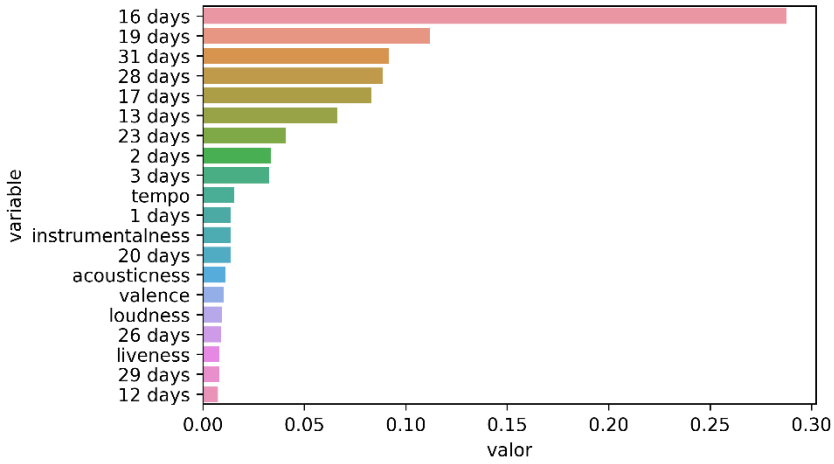


Figura 5.13. Matriz de importancias para el XGBoost del Melodifestivalen

Podemos ver que las cuotas antiguas, más específicamente las comprendidas entre las tres y cuatro semanas, son las que mayor peso tienen en el modelo.

A continuación, vemos uno de los árboles del modelo ya que no podemos observarlos todos:

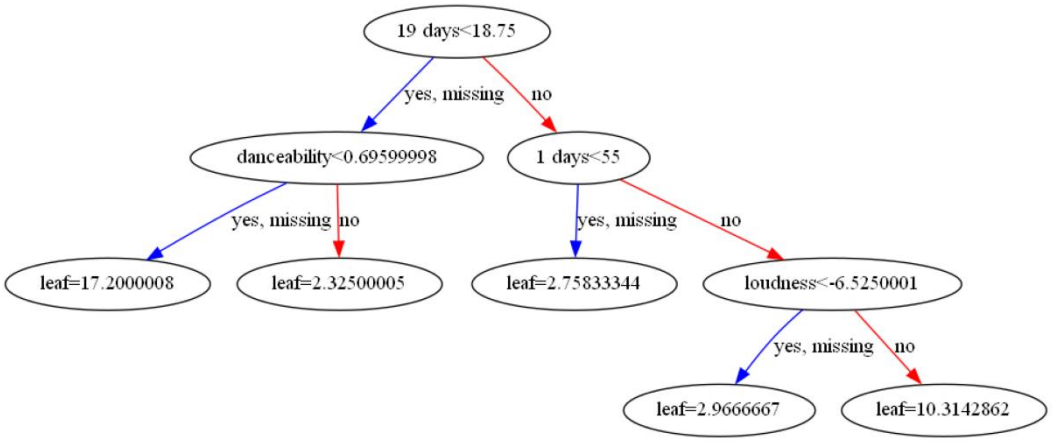


Figura 5.14. Muestra de árbol de decisión del XGBoost para el Melodifestivalen

Las métricas en este caso serían las siguientes:

Conjunto	Entrenamiento	Prueba
MAE	0.00025	39.7594
MSE	1.2365	2713.7635
RMSE	0.0003	52.0937

Otras	
R2 Value	0.9999
Average MAE	45.594 (14.334)

Tabla 5.5. Métricas tras el XGBoost en el Melodifestivalen

Tenemos otro caso de sobreajuste de los datos de entrenamiento, pero para este festival solo perdemos 14 puntos de media en las predicciones, por lo que tenemos que mirar los gráficos para comprobar si los resultados son mejores.

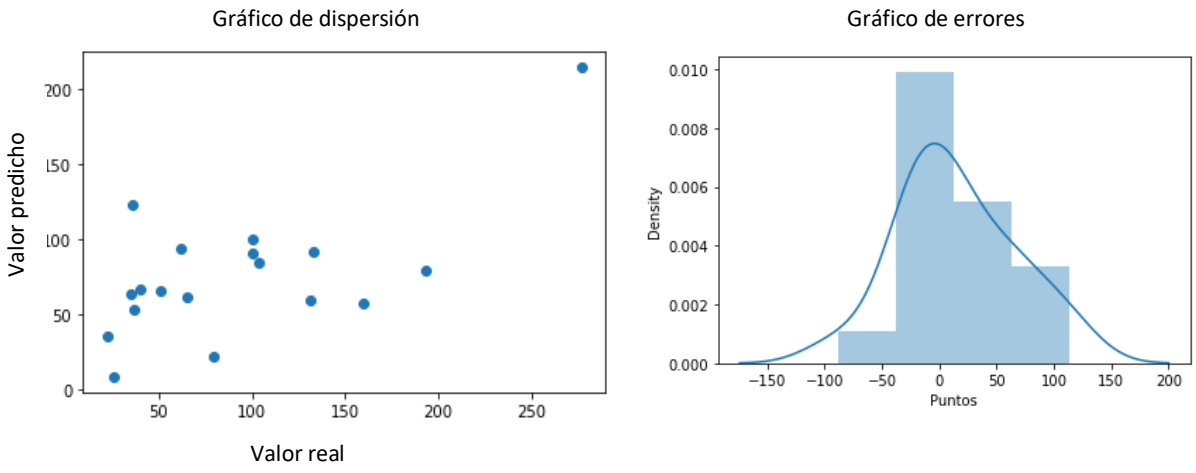


Figura 5.15. Gráficos para la visualización de las predicciones del XGBoost del Melodifestivalen

Podemos ver como los puntos en el gráfico de dispersión se distribuyen mucho más cercanos a la diagonal comentada, aunque veamos que la distribución de los errores está desplazada a la derecha, lo que tiende a subestimar las predicciones.

5.2.3. Eurovisión

Observamos de nuevo la matriz de importancias:

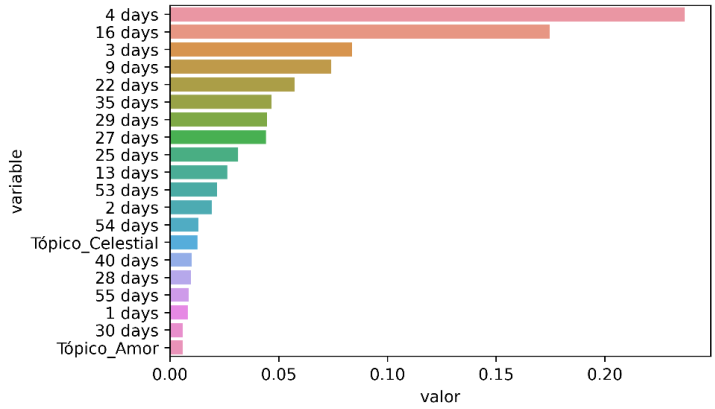


Figura 5.16. Matriz de importancias para el XGBoost de Eurovisión

En este caso vemos que los patrones vistos anteriormente no se repiten ya que las variables con más peso son las cuotas de la última semana antes del festival. Esto es el resultado que nos podríamos imaginar en un inicio ya que los días previos antes de la final siempre hay más movimiento en el mercado y son los valores que suelen acertar más los ganadores.

A continuación, vemos una extracción del árbol múltiple del XGBoost:

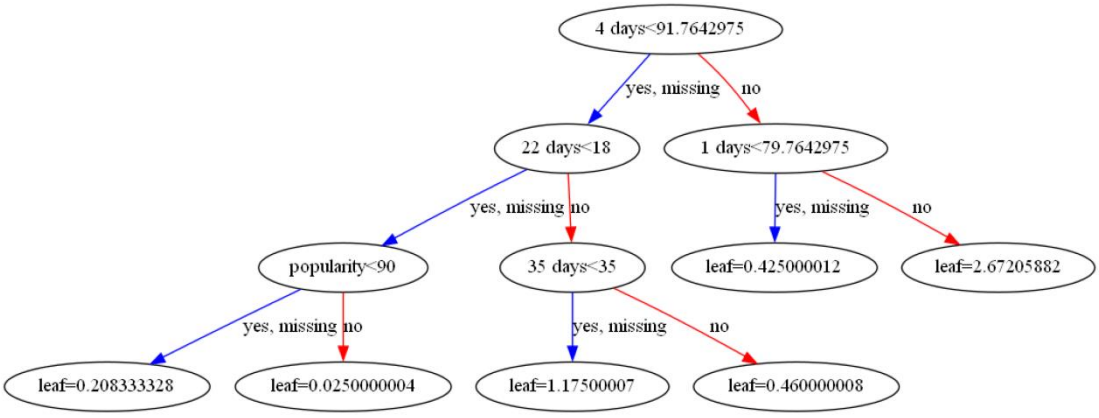


Figura 5.17. Muestra de árbol de decisión del XGBoost para Eurovisión

Vemos lo que comentábamos anteriormente, el árbol inicia en base a la variable con mayor peso, que son las cuotas a 4 días de la final. Las métricas obtenidas del modelo son:

Conjunto	Entrenamiento	Prueba
MAE	0.06677	4.5963
MSE	0.06677	35.8145
RMSE	0.2588	5.9845

Otras	
R2 Value	0.99947
Average MAE	6.034 (1.593)

Tabla 5.6. Métricas tras el XGBoost en Eurovisión

Volvemos a tener el sobreajuste en los datos de entrenamiento, pero esta vez solo perdemos entre una y dos posiciones de media a la hora de predecir. A priori, viendo los errores parece un buen modelo, pero volveremos a graficar para verificarlo.

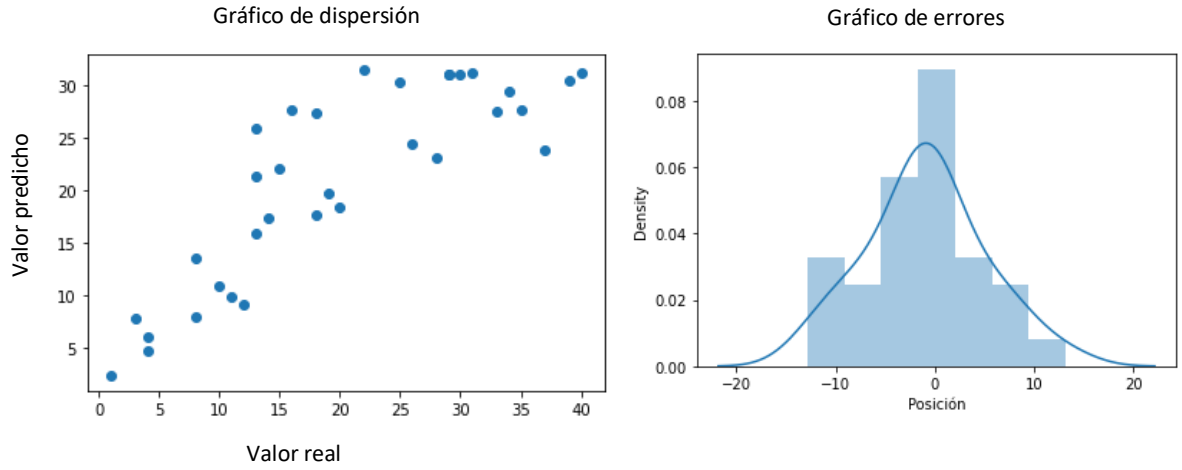


Figura 5.18. Gráficos para la visualización de las predicciones del XGBoost de Eurovisión

Vemos claramente que la dispersión de los valores es muy buena ya que están casi todos sobre la diagonal, y los errores presentan una distribución de campana sin anomalías.

5.3. Red neuronal

Una red neuronal es un modelo de predicción basado en el cerebro humano, compuesto por capas de neuronas interconectadas. Durante el entrenamiento, los pesos de las conexiones se ajustan para minimizar la diferencia entre las predicciones y los valores reales. Al recibir datos de entrada, la red realiza cálculos y aplica funciones de activación en cada neurona para generar una predicción. El objetivo será aprender patrones complejos y modelar relaciones no lineales.

La red neuronal implementada en este proyecto será un modelo de regresión que utilizará una arquitectura secuencial. Primero analizaremos el resumen según la densidad fijada para después proceder al ajuste del modelo, donde impondremos 30000 iteraciones para buscar el mejor, y finalmente obtener las predicciones y métricas evaluables.

Para programarla usaremos el paquete que hemos descrito en el primer capítulo: Keras. Este tiene un alto nivel para la construcción y el entrenamiento de redes neuronales en Python

5.3.1. Benidorm Fest

El resumen para el modelo de este festival es el siguiente:

Capa (tipo)	Forma	Parámetros
Densidad_3	300	69900
Densidad_4	25	7525
Densidad_5	1	26

Parámetros totales	77451
Parámetros entrenables	77451
Parámetros no entrenables	0

Tabla 5.7. Resumen del modelo por redes neuronales del Benidorm Fest

Podemos confirmar que el modelo se ha configurado correctamente ya que tenemos que todos los parámetros son entrenables.

Las métricas obtenidas son las siguientes:

Conjunto	**Entrenamiento**	**Prueba**
MAE	2.9051	32.9902
MSE	16.8496	1259.9752
RMSE	4.1048	35.4961
R2 Value	**0.9968**	

Tabla 5.8. Métricas tras la red neuronal en el Benidorm Fest

Observamos dos situaciones buenas en estos resultados, tenemos un coeficiente de determinación muy bueno de los datos sin presentar sobreajuste, donde se explica casi toda la variabilidad de los datos, y obtenemos unos errores no muy grandes.

A continuación, graficamos las predicciones para comprobar lo que se ha comentado:

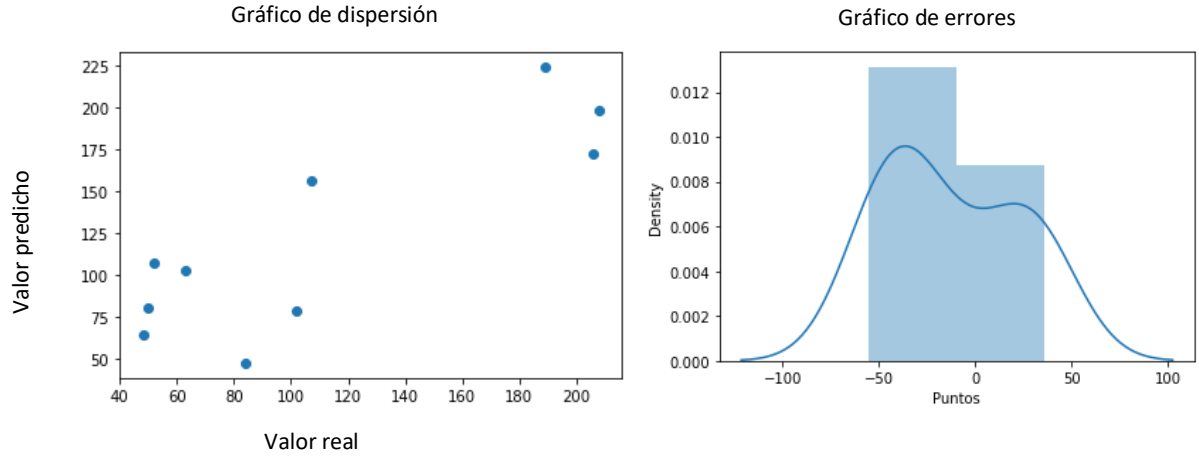


Figura 5.19. Gráficos para la visualización de las predicciones de la red neuronal del Benidorm Fest

Para el gráfico de la dispersión, aunque no se vea claramente, se nota la tendencia de los valores a acercarse a la diagonal. En el de los errores vemos que la distribución está un poco desplazada a la izquierda, por lo que puede haber sesgo por sobreestimación de los valores.

5.3.2. Melodifestivalen

El resumen para el modelo de este festival es el siguiente:

Capa (tipo)	Forma	Parámetros
Densidad	300	74100
Densidad_1	25	7525
Densidad_2	1	26

Parámetros totales	81651
Parámetros entrenables	81651
Parámetros no entrenables	0

Tabla 5.9. Resumen del modelo por redes neuronales del Melodifestivalen

Podemos confirmar que el modelo se ha configurado correctamente ya que tenemos que todos los parámetros son entrenables.

Las métricas obtenidas son las siguientes:

Conjunto	Entrenamiento	Prueba
MAE	2.7516	27.4673
MSE	11.7023	1039.3079
RMSE	3.4208	32.2382
R2 Value	**0.9976**	

Tabla 5.10. Métricas tras la red neuronal en el Melodifestivalen

Observamos unas métricas muy buenas ya que los errores no son muy altros tanto para los datos de entrenamiento como para los de prueba y el ajuste del modelo es muy bueno.

A continuación, graficamos las predicciones para comprobar lo que se ha comentado:

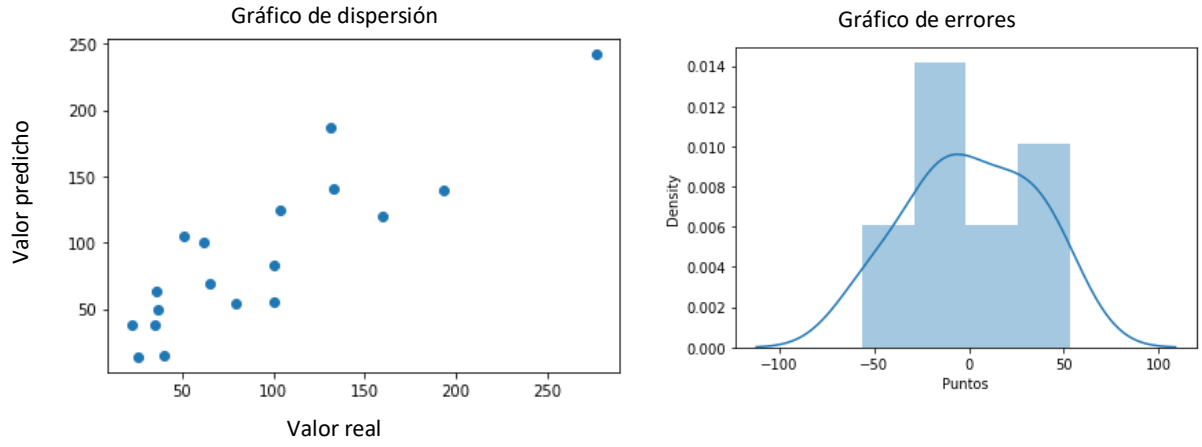


Figura 5.20. Gráficos para la visualización de las predicciones de la red neuronal del Melodifestivalen

A partir de los gráficos, vemos que la dispersión de los puntos está muy pareja a la diagonal y, sobre todo, tenemos una distribución de los errores casi simétrica y alrededor de cero.

5.3.3. Eurovisión

El resumen para el modelo de este festival es el siguiente:

Capa (tipo)	Forma	Parámetros
Densidad_6	300	27900
Densidad_7	25	7525
Densidad_8	1	26
Parámetros totales	35451	
Parámetros entrenables	35451	
Parámetros no entrenables	0	

Tabla 5.11. Resumen del modelo por redes neuronales de Eurovisión

Podemos confirmar que el modelo se ha configurado correctamente ya que tenemos que todos los parámetros son entrenables.

Las métricas obtenidas son las siguientes:

Conjunto	Entrenamiento	Prueba
MAE	1.0559	6.9593125545617305
MSE	1.5274	100.59671864022545
RMSE	1.2358	10.02979155517329
R2 Value	**0.9879**	

Tabla 5.12. Métricas tras la red neuronal en Eurovisión

A priori, obtenemos unas métricas muy buenas donde se minimizan los errores en los datos y el modelo explica casi un 99% de la variabilidad de estos, sin presentar sobreajuste como el modelo anterior.

A continuación, graficamos las predicciones para comprobar lo que se ha comentado:

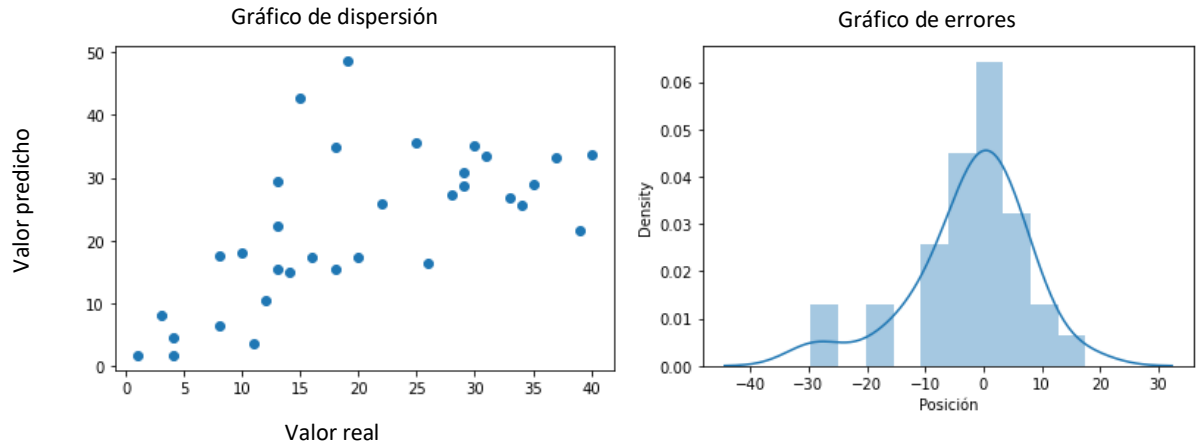


Figura 5.21. Gráficos para la visualización de las predicciones de la red neuronal de Eurovisión

Vemos en estos gráficos que la dispersión de los puntos está entorno a la diagonal, por lo que obtenemos buenas predicciones. Aunque también cabe remarcar una pequeña cola a la izquierda en la distribución de los errores, lo que puede sobreestimar los valores.

5.4. Resultados

Después de observar la implementación de todos los modelos descritos en cada festival, obtenemos varias conclusiones a anotar:

- En todas las matrices de importancia, las cuotas son las que más peso han tenido continuamente. Esto lo podríamos considerar como un hecho normal debido a que siempre tienen la última palabra a la hora de otorgar una probabilidad u otra a un concursante, pero también debemos volver a remarcar el hecho de que hemos creado los modelos mediante datos no estandarizados ya que obteníamos mejores resultados con estos. Ya que las cuotas son las únicas que pueden tomar valores muy alejados al del resto de variables de la base de datos, esto puede ocasionar que estas tiendan a sobreponerse más.
- La red neuronal es la que se ajusta mejor a los datos y predice mejor los resultados. Esta conclusión era la prevista antes del procedimiento ya que la red neuronal supera en muchos aspectos a los otros dos modelos, pero su capacidad para detectar patrones complejos a partir de aprender y modelar relaciones no lineales, la hacen destacar sobre el resto.

Dicho esto, veremos las predicciones de las ediciones de 2023 de cada festival usando la red neuronal sobre los datos:

Benidorm Fest 2023	*Puntos reales*	*Puntos predichos*
Alfred García	102	87.133
Alice Wonder	208	207.246
Agoney	306	286.456
Aritz Aren	107	172.187
Blanca Paloma	336	335.484
E'Femme	64	78.4764
Famous	87	96.2885
Fusa Nocta	189	239.58
José Otero	180	192.094
Karmento	192	198.661
Megara	217	233.266
Meler	84	52.5696
Rakky Ripper	52	124.849
Sharonne	87	96.8319
Siderland	88	99.8641
Sofía Martín	48	80.8595
Twin Melody	77	83.8087
Vicco	264	277.763

Tabla 5.13. Comparativa valores reales contra valores predichos para el Benidorm Fest 2023

Aunque para algunos valores vemos mucha disparidad respecto a la realidad, obtenemos buenas predicciones y, lo más destacable, acertamos de lleno la puntuación de la ganadora Blanca Paloma.

Melodifestivalen 2023	*Puntos reales*	*Puntos predichos*
Axel Schylström	55	46.0732
Casanovas	29	29.0762
Eden	12	14.0005
Elov & Beny	31	34.1826
Emil Henrohn	26	13.706
Fjällgren, North & Woods	181	184.217
Ida-Lova	36	62.9541
Kiana	145	144.954
Laurell	30	34.0208
Loreen	277	242.105
Marcus & Martinus	238	236.957
Maria Sur	147	142.673
Mariette	111	113.081
Melanie Wehbe	40	14.1952
Nordman	119	118.876
Panetoz	131	187.46
Paul Rey	142	145.434
Signe & Hjördis	27	30.0434
Smash Into Pieces	191	192.827
Tennessee Tears	62	100.109
Theoz	156	154.49
Tone Sekelius	114	114.14
Uje Brandelius	22	37.5653
Victor Crone	54	47.7702
Wiktoria	47	44.5996

Tabla 5.14. Comparativa valores reales contra valores predichos para el Melodifestivalen 2023

Nos encontramos con la misma situación que hemos comentados justo antes, hay algunos valores que distan de la realidad, pero la gran mayoría son predicho con alta precisión, sobre todo la participante de interés que es Loreen, la ganadora.

Final Eurovisión 2023	*Posición real*	*Posición predicha*
Suecia con Loreen	1	1.693
Finlandia con Käärijä	2	3.521
Israel con Noa Kirel	3	2.457
Italia con Marco Mengoni	4	6.563
Noruega con Alessandra	5	2.469
Ucrania con Tvorchi	6	8.201
Bélgica con Gustaph	7	9.698
Estonia con Alika	8	11.123
Australia con Voyager	9	6.236
Chequia con Vesna	10	12.036
Lituania con Monika	11	13.987
Chipre con Andrew Lambrou	12	15.236
Croacia con Let 3	13	13.789
Armenia con Brunette	14	12.369
Austria con Teya & Salena	15	15.324
Francia con La Zarra	16	20.236
España con Blanca Paloma	17	17.589
Moldavia con Pasha Parfeni	18	15.256
Polonia con Blanka	19	20.698
Suiza con Remo Forrer	20	18.354
Eslovenia con Joker Out	21	22.752
Albania con Albina & Familja Kelmendi	22	26.634
Portugal con Mimicat	23	21.987
Serbia con Luke Black	24	26.963
Reino Unido con Mae Muller	25	23.751
Alemania con Lord Of The Lost	26	28.463

Tabla 5.15. Comparativa valores reales contra valores predichos para Eurovisión 2023

En líneas generales, obtenemos buenas predicciones para la final de Eurovisión 2023, donde también damos por ganadora a Suecia con Loreen.

En conclusión, podemos afirmar que se obtienen buenas predicciones para todos los festivales, donde se consigue predecir el ganador/a de cada uno.

6. Conclusiones

En resumen, en este proyecto se ha conseguido realizar cada uno de los objetivos marcados en un inicio. Fruto del esfuerzo y dedicación para encontrar la mayor cantidad de cursos y ejercicios en internet, se ha logrado alcanzar un buen nivel de programación en Python, lo cual nos ha facilitado mucho más la realización de códigos y la solvencia de errores.

A nivel personal, la realización de este trabajo ha mejorado a gran escala las habilidades que se presentaban des de un inicio, aprendiendo a usar un nuevo programa y, a la vez, realizar un análisis estadístico con una amplia información.

Aunque la fase de investigación y creación de las bases de datos pueda quedar en un segundo plano, cabe destacar que es lo que más tiempo ha conllevado hacer y podemos decir que se ha adquirido el conocimiento de usar distintas técnicas de minería de datos que no se habían visto en la carrera. Esto último ha añadido un nivel de dificultad más a la hora de comprender los pasos a seguir y los resultados que se obtenían, pero también se ha logrado superar.

Entrando en el tema que atrae más las miradas, podemos decir que se ha logrado hacer uso de distintos modelos de aprendizaje automático para realizar las mejores predicciones de cada festival estudiado. Como se ha dicho en el capítulo anterior, se han obtenido unos resultados predichos muy cercanos a la realidad a partir de la red neuronal.

A partir de todas estas anotaciones, podemos decir que las expectativas del proyecto se han cumplido y los resultados obtenidos son satisfactorios.

7. Líneas futuras

Este apartado va dedicado a las posibles direcciones o áreas de investigación que podrían explorarse en el futuro basándose en los resultados y hallazgos del trabajo actual.

La primera y la más importante de todas es que se solamente hemos usado tres modelos de aprendizaje automático para sacar las mejores predicciones. Como vemos en los resultados, estas han sido muy buenas, pero para conseguir definir el mejor modelo para ello tendríamos que seguir aplicando nuevas técnicas y probar si, estandarizando finalmente los datos, se logran mejores predicciones.

También podríamos haber usado métodos de ensamblaje para combinar múltiples componentes o módulos de software en un conjunto coherente. A partir de ello lograríamos mejorar mucho la creación de modelos más ajustados.

Otro aspecto a tener en cuenta es el tiempo que se ha tenido para la realización del proyecto, ya que debido a que se debía hacer un primer paso de aprendizaje autónomo y des de cero del lenguaje de programación Python, hemos tomado vías rápidas para poder llegar al último paso de predecir y evaluar. Algunas de ellas han sido filtrar las bases de datos finales para que no tengamos valores faltantes una vez hecha la concatenación de años o fijar solo las cuotas para una casa de apuestas.

En definitiva, todos estos enfoques comentados podrías mejorar los resultados si en un futuro continuáramos el proyecto.

BIBLIOGRAFIA

- Agencia de Marketing Online BlackBeast. (2019, 8 julio). *¿Qué son las Stop Words y para qué sirven? – BlackBeast*. Blackbeast.pro. [**https://blackbeast.pro/diccionario/stop-words/**]
- Arroyo, I. (2022b). Spotipy: Extraer características de canciones. *tacosdedatos*. [**https://www.tacosdedatos.com/unisaacarroyov/spotipy-extraer-caracteristicas-de-canciones-9km**]
- Aznar, P. (2011). Spotify pone a disposición de los desarrolladores las herramientas para incluir su reproductor en. . . *Applesfera*. [**https://www.applesfera.com/aplicaciones-ios-1/spotify-pone-a-disposicion-de-los-desarrolladores-las-herramientas-para-incluir-su-reproductor-en-aplicaciones-ios#:~:text=Las%20API%20son%20herramientas%20%E2%80%93%20librer%C3%ADas,para%20acceder%20a%20sus%20productos**]
- Bansal, S. (2023). Beginners Guide to Topic Modeling in Python. *Analytics Vidhya*. [**https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/**]
- *BeniCalculadora 2023: Simula las puntuaciones del BenidormFest - eurovision-spain.com*. (2023, 27 enero). eurovision-spain.com. [**https://eurovision-spain.com/blogs/benicalculadora-2023-simula-las-puntuaciones-del-benidormfest/?fbclid=PAAaaimTkXk--Va8mbgya6HQGzRiPAhjEVLSEjo1fjc2OqBpDn624LHqNR3Rc**]
- Brownlee, J. (2020). How to Visualize Gradient Boosting Decision Trees With XGBoost in Python. MachineLearningMastery.com. [**https://machinelearningmastery.com/visualize-gradient-boosting-decision-trees-xgboost-python/**]
- *Choosing Colormaps in Matplotlib — Matplotlib 3.7.1 documentation*. (s. f.). [**https://matplotlib.org/stable/tutorials/colors/colormaps.html**]
- colaboradores de Wikipedia. (2022). Árbol de decisión. Wikipedia, la enciclopedia libre. [**https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n**]
- Contributors to Wikimedia projects. (2023). XGBoost. Viquipèdia, l'enciclopèdia lliure. [**https://ca.wikipedia.org/wiki/XGBoost**]
- colaboradores de Wikipedia. (2023). Apuesta deportiva. Wikipedia, la enciclopedia libre. [**https://es.wikipedia.org/wiki/Apuesta_deportiva**]
- colaboradores de Wikipedia. (2023). Python. *Wikipedia, la enciclopedia libre*. [**https://es.wikipedia.org/wiki/Python**]
- colaboradores de Wikipedia. (2023a). Festival de la Canción de Eurovisión. *Wikipedia, la enciclopedia libre*. [**https://es.wikipedia.org/wiki/Festival_de_la_Canci%C3%B3n_de_Eurovisi%C3%B3n**]
- colaboradores de Wikipedia. (2023a). Melodifestivalen. *Wikipedia, la enciclopedia libre*. [**https://es.wikipedia.org/wiki/Melodifestivalen**]
- datos.gob.es. (2022, 2 agosto). 10 Librerías populares de procesamiento del lenguaje natural. *datos.gob.es*. [**https://datos.gob.es/es/blog/10-librerias-populares-de-procesamiento-del-lenguaje-natural**]
- De Los Datos, E. M. (2021, 25 marzo). *Introducción al topic modeling con Gensim (I): fundamentos y preprocesamiento de textos*. El mundo de los datos. [**https://elmundodelosdatos.com/topic-modeling-gensim-fundamentos-preprocesamiento-textos/**]
- De Los Datos, E. M. (2021b, marzo 31). *Introducción al topic modeling con Gensim (II): asignación de tópicos*. El mundo de los datos. [**https://elmundodelosdatos.com/topic-modeling-gensim-asignacion-topicos/**]
- Diezminutos.es. (2023, 12 mayo). Eurovisión 2023: cambios en el sistema de votación. *Diez Minutos*. [**https://www.diezminutos.es/teleprograma/series-tv/a42239770/sistema-votacion-eurovision-2023/**]
- Eurovisión, E. (2023, 23 marzo). Eurovisión 2023 | Revelado el orden de actuación de las semifinales. *RTVE.es*. [**https://www.rtve.es/television/20230322/revelado-orden-actuacion-semifinales-eurovision-2023/2432531.shtml**]

- Fernandez, R. (2019, 6 septiembre). *Predicción con Series Temporales con LSTM, Redes neuronales recurrentes - ▷ Cursos de Programación de 0 a Experto © Garantizados*. ▷ Cursos de Programación de 0 a Experto © Garantizados. [**https://unipython.com/prediccion-con-series-temporales-con-lstm-redes-neuronales-recurrentes/**]
- Fernandez, R. (2019, 6 septiembre). *Predicción con Series Temporales con LSTM, Redes neuronales recurrentes - ▷ Cursos de Programación de 0 a Experto © Garantizados.* ▷ Cursos de Programación de 0 a Experto © Garantizados. [**https://unipython.com/prediccion-con-series-temporales-con-lstm-redes-neuronales-recurrentes/**]
- *Genius API Documentation*. (s. f.). **[https://docs.genius.com/#/getting-started-h1**]
- *geopandas.datasets.get_path — GeoPandas 0.13.2+0.gd5add48.dirty documentation*. (s. f.). [**https://geopandas.org/en/stable/docs/reference/api/geopandas.datasets.get_path.html**]
- Ghanoum, T. (2022, 30 marzo). Topic Modelling in Python with spaCy and Gensim - Towards Data Science. *Medium*. [**https://towardsdatascience.com/topic-modelling-in-python-with-spacy-and-gensim-dc8f7748bdbf**]
- Gomez, O. A. (s. f.). *Visualizando Grafos usando Graphviz*. OSiUX. [**https://osiux.com/visualizando-grafos-graphviz.html**]
- GraphEverywhere, E. (2019). Algoritmo de Louvain. GraphEverywhere. [**https://www.grapheverywhere.com/algoritmo-de-louvain/**]
- *graphviz*. (2022, 23 julio). PyPI. [**https://pypi.org/project/graphviz/**]
- Heras, J. M. (2020, 10 octubre). 15 Librerías de Python para Machine Learning - IArtificial.net. *IArtificial.net*. [**https://www.iartificial.net/librerias-de-python-para-machine-learning**]
- *IBM Documentation*. (s. f.). [**https://www.ibm.com/docs/es/spss-modeler/18.2.0?topic=tab-handling-outliers-extreme-values**]
- KeepCoding, R. (2023, 22 febrero). ¿Qué es el topic modeling? | KeepCoding Bootcamps. *KeepCoding Bootcamps*. [**https://keepcoding.io/blog/que-es-el-topic-modeling/**]
- KeepCoding, R. (2023b, mayo 23). Entrenar un modelo LDA por medio de un ejercicio de topic modeling. *KeepCoding Bootcamps*. [**https://keepcoding.io/blog/entrenar-un-modelo-lda-con-topic-modeling/**]
- Lizana, A. M. (2023, 31 enero). Benidorm Fest 2023: Orden de actuaciones de la Semifinal 1. *FormulaTV*. [**https://www.formulatv.com/noticias/benidorm-fest-2023-orden-actuaciones-semifinal-1-120759/**]
- Maldeadora. (2017). Cuatro librerías de Machine Learning: TensorFlow, Scikit-learn, Pytorch y Keras. *Platzi*. [**https://platzi.com/blog/librerias-de-machine-learning-tensorflow-scikit-learn-pythorch-y-keras/**]
- Mimi. (2022, 7 febrero). Crear un GIF con imágenes - imageio python. *kipunaEc*. [**https://noemioocc.github.io/posts/Crear-un-gif-con-imagenes-imageio-python/**]
- Naveen. (2023). Pandas apply() with Lambda Examples. *Spark By {Examples}*. [**https://sparkbyexamples.com/pandas/pandas-apply-with-lambda-examples/**]
- *Newest «python» Questions*. (s. f.). Stack Overflow. [**https://stackoverflow.com/questions/tagged/python**]
- Płoński, P. (2020, 22 junio). Visualize a Decision Tree in 4 Ways with Scikit-Learn and Python. MLJAR. [**https://mljar.com/blog/visualize-decision-tree/**]
- *PRESELECCIÓN ESPAÑA 2023*. (2023, 19 mayo). Ogae Spain. [**https://www.ogaespain.com/preselecciones/preselecciones-2023/preseleccion-espana-2023/**]
- *PRESELECCIÓN SUECIA 2023*. (2023, 19 mayo). Ogae Spain. [**https://www.ogaespain.com/preselecciones/preselecciones-2023/preseleccion-suecia-2023/**]
- Research, I. (2022, 16 mayo). Estado del arte ¿Qué es y qué permite? - Infinitia Research. *INFINITIA Industrial Consulting*.
- [**https://www.infinitiaresearch.com/noticias/que-es-el-estado-del-arte-y-que-permite/**]
- Rueda, J. M. F. (s. f.). *9 Topic Modeling | Cuentapalabras*. [**http://www.aic.uva.es/cuentapalabras/topic-modeling.html**]

- Rvaquerizo, & Rvaquerizo. (2021, 25 octubre). Animación de un mapa con Python. Porcentaje de vacunas administradas - Análisis y Decisión. *Análisis y Decisión - Transformando datos en decisiones*. [**https://analisisydecision.es/animacion-de-un-mapa-con-python-porcentaje-de-vacunas-administradas/**]
- sklearn.tree.plot_tree. (s. f.). scikit-learn. [**https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html**]
- *Spotify*. (s. f.). Playlist Benidorm Fest 2023 [**https://open.spotify.com/playlist/37i9dQZF1DXcohULCINaEo?si=97b9e03b132a42ee**]
- *Spotify*. (s. f.-b). Playlist Melodifestivalen 2023 [**https://open.spotify.com/playlist/2Bk4Q2b8ZunM8pBbVpg0bI?si=067da89016094b46**]
- *Spotify*. (s. f.-c). Playlist Eurovisión 2023 [**https://open.spotify.com/playlist/37i9dQZF1DWVCKO3xAlT1Q?si=fe8a0d4bb91244a2**]
- StackPath. (s. f.). [**https://www.luckia.es/blog/cuota/#:~:text=%C2%BFQu%C3%A9%20es%20la%20cuota%3F,va%20a%20ser%20nuestra%20ganancia**.]
- *Sweden: Melodifestivalen 2023*. (s. f.). Eurovisionworld. [**https://eurovisionworld.com/national/sweden/melodifestivalen-2023**]
- Vega, J. B. M. (2022, 24 enero). Tutorial: XGBoost en Python - Juan Bosco Mendoza Vega - Medium. *Medium*. [**https://medium.com/@jboscomendoza/tutorial-xgboost-en-python-53e48fc58f73**]
- Wikipedia contributors. (2023). Anaconda (Python distribution). *Wikipedia*. [**https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)**]
- Wikipedia contributors. (2023b). Spyder (software). *Wikipedia*. [**https://en.wikipedia.org/wiki/Spyder_(software)**]
- Zhang, Z. (2022, 21 mayo). Clustering Contextual Embeddings for Topic Modelling. *Medium*. [**https://towardsdatascience.com/clustering-contextual-embeddings-for-topic-model-1fb15c45b1bd**]

ANEXOS

Orden de los festivales

Benidorm Fest 2022

Orden	*Primera semifinal (26 de enero)*	*Segunda Semifinal (27 de enero)*	*Final (29 de enero)*
1	Varry Brava: "Raffaella"	Xeinn: "Eco"	Rayden: "Calle de la llorería "
2	Axúcar Moreno: "Postureo"	Marta Sango: "Sigues en mi mente"	Tanxugueiras: "Terra"
3	Blanca Paloma: "Secreto de agua"	Javiera Mena: "Culpa"	Varry Brava: "Raffaella"
4	Unique: "Mejores"	Gonzalo Hermida: "Quién lo diría"	Chanel: "SloMo"
5	Tanxugueiras: "Terra"	Rigoberta Bandini: "Ay mamá"	Rigoberta Bandini: "Ay mamá"
6	Chanel: "SloMo"	Rayden: "Calle de la llorería "	Xeinn: "Eco"
7		Sara Deop: "Make You Say"	Gonzalo Hermida: "Quién lo diría"
8			Blanca Paloma: "Secreto de agua"

Benidorm Fest 2023

Orden	*Primera semifinal (31 de enero)*	*Segunda Semifinal (2 de febrero)*	*Final (4 de febrero)*
1	Sharonne: "Aire"	Famous: "La Lola"	Karmento: "Quiero y duelo"
2	Aritz Aren: "Flamenco"	José Otero: "Inviernos en Marte"	Megara: "Arcadia"
3	Sofía Martín: "Tuki"	Karmento: "Quiero y duelo"	Alice Wonder: "Yo quisiera"
4	Agoney: "Quiero arder"	Rakky Ripper: "Tracción"	Fusa Nocta: "Mi familia"
5	Megara: "Arcadia"	Blanca Paloma: "Eaea"	Agoney: "Quiero arder"
6	Alice Wonder: "Yo quisiera"	E'Femme: "UFF!"	Blanca Paloma: "EAEA"
7	Meler: "No nos moverán"	Siderland: "Que esclati tot"	José Otero: "Inviernos en Marte"
8	Fusa Nocta: "Mi familia"	Alfred García: "Desde que tú estás"	Vicco: "Nochentera"
9	Twin Melody: "Sayonara"	Vicco: "Nochentera"	

Melodifestivalen 2022

Orden	*Primera semifinal* *(4 de febrero)*	*Segunda semifinal* *(11 de febrero)*	*Tercera semifinal* *(18 de febrero)*	*Cuarta semifinal* *(25 de febrero)*	*Quinta semifinal - Andra Chansen* *(4 de marzo)*
1	Malou Prytz: "Bananas"	Laimoo: "Bluffin"	Cazzi Opeia: "I Can't Get Enough"	Anna Bergendahl: "Higher Power"	Tone Sekelius: "My Way"
2	Theoz: "Son Du Vill"	Niello & Lisa Ajax: "Tror Du Att Jag Bryr Mig"	Lancelot: "Lyckligt Slut"	Lillasyster: "Til Our Days Are Over"	Alvaro Estrella: "Suave"
3	Shirley Clamp: "Let There Be Angels"	Samira Manners: "I Want To Be Loved"	Lisa Miskovsky: "Best To Come"	Malin Christin: "Synd On Dig"	Danne Stråhed: "Hallabaloo"
4	Omar Rudberg: "Moving Like That"	Alvaro Estrella: "Suave"	Tribe Friday: "Shut Me Up"	Tenori: "La Stella"	Anna Bergendahl: "Higher Power"
5	Danne Stråhed: "Hallabaloo"	Browsing Collection: "Face In The Crowd"	Faith Kakembo: "Freedom"	Medina: "In I Dimman"	Theoz: "Son Du Vill"
6	Cornelia Jakobs: "Hold Me Closer"	John Lundvik: "Änglavakt"	Linda Bengtzing: "Fyrfaldigt Hurra!"	Angelino: "The End"	Lisa Miskovsky: "Best To Come"
7	Robin Bengtsson: "Inocent Love"	Tone Sekelius: "My Way"	Anders Bagge: "Bigger Than The Universe"	Klara Hammarström: "Run To The Hills"	Lillasyster: "Til Our Days Are Over"
8					Cazzi Opeia: "I Can't Get Enough"

Orden	*Final (11 de marzo):*
1	Klara Hammarström: "Run To The Hills"
2	Theoz: "Som du vill"
3	Anna Bergendahl: "Higher Power"
4	John Lundvik: "Änglavakt"
5	Tone Sekelius: "My Way"
6	Anders Bagge: "Bigger Than The Universe"
7	Robin Bengtsson: "Innocent Love"
8	Faith Kakembo: "Freedom"
9	Liamoo: "Bluffin"

10	Cornelia Jakobs: “Hold Me Closer”
11	Cazzi Opeia: “I Can’t Get Enough”
12	MEDINA: “In i dimman”

Melodifestivalen 2023

Orden	*Primera semifinal* *(4 de febrero)*	*Segunda semifinal* *(11 de febrero)*	*Tercera semifinal* *(18 de febrero)*	*Cuarta semifinal* *(25 de febrero)*	*Quinta semifinal - Andra Chansen* *(4 de marzo)*
1	Tone Selkelius: “Rhythm Of My Show”	Wiktoria: “All My Life (Where Have You Been)”	Paul Rey: “Royals”	Kiana: “Where Did You Go”	Theoz: “Mer Av Dig”
2	Loulou LaMotte: “Inga Sorger”	Eden: “Comfortable”	Casanovas: “Så Kommer Kanslorna Tillbaka”	Signe & Hjördis: “Edelweiss”	Mariette: “One Day”
3	Rejhan: “Haunted”	Uje Brandelius: “Grytan”	Melanie Wehbe: “For The Show”	Smash Into Pieces: “Six Feet Under”	Victor Crone: “Diamonds”
4	Elov & Beny: “Raggen går”	Panetoz: “On My Way”	Nordman: “Släpp Alla Sorger”	Mariette: “One Day”	Tennessee tears: “Now I Know”
5	Victor Crone: “Diamonds”	Tennessee tears: “Now I Know”	Laurell: “Sober”	Emil Henrohn: “Mera Mera Mera”	Elov & Beny: “Raggen går”
6	Eva Rydberg & Ewa Roos: “Länge Leve Livet”	Maria sur: “Never Give Up”	Ida-Lova: “Låt Hela Stan Se På”	Axel Schylström: “Gorgeous”	Melanie Wehbe: “For The Show”
7	Fjällgren, Arc north ft. Adam Woods: “Where Are You (Sávečan)"	Theoz: “Mer Av Dig”	Marcus & Martinus: “Air”	Loreen: “Tattoo”	Nordman: “Släpp Alla Sorger”
8					Kiana: “Where Did You Go”

Orden	*Final (11 de marzo):*
1	Jon Henrik Fjällgren, Arc north ft. Adam Woods: “Where Are You (Sávečan)
2	Tone Selkelius: “Rhythm Of My Show”
3	Mariette: “One Day”
4	Marcus & Martinus: “Air”
5	Panetoz: “On My Way”
6	Maria sur: “Never Give Up”
7	Smash Into Pieces: “Six Feet Under”
8	Kiana: “Where Did You Go”
9	Nordman: “Släpp Alla Sorger”
10	Loreen: “Tattoo”

11	Theoz: “Mer Av Dig”
12	Paul Rey: “Royals”

Eurovisión 2021

Orden	*Primera Semifinal (18 de mayo)*
1	Lituania con The Roop: "Discoteque"
2	Eslovenia con Ana Soklič: "Amen"
3	Rusia con Manizha: "Russian Woman"
4	Suecia con Tusse: "Voices"
5	Australia con Montaigne: "Technicolour"
6	Macedonia del Norte con Vasil: "Here I Stand"
7	Irlanda con Lesley Roy: "Maps"
8	Chipre con Elena Tsagrinou: "El diablo"
9	Noruega con Tix; "Fallen Angel"
10	Croacia con Albina: "Tick-Tock"
11	Bélgica con Hooverphonic: "The Wrong Place"
12	Israel con Eden Alene: "Set Me Free"
13	Rumanía con Roxen: "Amnesia"
14	Azerbaiyán con Efendi: "Mata Hari"
15	Ucrania con Go_A: "Shum"
16	Malta con Destiny: "Je me casse"

Orden	*Segunda Semifinal (20 de mayo)*
1	San Marino con Senhit: "Adrenalina"
2	Estonia con Uku Suviste: "The Lucky One"
3	República Checa con Benny Cristo: "Omaga"
4	Grecia con Stefania: "Last Dance"
5	Austria con Vincent Bueno: "Amen"
6	Polonia con RAFAŁ: "The Ride"
7	Moldavia con Natalia Gordienko: "Sugar"
8	Islandia con Daði og Gagnamagnið: "10 Years"
9	Serbia con Hurricane: "Loco Loco"
10	Georgia con Tornike Kipiani: "You"
11	Albania con Anxhela Peristeri: "Karma"
12	Portugal con The Black Mamba: "Love Is On My Side"
13	Bulgaria con Victoria: "Growing Up Is Getting Old"
14	Finlandia con Blind Channel: "Dark Side"
15	Letonia con Samanta Tina: "The Moon Is Rising"
16	Suiza con Gjon's Tears: "Tout l'Univers"
17	Dinamarca con Fyr & Flamme: "Øve Os På Hinanden"

Final (22 de mayo)

Orden		**Orden**	
1	Chipre con Elena Tsagrinou: "El diablo"	14	Moldavia con Natalia Gordienko: "Sugar"
2	Albania con Anxhela Peristeri: "Karma"	15	Alemania con Jendrik: "I Don't Feel Hate"
3	Israel con Eden Alene: "Set Me Free"	16	Finlandia con Blind Channel: "Dark Side"
4	Bélgica con Hooverphonic: "The Wrong Place"	17	Bulgaria con Victoria: "Growing Up Is Getting Old"
5	Rusia con Manizha: "Russian Woman"	18	Lituania con The Roop: "Discoteque"
6	Malta con Destiny: "Je me casse"	19	Ucrania con Go_A: "Shum"
7	Portugal con The Black Mamba: "Love Is On My Side"	20	Francia con Barbara Pravi: "Voilà"
8	Serbia con Hurricane: "Loco Loco"	21	Azerbaiyán con Efendi: "Mata Hari"
9	Reino Unido con James Newman: "Embers"	22	Noruega con Tix; "Fallen Angel"
10	Grecia con Stefania: "Last Dance"	23	Países Bajos con Jeangu Macrooy: "Birth Of A New Age"
11	Suiza con Gjon's Tears: "Tout l'Univers"	24	Italia con Måneskin: "Zitti E Buoni"
12	Islandia con Daði og Gagnamagnið: "10 Years"	25	Suecia con Tusse: "Voices"
13	España con Blas Cantó: "Voy A Quedarme"	26	San Marino con Senhit: "Adrenalina"

Eurovisión 2022

Orden	*Primera Semifinal (10 de mayo)*
1	Albania con Ronela Hajati: "Sekret"
2	Letonia con Citi Zēni: "Eat Your Salad"
3	Lituania con Monika Liu: "Sentimentai"
4	Suiza con Marius Bear: "Boys Do Cry"
5	Eslovenia con LPS: "Disko"
6	Ucrania con Kalush Orchestra: "Stefania"
7	Bulgaria con Intelligent Music Project: "Intention"
8	Países Bajos con S10: "De Diepte"
9	Moldavia con Zdob şi Zdub & Fraţii Advahov: "Trenuleţul"
10	Portugal con MARO: "Saudade Saudade"
11	Croacia con Mia Dimšić: "Guilty Pleasure"
12	Dinamarca con REDDI: "The Show"
13	Austria con LUM!X feat. Pia Maria: "Halo"
14	Islandia con Systur: "Með Hækkandi Sól"
15	Grecia con Amanda Georgiadi Tenfjord: "Die Together"

16	Noruega con Subwoolfer: "Give That Wolf A Banana"
17	Armenia con Rosa Linn: "Snap"

Orden	*Segunda Semifinal (12 de mayo)*
1	Finlandia con The Rasmus: “Jezebel”
2	Israel con Michael Ben David: “I.M”
3	Serbia con Konstrakta: “In Corpore Sano”
4	Azerbaiyán con Nadir Rustamli: “Fade To Black”
5	Georgia con Circus Mircus: “Lock Me In”
6	Malta con Emma Muscat: “I Am What I Am”
7	San Marino con Achille Lauro: “Stripper”
8	Australia con Sheldon Riley: “Not The Same”
9	Chipre con Andromache: “Ela”
10	Irlanda con Brooke: “That’s Rich”
11	Macedonia del Norte con Andrea: “Circles”
12	Estonia con Stefan: “Hope”
13	Rumanía con WRS: “Llámame”
14	Polonia con Ochman: “River”
15	Montenegro con Vladana: “Breathe”
16	Bélgica con Jérémie Makiese: “Miss You”
17	Suecia con Cornelia Jakobs: “Hold Me Closer”
18	República Checa con We Are Domi: “Lights Off”

Final (13 de mayo)

Orden		**Orden**	
1	República Checa con We Are Domi: “Lights Off”	14	Lituania con Monika Liu: “Sentimentai”
2	Rumanía con WRS: “Llámame”	15	Azerbaiyán con Nadir Rustamli: “Fade To Black”
3	Portugal con MARO: “Saudade, Saudade”	16	Bélgica con Jérémie Makiese: “Miss You”
4	Finlandia con The Rasmus: “Jezebel”	17	Grecia con Amanda Georgiadi Tenfjord: “Die Together”
5	Suiza con Marius Bear: “Boys Do Cry”	18	Islandia con Systur: “Með Hækkandi Sól”
6	Francia con Alvan & Ahez: “Fulenn”	19	Moldavia con Zdob şi Zdub & Advahov Brothers: “Trenulețul”
7	Noruega con Subwoolfer: “Give That Wolf A Banana”	20	Suecia con Cornelia Jakobs: “Hold Me Closer”
8	Armenia con Rosa Linn: “Snap”	21	Australia con Sheldon Riley: “Not The Same”
9	Italia con Mahmood & Blanco: “Brividi”	22	Reino Unido con Sam Ryder: “Space Man”
10	España con Chanel: “SloMo”	23	Polonia con Ochman: “River”
11	Países Bajos con S10: “De Diepte”	24	Serbia con Konstrakta: “In Corpore Sano”

12	Ucrania con Kalush Orchestra: “Stefania”	25	Estonia con Stefan: “Hope”
13	Alemania con Malik Harris: “Rockstars”		

Eurovisión 2023

Orden	*Primera Semifinal (9 de mayo)*
1	Noruega con Alessandra: “Queen of Kings”
2	Malta con The Busker: “Dance (Our Own Party)”
3	Serbia con Luke Black: “Samo mi se spava”
4	Letonia con Sudden Lights: “Aijā”
5	Portugal con Mimicat: “Ai Coração”
6	Irlanda con Wild Youth: “We Are One”
7	Croacia con Let 3: “Mama *ŠČ*!”
8	Suiza con Remo Forrer: “Watergun”
9	Israel con Noa Kirel: “Unicorn”
10	Moldavia con Pasha Parfeny: “Soarele și Luna”
11	Suecia con Loreen: “Tattoo”
12	Azerbaiyán con TuralTuranX: “Tell Me More”
13	Chequia con Vesna: “My Sister’s Crown”
14	Países Bajos con Mia Nicoali y Dion Cooper: “Burning Daylight”
15	Finlandia con Käärijä: “Cha Cha Cha”

Orden	*Segunda Semifinal (11 de mayo)*
1	Dinamarca con Reiley: “Breaking My Heart”
2	Armenia con Brunette: “Future Lover”
3	Rumanía con Theodor Andrei: “D.G.T. (Off and On)”
4	Estonia con Alika: “Bridges”
5	Bélgica con Gustaph: “Because of You”
6	Chipre con Andrew Lambrou: “Break A Broken Heart”
7	Islandia con Diljá: “Power”
8	Grecia con Victor Vernicos: “What They Say”
9	Polonia con Blanka: “Solo”
10	Eslovenia con Joker Out: “Carpe Diem”
11	Georgia con Iru: “Echo”
12	San Marino con Piqued Jacks: “Like An Animal”
13	Austria con Teya & Salena: “Who The Hell Is Edgar?”
14	Albania con Albina & Familja Kelmendi: “Duje”
15	Lituania con Monika: “Stay”
16	Australia con Voyager: “Promise”

Final (13 de mayo)

Orden		**Orden**	
1	Austria con Teya & Salena: “Who The Hell Is Edgar?”	14	Chequia con Vesna: “My Sister’s Crown”
2	Portugal con Mimicat: “Ai Coração”	15	Australia con Voyager: “Promise”
3	Suiza con Remo Forrer: “Watergun”	16	Bélgica con Gustaph: “Because of You”
4	Polonia con Blanka: “Solo”	17	Armenia con Brunette: “Future Lover”
5	Serbia con Luke Black: “Samo mi se spava”	18	Moldavia con Pasha Parfeny: “Soarele și Luna”
6	Francia con La Zarra: “Évidemment”	19	Ucrania con Tvorchi: “Heart of Steel”
7	Chipre con Andrew Lambrou: “Break A Broken Heart”	20	Noruega con Alessandra: “Queen of Kings”
8	España con Blanca Paloma: “Eaea”	21	Alemania con Lord Of The Lost: “Blood & Glitter”
9	Suecia con Loreen: “Tattoo”	22	Lituania con Monika: “Stay”
10	Albania con Albina & Familja Kelmendi: “Duje”	23	Israel con Noa Kirel: “Unicorn”
11	Italia con Marco Mengoni: “Due vite”	24	Eslovenia con Joker Out: Carpe Diem
12	Estonia con Alika: “Bridges”	25	Croacia con Let 3: “Mama ŠČ!”
13	Finlandia con Käärijä: “Cha Cha Cha”	26	Reino Unido con Mae Muller: “I Wrote A Song”

Primera prueba del indicador de amistad entre países

La explicación que tenemos a continuación se basa en la primera prueba que se hizo para crear el indicador de amistad entre países de Eurovisión, la cual cambiamos por la que ya se ha leído en el cuerpo del proyecto ya que era más representativa.

“Este indicador tendrá un rango de -1 a 1, siendo -1 el peor valor y 1 el mejor. Poniendo un ejemplo sencillo inventado para un solo año, si España diese a Portugal 12 puntos (la puntuación máxima) y también viceversa, el indicador equivaldría a 1 ya que define total amistad entre países.

Para obtenerlo hemos recurrido a la página web “Eurovision_Song_Contest_YEAR” de Wikipedia siendo YEAR el valor del año a investigar. Hemos tenido que pasar la información manualmente a un documento Excel ya que a través de código no hemos logrado hacer el scrapping anualmente de la tabla de puntuaciones.

Cabe destacar que ha habido muchos cambios en el sistema de votaciones durante el paso de los años. El más relevante es que, a partir del 1997, se introdujo el televoto, permitiendo así mayor participación al público y una mayor audiencia.

Por esta razón, las puntuaciones han ido incrementado durante el paso del tiempo y no sería correcto mezclar resultados con sistemas de puntuación diferentes Por lo que los dividiremos en distintas etapas de años, creando una puntuación para cada indicador según la etapa en la que nos encontramos"

- *Etapa 1 (1957 – 1970 & 1974) i Etapa 2 (1971-1973)*

En la primera etapa, cada país participante tenía 10 miembros del jurado, y cada miembro del jurado podía otorgar un punto a una canción. En la segunda era parecido, pero cada país tenía dos miembros del jurado, y cada miembro del jurado otorgó de 1 a 5 puntos por cada canción.

El indicador para esta etapa es:

Puntos	Indicador de amistad
10	1
5	0
0	-1

- *Etapa 3 (1975-1997), Etapa 4 (1998-2008) i Etapa 5 (2009-2015)*

En la etapa 3 el jurado de cada país otorgó 12, 10, 8-1 puntos a sus 10 canciones favoritas. En la etapa 4 pasaba lo mismo pero el peso de los puntos recaía totalmente en el televoto. A partir de la proporción de votos de la audiencia para cada canción respecto el total de votos, se otorgaba un 12, 10, 8-1 puntos a las 10 canciones favoritas de cada país. Finalmente, en la etapa 5 se hacía una combinación 50-50% de los dos sistemas anteriores, donde se otorgaba la media de las mismas puntuaciones.

El indicador para esta etapa:

Puntos	Indicador de amistad
12	1
6	0
0	-1

- *Etapa 6 (2016-2022)*

50% televoto y 50% jurados nacionales: Los jurados de un país elegían sus 10 canciones favoritas otorgando un 12, 10, 8-1 puntos y el público de ese país lo mismo, siendo la máxima puntuación un 24.

Por esta razón, las tablas que se sacan de la página web son iguales, una para el jurado y la otra por el televoto, y serán parecidas a esta:

El indicador para esta etapa:

Puntos	Indicador de amistad
24	1
12	0
0	-1

Para cada relación de dos países se sacaría la proporción de cada tabla correspondiente.

Código Python usado para Eurovisión 2023

Scrapping cuotas casas de apuestas

```
# Cargamos las librerias necesarias
import requests
from bs4 import BeautifulSoup
import pandas as pd
import datetime

PATH = 'C:/Users/polgs/Desktop/TFG/Datos Eurovisión/'

# Declaramos la url que queremos extraer la información
url = 'https://eurovisionworld.com/odds/eurovision'

# Parámetros necesarios para hacer el scrapping de los datos
header = {
  "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36",
  "X-Requested-With": "XMLHttpRequest"
}

# Creamos la url necesaria
r = requests.get(url, headers=header)

# SCRAPPING
# Descargamos la informción si el codigo del estado es 200
status_code = r.status_code
if status_code == 200:

   # We pass the HTML content of the web to a BeautifulSoup() object
   html = BeautifulSoup(r.text, "html.parser")
   dfs = pd.read_html(r.text)

   # We get all the divs where the inputs are
   entradas = html.find_all('div', {'class': 'poll_inner'})

else:
```

```
   print("Status Code %d" % status_code)

# Nos quedamos con la tabla necesaria
df = dfs[2]
df = df.loc[:38, ]

# Eliminamos la primera columna sobrante
df.drop(['Unnamed: 0', 'Unnamed: 1'], axis=1, inplace=True)



# Renombramos las columnas
df.rename(columns={'Unnamed: 2': 'info'}, inplace=True)

# Separamos el artista de la cancion
df['cancion'] = [i.split(" - ")[1] if " - " in i else "" for i in df['info']]
df['pais_artista'] = [i.split(" - ")[0] if " - " in i else i for i in df['info']]
df[['pais', 'artista']] = df['pais_artista'].str.split(n=1, expand=True)

df['fecha'] = datetime.datetime.now()

# Creamos la lista de columnas que queremos visualizar en la tabla
cols_visualizar = ['pais', 'artista', 'cancion', 'winning chance', 'BET365', 'UNIBET',
'BETFAIR SPORT', 'COOL BET', 'SKY BET', 'LAD BROKES', 'BETSSON',
      'COMEON', 'BOYLE SPORTS', '888 SPORT', 'SMARKETS', '10BET', 'WILLIAM HILL',
'BET FRED', 'BETFAIR EXCHANGE', "fecha"]

# Nos aseguramos de que todas las columnas de la lista de visualización estén
presentes en el DataFrame
df = df.reindex(columns=cols_visualizar)

# Rellenamos los valores faltantes con "NA"
df = df.fillna("NA")

df = df.drop([26,27])

df.to_csv(PATH + 'Eurovisión23Apuestas.csv', mode = "a")  # mode = "a" append
```

Scrapping parámetros API Spotify

```
# Importamos las librerías necesarias
import math
import spotipy
import pandas as pd
from spotipy.oauth2 import SpotifyClientCredentials
```

```
path = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de
datos/Eurovisión Definitivo/TopicosLyricsEurovision.csv'
datos = pd.read_csv(path)
columnas = ['Artist', 'Title']
datos = datos[columnas]

client_id = '670cdf8c18b0441e83b18524a16b2bb5'
client_secret = '245b5b8c1ece4355b265c04345f38722'

client_credentials_manager = SpotifyClientCredentials(client_id = client_id,
                                        client_secret = client_secret)
sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
# Calculamos los id's de las canciones

def get_spotify_uris(df, artista, cancion):
   saved_uris = []
   artist_names = df[artista].values
   track_names = df[cancion].values

   for i in range(0, len(artist_names)):
      artist = artist_names[i]
      track = track_names[i]

      try:
         math.isnan(artist)
         saved_uris.append('Na')
      except:
         q = 'artist:{} track: {}'.format(artist, track)
         results = sp.search(q = q, limit = 1, type = 'track')
         if results['tracks']['items'] != []:
            uri = results['tracks']['items'][0]['uri']
            saved_uris.append(uri)
         else :
            saved_uris.append('Na')

   return saved_uris


uris = get_spotify_uris(datos, 'Artist', 'Title')
uris[22] = "spotify:track:25l25dLwkUGmvWx8N9gE8g"

datos['uris'] = uris

def get_audio_features(saved_uris):
   artist = []
   track = []
```

```
popularity = []
markets = []

danceability = []
energy = []
key = []
loudness = []
mode = []
speechiness = []
acousticness = []
instrumentalness = []
liveness = []
valence = []
tempo = []
duration_ms = []

for uri in saved_uris:
    if uri != 'Na':
        x = sp.audio_features(uri)
        y = sp.track(uri)
        for audio_features in x:
            danceability.append(audio_features['danceability'])
            energy.append(audio_features['energy'])
            key.append(audio_features['key'])
            loudness.append(audio_features['loudness'])
            mode.append(audio_features['mode'])
            speechiness.append(audio_features['speechiness'])
            acousticness.append(audio_features['acousticness'])
            instrumentalness.append(audio_features['instrumentalness'])
            liveness.append(audio_features['liveness'])
            valence.append(audio_features['valence'])
            tempo.append(audio_features['tempo'])
            duration_ms.append(audio_features['duration_ms'])

        artist.append(y['album']['artists'][0]['name'])
        track.append(y['name'])
        popularity.append(y['popularity'])
        markets.append(y['available_markets'])
    else:
        danceability.append('na')
        energy.append('na')
        key.append('na')
        loudness.append('na')
        mode.append('na')
        speechiness.append('na')
        acousticness.append('na')
```

```
            instrumentalness.append('na')
            liveness.append('na')
            valence.append('na')
            tempo.append('na')
            duration_ms.append('na')
            artist.append('na')
            track.append('na')
            popularity.append('na')
            markets.append('na')

        df = pd.DataFrame()
        df['artist'] = artist
        df['track'] = track
        df['popularity'] = popularity
        df['markets'] = markets
        df['danceability'] = danceability
        df['energy'] = energy
        df['key'] = key
        df['loudness'] = loudness
        df['mode'] = mode
        df['speechiness'] = speechiness
        df['acousticness'] = acousticness
        df['instrumentalness'] = instrumentalness
        df['liveness'] = liveness
        df['valence'] = valence
        df['tempo'] = tempo
        df['duration_ms'] = duration_ms
        df['markets'] = markets

        return df

data_features = get_audio_features(datos['uris'])
final = pd.concat([datos, data_features], axis = 1)

## Creamos las variables dummies
dummies = pd.get_dummies(final.markets.apply(pd.Series).stack()).sum(level=0)

## Añadimos las variables en la bbdd
datos_preprocessing = pd.concat([final, dummies], axis=1)

datos_preprocessing.to_csv("C:/Users/polgs/Desktop/TFG/Datos
Eurovisión/ParamSpotifyEuro.csv")
```

Extracción de las letras de las canciones en la API Genius

```
import spotipy
```

```
from spotipy.oauth2 import SpotifyClientCredentials
import lyricsgenius
import pandas as pd
import re

# Autenticación de Spotify API
client_credentials_manager =
SpotifyClientCredentials(client_id='670cdf8c18b0441e83b18524a16b2bb5',
client_secret='245b5b8c1ece4355b265c04345f38722')
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

# Obtención de los IDs de las canciones de la playlist
playlist_uri = 'spotify:playlist:37i9dQZF1DWVCKO3xAlT1Q'
results = sp.playlist_items(playlist_uri)
song_ids = []
for item in results['items']:
    track = item['track']
    song_ids.append(track['id'])

# Autenticación de Genius API
genius =
lyricsgenius.Genius('sjKE6HMrdQrPKzrnt714BKQQg8a16nT6rdEYyX7YzW0ipu4r3tJS2DNyJP
WcrpAZ')

# Obtención de las letras de las canciones
song_titles = []
artists = []
lyrics = []
for song_id in song_ids:
    track_info = sp.track(song_id)
    song_title = track_info['name']
    artist_name = track_info['album']['artists'][0]['name']
    song = genius.search_song(song_title, artist_name)
    if song is not None:
        song_lyrics = re.sub(r"^.*Lyrics", "", song.lyrics).replace("Lyrics", "")
        # Eliminar corchetes y lo que está dentro de ellos
        lyrics_cleaned = re.sub(r'\[.*\]', '', song.lyrics)
        # Eliminar los caracteres especiales y convertir a minúsculas
        lyrics_cleaned = re.sub(r'[^\w\s]', '', lyrics_cleaned).lower()
        # Eliminar los números
        lyrics_cleaned = re.sub(r'\d+', '', lyrics_cleaned)
        # Agregar la letra limpiada a la lista de letras
        lyrics.append(lyrics_cleaned)
        song_titles.append(song_title)
        artists.append(artist_name)
    else:
```

```
    # Si la canción no se encuentra en Genius, imprimir un mensaje y no agregarla a los arrays
    print(f"No se encontró la letra de la canción: {song_title} - {artist_name}")


# La letra de la canción Arcadia no se extrae correctamente, la añadimos manualmente
echo_lyrics = """
Days in a row I'm thinking, I know
I've got a big faith, my love is my crown
Will be better way, will be better day now
It is not a secret

Life is love
Thing is known
Like in dreams
Going through the life together
Going through the life together
Like in dreams

Days in a row I'm thinking, I know
I've got a big faith, my love is my crown
Will be better way, will be better day now
It is not a secret

Days in a row I'm thinking, I know
I've got a big faith, my love is my crown
Will be better way, will be better day now
It is not a secret

Going through the life together
Going through the life together

Days in a row I'm thinking, I know
I've got a big faith, my love is my crown
Will be better way, will be better day now
It is not a secret
Days in a row I'm thinking, I know
My love is not a fake, this feeling is, lord
Will be better way, will be better day now
It is not a secret

My soul's like a fortress, I feel I progressed
Words getting worthless, love is a wordless
Oh, when life is loved, loved

Days in a row I'm thinking, I know
```

```
I've got a big faith, my love is my crown
Will be better way, will be better day now
It is not a secret
"""

aicoracao_lyrics = """
Ai, coração
Que não me deixas em paz
Não me dás sossego, não me deixas capaz
Tenho a cabeça e a garganta num nó
Que não se desfaz e nem assim tu tens dó
Sinto-me tonta, cada dia pior
Já não sei de coisas que sabia de cor
As pulsações subiram quase pra mil
Estou louca, completamente senil
O peito a arder, a boca seca, eu sei lá
O que te fazer, amor, pra mim assim não dá
Porque parece que nem sou mais eu
Ai, coração
Ai, coração
Diz-me lá se és meu (ihu) (hey)
As horas passam e o sono não vem
Ouço as corujas e os vizinhos também
O meu juízo foi-se e por lá ficou
Alguém me tire deste estado em que estou
O doutor diz que não há nada a fazer
'Caso perdido', vi-o eu a escrever
Ando perdida numa outra dimensão
Toda eu sou uma grande confusão
O peito a arder, a boca seca, eu sei lá
O que te fazer, amor, pra mim assim não dá
Porque parece que nem sou mais eu
Ai, coração
Ai, coração
Ai, coração
Diz-me lá se és meu (aahm-ahm, ahm-ahm-ahm, ah-ah)
O peito a arder, a boca seca, eu sei lá
O que te fazer, amor, pra mim assim não dá
Porque parece que nem sou mais eu
Ai, coração
Ai, coração
Ai, coração
Ai, coração
Diz-me lá se és meu (hey)
"""
```

```
# Limpiamos las dos letras que hemos introducido manualmente y las insertamos en su
sitio correspondiente
echo_lyrics = re.sub(r'\[.*\]', '', echo_lyrics)
echo_lyrics = re.sub(r'[^\w\s]', '', echo_lyrics).lower()
echo_lyrics = re.sub(r'\d+', '', echo_lyrics)
lyrics.insert(27, echo_lyrics)

aicoracao_lyrics = re.sub(r'\[.*\]', '', aicoracao_lyrics)
aicoracao_lyrics = re.sub(r'[^\w\s]', '', aicoracao_lyrics).lower()
aicoracao_lyrics = re.sub(r'\d+', '', aicoracao_lyrics)
posicion = 22

if len(lyrics) > posicion:
    lyrics[posicion] = aicoracao_lyrics
else:
    print("La posición especificada está fuera del rango de la lista")

# Eliminar la primera línea que especifica la canción + lyrics
for i in range(len(lyrics)):
    lines = lyrics[i].split('\n')
    lyrics[i] = '\n'.join(lines[1:])

# Creación del DataFrame de canciones y letras
song_titles = [sp.track(song_id)['name'] for song_id in song_ids]
artists = [sp.track(song_id)['album']['artists'][0]['name'] for song_id in song_ids]
topic = pd.DataFrame({'Title': song_titles, 'Artist': artists, 'Lyrics': lyrics})

from googletrans import Translator
import pandas as pd
import time

# Función para traducir texto a español
def translate_to_english(text):
    # Inicializar traductor
    translator = Translator(service_urls=['translate.google.com'])
    translation = translator.translate(text, dest='en')
    return translation.text

# Aplicar la traducción a la columna 'Lyrics'
for index, row in topic.iterrows():
    try:
        row['Lyrics'] = translate_to_english(row['Lyrics'])
    except TypeError:
        print(f"Error en la fila {index}: {row['Lyrics']}")
```

```
# Guardar el DataFrame actualizado en un nuevo archivo CSV
topic.to_csv("C:/Users/polgs/Desktop/TFG/Datos Eurovisión/LyricsEurovision.csv",
index=False)
```

Topic Modelling y creamos la variable Tópico

```
import re
import spacy
import pandas as pd
from gensim.corpora import Dictionary
from gensim.models import LdaModel
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from wordcloud import WordCloud

stop_words_en = set(stopwords.words('english'))
stop_words_es = set(stopwords.words('spanish'))
stop_words_ro = set(stopwords.words('romanian'))
stop_words_fr = set(stopwords.words('french'))
stop_words_fi = set(stopwords.words('finnish'))
stop_words_po = set(stopwords.words('portuguese'))
stop_words_sl = set(stopwords.words('slovene'))
stop_words_it = set(stopwords.words('italian'))

nlp = spacy.load('en_core_web_md')

topicmodel = pd.read_csv('C:/Users/polgs/Desktop/TFG/Datos
Eurovisión/LyricsEurovision.csv')

def limpiar_texto(texto):
   # Eliminamos los caracteres especiales
   texto = re.sub(r'[\W\d]', ' ', str(texto))
   # Eliminado las palabras que tengo un solo caracter
   texto = re.sub(r'\s+[a-zA-Z]\s+', ' ', texto)
   # Sustituir los espacios en blanco en uno solo
   texto = re.sub(r'\s+', ' ', texto, flags=re.I)
   # Convertimos textos a minusculas
   texto = texto.lower()
   return texto

topicmodel["Lyrics"] = topicmodel.Lyrics.apply(limpiar_texto)
topicmodel.head()

otras_palabras = ["ohohoh", "zero", "ya", "poe", "cha", "til", "ill", "ooh", "one",
"two", "three", "pa", "uff", "la", "arcadia", "imma", "miri", "hey", "it", "the", "ea",
"ole", "dos", "tres", "ay", "si", "eh", "ah", "na", "lorolólolo", "loroló", "mmm",
```

```
"efemme", "eheh", "oh", "yeah", "upf", "lola", "ahah", "oooh", "ey", "ta", "chuli",
"sofia", "tuki", "sayonara"]

def remove_stopwords(tokens):
   filtered_tokens = [token for token in tokens if token.lower() not in stop_words_en
and stop_words_es and stop_words_ro and stop_words_fr and stop_words_fi and
stop_words_sl and stop_words_it and stop_words_po and token not in
otras_palabras]
   return filtered_tokens

def lemmatize(text):
   doc = nlp(text)
   lemmas = [token.lemma_ for token in doc]
   filtered_lemmas = remove_stopwords(lemmas)
   return filtered_lemmas

topicmodel["Lemas"] = topicmodel["Lyrics"].apply(lemmatize)
tokens = topicmodel.Lemas.tolist()

# Crear el diccionario
diccionario = Dictionary(tokens)
print(f'Número de tokens: {len(diccionario)}')

diccionario.filter_extremes(no_below=2, no_above = 0.8)
print(f'Número de tokens: {len(diccionario)}')

# Creamos el corpus
corpus = [diccionario.doc2bow(lyrics) for lyrics in tokens]

# Mostramos el BOW de una canción
print(corpus[6])

# Construimos el modelo LDA
lda = LdaModel(corpus=corpus, id2word=diccionario,
          num_topics=7, random_state=42,
          chunksize=50, passes=10, alpha='auto')

# Visualizamos los tópicos
topicos = lda.print_topics(num_words=10, num_topics=7)
for topico in topicos:
   print(topico)

# Visualizamos las nubes de palabras por tópico
for i in range(0, 7):
   plt.figure()
   plt.imshow(WordCloud(background_color='white', prefer_horizontal=1.0)
```

```
            .fit_words(dict(lda.show_topic(i, 20))))
    plt.axis("off")
    plt.title("Tópico " + str(i))
    plt.show()

# Para cada canción asignamos un tópico y calculamos las distancias entre canciones
topicmodel['Tópico'] = pd.Series()

for i in range(len(topicmodel)):
    bow_cancion = corpus[i]
    distribucion_cancion = lda[bow_cancion]

    # Indices de los topicos mas significativos
    dist_indices = [topico[0] for topico in lda[bow_cancion]]

    # Contribución de los topicos mas significativos
    dist_contrib = [topico[1] for topico in lda[bow_cancion]]

    distribucion_topicos = pd.DataFrame({'Topico':dist_indices,
                                'Contribucion':dist_contrib })
    distribucion_topicos.sort_values('Contribucion',
                            ascending=False, inplace=True)

    # Asignar nombre de tópico correspondiente a cada canción
    if distribucion_topicos.iloc[0].Topico == 0:
        topicmodel["Tópico"][i] = "Diversidad"
    elif distribucion_topicos.iloc[0].Topico == 1:
        topicmodel["Tópico"][i] = "Creatividad"
    elif distribucion_topicos.iloc[0].Topico == 2:
        topicmodel["Tópico"][i] = "Experiencia"
    elif distribucion_topicos.iloc[0].Topico == 3:
        topicmodel["Tópico"][i] = "Persistencia"
    elif distribucion_topicos.iloc[0].Topico == 4:
        topicmodel["Tópico"][i] = "Entretenimiento"
    elif distribucion_topicos.iloc[0].Topico == 5:
        topicmodel["Tópico"][i] = "Conocimiento"
    else:
        topicmodel["Tópico"][i] = "Emoción"

    # Imprime el tópico más significativo para cada canción
    print("Canción " + str(i))
    print("*** Tópico: " + str(int(distribucion_topicos.iloc[0].Topico)) + " ***")
    palabras = [palabra[0] for palabra in lda.show_topic(
        topicid=int(distribucion_topicos.iloc[0].Topico))]
    palabras = ', '.join(palabras)
    print(palabras, "\n")
```

```
#topicmodel.to_csv("C:/Users/polgs/Desktop/TFG/Datos
Eurovisión/TopicosLyricsEurovision.csv", index=False)

import seaborn as sns

sns.set(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.countplot(x="Tópico", data=topicmodel)
plt.show()
plt.savefig("C:/Users/polgs/Desktop/TFG/Datos Melodifestivalen/dist_topicos.jpg")

from collections import Counter

all_words = [word for sublist in tokens for word in sublist]
word_counts = Counter(all_words)

print("Palabras más frecuentes en el conjunto de datos:")
for word, count in word_counts.most_common(5):
   print(f"{word}: {count}")

playlists = {
   "Diversidad": [],
   "Creatividad": [],
   "Experiencia": [],
   "Persistencia": [],
   "Entretenimiento": [],
   "Conocimiento": [],
   "Emoción": []
}

for i, row in topicmodel.iterrows():
   playlists[row["Tópico"]].append((row["Title"], row["Artist"]))

for topic, songs in playlists.items():
   print("Playlist de {}: \n".format(topic))
   for song in songs:
      print("{} - {}".format(song[0], song[1]))
   print("\n")
```

Indicador de amistad de países

```
import pandas as pd

PATH = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/IndicadorAmistad.xlsx'
data_dict = pd.read_excel(PATH, sheet_name=None)
```

```
pair_data = []

# Iterar a través de todas las hojas del diccionario y para cada hoja, obtener la lista de países que han participado
for sheet_name, sheet_data in data_dict.items():
    # Agregar una nueva columna 'Año' con el valor del año de la hoja correspondiente
    sheet_data['Año'] = sheet_name

    # Renombrar la columna "Country" a "Pais_Donado"
    sheet_data.rename(columns={'Country': 'Pais_Donado'}, inplace=True)

    # Llenar cualquier valor NaN en la hoja de datos actual con 0
    sheet_data.fillna(value=0, inplace=True)

    # Derretir las columnas (excepto 'Pais_Donador' y 'Año')
    melted_data = pd.melt(sheet_data, id_vars=['Pais_Donado', 'Año'], var_name='Pais_Donante', value_name='Cantidad')

    # Eliminar las filas donde la columna "Pais_Donador" y la columna "País_Donado" tengan el mismo valor
    melted_data = melted_data.query('Pais_Donante != `Pais_Donado`')

    # Agregar los datos derretidos a la lista pair_data
    pair_data.append(melted_data)

# Concatenar todos los datos en un solo DataFrame
all_data = pd.concat(pair_data)

# Restablecer el índice del DataFrame resultante
all_data.reset_index(drop=True, inplace=True)

all_data = all_data.sort_values(["Pais_Donante", "Pais_Donado", "Año"])

all_data = all_data.reindex(columns=['Pais_Donante', 'Pais_Donado', 'Cantidad', 'Año'])

import pandas as pd
import io
import requests
import geopandas as gpd
from shapely import affinity
import mapclassify
import igraph as ig
import matplotlib.pyplot as plt
import igraph as ig
from bs4 import BeautifulSoup
```

```
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression

# leemos datos
path ='C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/Eurovisión
Definitivo/IndicadorPaisesDefinitivo.csv'
datos = pd.read_csv(path, encoding='utf-8')

# Lista para guardar los resultados de cada año
resultados = []

# Iteramos sobre los años únicos
for year in datos['Año'].unique():
    # Subconjunto de datos para el año actual
    subset = datos[datos['Año'] == year]

    # Edges / Nodos
    edges = pd.DataFrame(subset, columns=["Pais_Donante", "Pais_Donado", "Cantidad"])

    # Cremos el grafo de Louvain
    graph = ig.Graph.TupleList(edges.itertuples(index=False), directed=False)

    # Detectamos comunidades con el algoritmo de Louvain
    cluster = graph.community_multilevel(weights=edges['Cantidad'], resolution=1.1)

    # Obtenemos el índice de modularidad
    modularity = cluster.modularity

    # Asignamos cada nodo a su comunidad correspondiente
    cluster_df = pd.DataFrame({'label': graph.vs['name'],
                               'cluster': cluster.membership,
                               'modularity': modularity})

    # Añadimos el resultado a la lista
    resultados.append({'Año': year, 'cluster_df': cluster_df})

# Combinamos los resultados en un único DataFrame
final_df = pd.concat([r['cluster_df'].assign(Año=r['Año']) for r in resultados])

indice = final_df['modularity'].unique()

# Creamos un array de características (X) y un array de etiquetas (y)
X = np.arange(len(indice)).reshape(-1, 1)
y = np.array(indice)
```

```
# Creamos el modelo de regresión lineal
modelo = LinearRegression()

# Entrenamos el modelo
modelo.fit(X, y)

# Predecimos el siguiente valor de índice de modularidad para el año 2023
siguiente_valor = modelo.predict([[len(indice)]])

valor_mas_cercano = min(indice, key=lambda x: abs(x - siguiente_valor))

for year in final_df['Año'].unique():
   # Filtramos las filas correspondientes al año actual y Yugoslavia
   yugoslavia_rows = (final_df['Año'] == year) & (final_df['label'] == 'Yugoslavia')
   serbiamontenegro_rows = (final_df['Año'] == year) & (final_df['label'] == 'Serbia and
Montenegro')

   if any(yugoslavia_rows):
      yugoslavia_df = pd.DataFrame({
         'label': ['Bosnia Herz.', 'Croatia', 'Slovenia', 'North Macedonia', 'Montenegro',
'Serbia'],
         'cluster': final_df.loc[yugoslavia_rows, 'cluster'].values[0],
         'modularity': final_df.loc[yugoslavia_rows, 'modularity'].values[0],
         'Año': year
      })

      # Reemplazamos las filas correspondientes a Yugoslavia con el dataframe
yugoslavia_df
      final_df = final_df[~yugoslavia_rows].append(yugoslavia_df, ignore_index=True)

   if any(serbiamontenegro_rows):
      serbiamontenegro_df = pd.DataFrame({
         'label': ['Serbia', 'Montenegro'],
         'cluster': final_df.loc[serbiamontenegro_rows, 'cluster'].values[0],
         'modularity': final_df.loc[serbiamontenegro_rows, 'modularity'].values[0],
         'Año': year
      })
      # Reemplazamos las filas correspondientes a Yugoslavia con el dataframe
yugoslavia_df
      final_df = final_df[~serbiamontenegro_rows].append(serbiamontenegro_df,
ignore_index=True)

final_df = final_df.sort_values(by=['Año', 'label'], ascending=[True, True])

# Asignamos los valores al año 2023 teniendo un índice de modularidad de 0.1434
```

```
participantes_2023 = ['Albania', 'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Belgium',
'Croatia', 'Cyprus', 'Czech Republic', 'Denmark', 'Estonia', 'Finland', 'France', 'Georgia',
'Germany', 'Greece', 'Iceland', 'Ireland', 'Israel', 'Italy', 'Latvia', 'Lithuania', 'Malta',
'Moldova', 'Netherlands', 'Norway', 'Poland', 'Portugal', 'Romania', 'San Marino', 'Serbia',
'Slovenia', 'Spain', 'Sweden', 'Switzerland', 'Ukraine', 'United Kingdom']
datos_2002 = final_df.loc[final_df['Año'] == 2002]
datos_2023 = datos_2002[datos_2002['label'].isin(participantes_2023)]
participantes_no_coincidentes =
list(set(participantes_2023).difference(datos_2023['label']))

datos_1999 = final_df.loc[final_df['Año'] == 1999]
datos_2023_2 = datos_1999[datos_1999['label'].isin(participantes_no_coincidentes)]
participantes_no_coincidentes_2 =
list(set(participantes_no_coincidentes).difference(datos_2023_2['label']))

datos_2019 = final_df.loc[final_df['Año'] == 2019]
datos_2023_3 = datos_2019[datos_2019['label'].isin(participantes_no_coincidentes_2)]
participantes_no_coincidentes_3 =
list(set(participantes_no_coincidentes_2).difference(datos_2023_3['label']))

datos_2016 = final_df.loc[final_df['Año'] == 2016]
datos_2023_4 = datos_2016[datos_2016['label'].isin(participantes_no_coincidentes_3)]

datos_finales = pd.concat([datos_2023, datos_2023_2, datos_2023_3, datos_2023_4],
ignore_index=True)

# Cambiamos para el 2023
datos_finales['modularity'] = 0.143419
datos_finales['Año'] = 2023

from itertools import combinations

# Obtenemos las combinaciones únicas de países y contar cuántas veces coinciden en el
mismo cluster
combinaciones_paises = list(combinations(final_df['label'].unique(), 2))

# Creamos una lista para almacenar los resultados
recuento = []

# Iteramos sobre las combinaciones de pares de países
for combinacion in combinaciones_paises:
    pais1, pais2 = combinacion

    # Filtramos el DataFrame por las combinaciones de países
    coincidencias = final_df[(final_df['label'] == pais1) | (final_df['label'] == pais2)]
```

```
    # Creamos un diccionario para almacenar los resultados por año
    num_coincidencias = 0
    num_veces = 0

    # Iteramos sobre los años únicos
    for year in coincidencias['Año'].unique():
        # Filtramos las coincidencias por año
        coincidencias_año = coincidencias[coincidencias['Año'] == year]

        # Verificamos si ambos países están presentes en el año actual
        if pais1 in coincidencias_año['label'].values and pais2 in
coincidencias_año['label'].values:
            coincidencias_año = coincidencias_año.reset_index(drop=True)

            # Verificamos si el valor de cluster es el mismo para ambas coincidencias
            cluster_pais1 = coincidencias_año.loc[0, 'cluster']
            cluster_pais2 = coincidencias_año.loc[1, 'cluster']

            # Verificamos si el valor de cluster es el mismo
            if cluster_pais1 == cluster_pais2:
                num_coincidencias = num_coincidencias + 1

            num_veces = num_veces + 1

    # Agregamos el resultado a la lista
    recuento.append({'Pais1': pais1, 'Pais2': pais2, 'Coincidencias': num_coincidencias,
'Total': num_veces})

# Creamos un DataFrame con los resultados
resultados_df = pd.DataFrame(recuento)

# Imprimimos los resultados
print(resultados_df)

# Ordenamos el DataFrame por el número de coincidencias en orden descendente
resultados_df = resultados_df.sort_values('Coincidencias', ascending=False)

'''
# Leemos el mapa
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

participating_countries = [
    'Albania', 'Andorra', 'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Belarus', 'Belgium',
'Black Sea', 'Bosnia and Herz.',
    'Bulgaria', 'Croatia', 'Cyprus', 'Czechia', 'Denmark', 'Estonia', 'Finland', 'France', 'Georgia',
```

```
   'Germany', 'Greece', 'Hungary', 'Iceland', 'Ireland', 'Israel', 'Italy', 'Kosovo', 'Latvia',
'Liechtenstein', 'Lithuania', 'Luxembourg',
   'Malta', 'Moldova', 'Monaco', 'Montenegro', 'Morocco', 'Netherlands', 'North
Macedonia', 'Norway', 'Poland', 'Portugal',
   'Romania', 'Russia', 'San Marino', 'Serbia', 'Slovakia', 'Slovenia', 'Spain', 'Sweden',
'Switzerland', 'Turkey',
   'Ukraine', 'United Kingdom'
]

# Creamos un DataFrame a partir de la lista de países
df = pd.DataFrame(participating_countries, columns=['Country'])

merged_df = pd.merge(world, df, left_on=['name'], right_on=['Country'], how='inner')


import imageio
import os

# Carpeta para almacenar los frames de la animación
frames_folder = 'frames'

# Creamos la carpeta si no existe
if not os.path.exists(frames_folder):
   os.makedirs(frames_folder)


import os
import matplotlib.cm as cm


# Tamaño de la figura
fig_size = (10, 6)

# Determinamso el número de clústers únicos
num_clusters = len(final_df['cluster'].unique())

# Elige una paleta de colores con el número deseado de colores.

cmap = cm.get_cmap('tab10', num_clusters)


# Iteramos sobre los años únicos y creamos un plot para cada año
for year in final_df['Año'].unique():
   # Filtramos los datos por año
   subset = final_df[final_df['Año'] == year]
```

```
    # Creamos el dataframe vacío
    no_participan = pd.DataFrame(columns=['Pais', 'Año'])

    # Añadimos los países que no participan
    for pais in participating_countries:
        if pais not in subset['label'].values:
            new_row = pd.DataFrame({'Pais': [pais], 'Año': year})
            no_participan = pd.concat([no_participan, new_row], ignore_index=True)

    # Unimos los datos con el mapa europeo
    map_data = merged_df.merge(subset, left_on='name', right_on='label')
    map_data2 = merged_df.merge(no_participan, left_on='name', right_on='Pais')

    # Creamos un plot con geopandas
    fig, ax = plt.subplots(figsize=fig_size)

    # Obtenemos la lista de colores para los clústeres
    map_data.plot(column='cluster', cmap=cmap, linewidth=0.5, edgecolor='black', ax=ax)

    # Pintamos los países de map_data2 de color gris
    map_data2.plot(ax=ax, color='gray')

    # Configuramos el título y los límites del eje
    ax.set_title('Clusters de países en Europa en {}'.format(year))
    ax.set_xlim(-20, 40)
    ax.set_ylim(30, 75)

    # Ocultamos los ejes
    ax.axis('off')

    # Guardamos el plot como una imagen
    filename = os.path.join(frames_folder, 'frame_{}.png'.format(year))
    plt.savefig(filename)

    # Cerrarom la figura para liberar memoria
    plt.close(fig)

# Creamos una lista de los nombres de archivo de los frames
frame_files = [os.path.join(frames_folder, f) for f in os.listdir(frames_folder) if
f.endswith('.png')]

# Creamos una lista para almacenar los frames de la animación
frames = []

# Abrimos cada imagen y la añadimos a la lista de frames
for frame_file in frame_files:
```

```
    img = imageio.imread(frame_file)
    frames.append(img)

# Guardamos la animación en formato MP4
output_file = 'animacion.mp4'
imageio.mimsave(output_file, frames, fps=1.5)

# Eliminamos los archivos de imagen individuales
for frame_file in frame_files:
    os.remove(frame_file)

# Eliminamos la carpeta de los frames
os.rmdir(frames_folder)

print('Animación guardada en: {}'.format(output_file))


participating_countries = [
    'Albania', 'Andorra', 'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Belarus', 'Belgium',
'Bosnia and Herz.',
    'Bulgaria', 'Croatia', 'Cyprus', 'Czechia', 'Denmark', 'Estonia', 'Finland', 'France', 'Georgia',
    'Germany', 'Greece', 'Hungary', 'Iceland', 'Ireland', 'Israel', 'Italy', 'Latvia', 'Lithuania',
'Luxembourg',
    'Malta', 'Moldova', 'Monaco', 'Montenegro', 'Morocco', 'Netherlands', 'North
Macedonia', 'Norway', 'Poland', 'Portugal',
    'Romania', 'Russia', 'San Marino', 'Serbia', 'Slovakia', 'Slovenia', 'Spain', 'Sweden',
'Switzerland', 'Turkey',
    'Ukraine', 'United Kingdom'
]

grupos =
[1,3,4,5,5,4,4,5,1,4,1,0,2,2,4,2,3,4,5,0,2,2,5,3,3,4,4,5,0,4,3,1,3,5,1,2,4,3,3,4,3,1,2,1,3,2,2,1,
4,5]

mapa_final = pd.DataFrame({'País': participating_countries, 'Grupo': grupos})

mapa = pd.merge(world, mapa_final, left_on=['name'], right_on=['País'], how='inner')

num_clusters2 = len(mapa_final['Grupo'].unique())

cmap = cm.get_cmap('tab10', num_clusters2)

fig_size = (10, 6)

# Creamos un plot con geopandas
fig, ax = plt.subplots(figsize=fig_size)
```

```
# Obtenemos la lista de colores para los clústeres
mapa.plot(column='Grupo', cmap=cmap, linewidth=0.5, ax=ax)

# Configuraramos el título y los límites del eje
ax.set_title('Comunidades de países en Eurovisión')
ax.set_xlim(-20, 40)
ax.set_ylim(30, 75)

# Ocultamos los ejes
ax.axis('off')

plt.show()
'''

indicador = pd.concat([final_df, datos_finales], ignore_index=True)
indicador.to_csv("C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de
datos/Eurovisión Predicción/indicador.csv")
```

Agrupamiento de bases de datos, análisis descriptivo de variables y tratamiento de missings y outliers.

```
# Importamos las librerías necesarias
import pandas as pd
import numpy as np
from datetime import datetime
import requests
from bs4 import BeautifulSoup
import math
import matplotlib.pyplot as plt
import seaborn as sns
import gender_guesser.detector as gender_detector

# APUESTAS

# 2023
path = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/Eurovisión
Definitivo/Eurovisión23Apuestas.csv'
datos = pd.read_csv(path, encoding='utf-8')
datos = datos[datos['pais'] != 'pais']
datos = datos.drop(datos.columns[0], axis=1)

# Rellenamos los valores faltantes en las variables Artista y Canción
datos.loc[datos['pais'] == 'Sweden', 'artista'] = 'Loreen'
datos.loc[datos['pais'] == 'Sweden', 'cancion'] = 'Tattoo'
datos.loc[datos['pais'] == 'Portugal', 'artista'] = 'Mimicat'
```

```
datos.loc[datos['pais'] == 'Portugal', 'cancion'] = 'Ai Coração'
datos.loc[datos['pais'] == 'Georgia', 'artista'] = 'Iru'
datos.loc[datos['pais'] == 'Georgia', 'cancion'] = 'Echo'

# Corregimos algunos datos
datos.loc[datos['pais'] == 'UKUnited', 'artista'] = 'Mae Muller'
datos.loc[datos['pais'] == 'UKUnited', 'cancion'] = 'I Wrote A Song'
datos.loc[datos['pais'] == 'UKUnited', 'pais'] = 'United Kingdom'
datos.loc[datos['pais'] == 'San', 'artista'] = 'Piqued Jacks'
datos.loc[datos['pais'] == 'San', 'pais'] = 'San Marino'
datos.loc[datos['pais'] == 'Moldova', 'artista'] = 'Pasha Parfeni'

# Hacemos melt de las variables de casas de apuestas para agruparlas en una sola
id_vars = ['pais', 'artista', 'cancion', 'winning chance', 'fecha']  # Variables que no se van a
convertir en valores
value_vars = list(set(datos.columns) - set(id_vars))  # Variables que se convertirán en
valores

datos_melt = pd.melt(datos, id_vars=id_vars, value_vars=value_vars,
var_name='Casa_apuesta', value_name='Valor_casa_apuesta')
datos_melt['fecha'] = pd.to_datetime(datos_melt['fecha']).dt.floor('D') # Quitamos la hora
de extracción de datos ya que será irrelevante
datos_melt['Valor_casa_apuesta'] = pd.to_numeric(datos_melt['Valor_casa_apuesta'],
errors='coerce')

# Obtenemos los nombres de las columnas actuales
columnas_actuales = datos_melt.columns.tolist()

# Modificamos el nombre de la tercera columna
columnas_actuales[3] = 'Probabilidad_ganar'  # Cambia el índice [2] por el número de
columna que desees

# Asignamos los nuevos nombres de las columnas al dataframe
datos_melt.columns = columnas_actuales

# Creamos un rango de fechas entre el 17 de marzo y el 13 de mayo
fechas_nuevas = pd.date_range(start='2023-03-17', end='2023-05-14', closed='left')

# Creamos un dataframe vacío para almacenar los nuevos registros generados
nuevos_registros = pd.DataFrame(columns=datos_melt.columns)

# Iteramos por cada conjunto de Pais, Artista, Canción y Casa_apuesta en el dataframe
original
for _, grupo in datos_melt.groupby(['pais', 'artista', 'cancion', 'Casa_apuesta']):
    pais, artista, cancion, casa_apuesta = _
```

```
    # Obtenemos los registros existentes para el conjunto actual
    registros_actuales = grupo.copy()

    # Obtenemos las fechas existentes en registros_actuales según la Casa_apuesta
    fechas_actuales = registros_actuales['fecha']

    # Buscamos las fechas faltantes en fechas_nuevas
    fechas_faltantes = set(fechas_nuevas) - set(fechas_actuales)

    # Verificamos si falta alguna fecha en registros_actuales y añadimos una fila con valores
faltantes
    for fecha in fechas_faltantes:
        nuevo_registro = {
            'pais': pais,
            'artista': artista,
            'cancion': cancion,
            'Casa_apuesta': casa_apuesta,
            'fecha': fecha,
        }
        registros_actuales = registros_actuales.append(nuevo_registro, ignore_index=True)

    # Añadimos los registros actuales al dataframe de nuevos registros
    nuevos_registros = nuevos_registros.append(registros_actuales)

datos_finales = nuevos_registros

# Rellenamos los valores faltantes con la media correspondiente
datos_finales['Valor_casa_apuesta'] = datos_finales.groupby(['fecha',
'cancion'])['Valor_casa_apuesta'].transform(lambda x: x.fillna(x.mean()))

# Transformamos la variable Probabilidad_ganar de porcentaje a probabilidad
datos_finales['Probabilidad_ganar'] = np.where(datos_finales['Probabilidad_ganar'] ==
'<1%', '0.5%', datos_finales['Probabilidad_ganar'])
datos_finales['Probabilidad_ganar'] = datos_finales['Probabilidad_ganar'].str.replace('%',
'')
datos_finales['Probabilidad_ganar'] = pd.to_numeric(datos_finales['Probabilidad_ganar'])
/ 100
datos_finales['Probabilidad_ganar'] = pd.to_numeric(datos_finales['Probabilidad_ganar'],
errors='coerce')

datos_finales['Probabilidad_ganar'] = datos_finales.groupby(['fecha',
'cancion'])['Probabilidad_ganar'].transform(lambda x: x.fillna(x.mean()))

# Interpolamos los valores que faltan de quedar eliminados
datos_finales = datos_finales.sort_values(by=['cancion', 'Casa_apuesta', 'fecha'])
```

```
datos_finales['Valor_casa_apuesta'] = datos_finales['Valor_casa_apuesta'].ffill()
datos_finales['Probabilidad_ganar'] = datos_finales['Probabilidad_ganar'].ffill()


# Exportamos la variable de los votos de la encuesta
path = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/Eurovisión
Definitivo/VotosEncuesta.xlsx'
votos = pd.read_excel(path)
votos.loc[votos['Pais'] == 'Austria', 'Canción'] = 'Who The Hell Is Edgar?'
votos.loc[votos['Pais'] == 'Italy', 'Canción'] = 'Due vite'
votos.loc[votos['Pais'] == 'Moldova', 'Canción'] = 'Soarele și Luna'
votos.loc[votos['Pais'] == 'Netherlands', 'Artista'] = 'Mia Nicolai & Dion Cooper'
votos.loc[votos['Pais'] == 'Lithuania', 'Artista'] = 'Monika Linkytė'
votos['Votos_Encuesta'] = pd.to_numeric(votos['Votos_Encuesta'], errors='coerce')

# Rellenamos los valores faltantes aleatorios usando interpolación por cada grupo
for _, grupo in votos.groupby(['Pais', 'Artista', 'Canción']):
   pais, artista, cancion = _
   registros_actuales = pd.DataFrame(grupo).copy()
   registros_actuales['Votos_Encuesta'].interpolate(method='linear', inplace=True)
   votos.loc[registros_actuales.index, 'Votos_Encuesta'] =
registros_actuales['Votos_Encuesta']

# Juntamos los datos y nos quedamos con los datos que precisamos en ambas bases
datos1 = pd.merge(datos_finales, votos, how='inner', left_on=['pais', 'artista', 'cancion',
'fecha'], right_on=['Pais', 'Artista', 'Canción', 'Fecha'])

datos1 = datos1.drop(datos1.columns[:3], axis=1)
datos1 = datos1.drop(datos1.columns[1], axis=1)

columnas_ordenadas = ['Pais', 'Artista', 'Canción', 'Probabilidad_ganar', 'Fecha',
'Casa_apuesta', 'Valor_casa_apuesta', 'Votos_Encuesta']  # Define el orden deseado de las
columnas
datos1 = datos1.reindex(columns=columnas_ordenadas)  # Reordenar las columnas

numeric_vars = datos1.select_dtypes(include=['float', 'int']).columns

# Análisis univariante para variables numéricas
for column in numeric_vars:
   variable = datos1[column]

   # Resumen estadístico de la variable
   summary = variable.describe()
   print(f"Resumen estadístico de {column}:")
   print(summary)
```

```
    # Histograma de la variable
    plt.hist(variable, bins=10)
    plt.xlabel('Valor')
    plt.ylabel('Frecuencia')
    plt.title(f'Histograma de {column}')
    plt.show()

    # Gráfico de caja y bigotes de la variable
    plt.boxplot(variable)
    plt.ylabel('Valor')
    plt.title(f'Diagrama de caja y bigotes de {column}')
    plt.show()

# Creamos el gráfico de dispersión para todas las variables numéricas
data_numeric = datos1.select_dtypes(include='number')
sns.pairplot(data_numeric)
plt.show()

# Calculamos la matriz de correlación
correlation_matrix = data_numeric.corr()

# Visualizamos la matriz de correlación utilizando un mapa de calor
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Matriz de correlación')
plt.show()

datos1 = datos1.drop(datos1.columns[[3, 7]], axis=1)

import numpy as np

def tratar_outliers(columna, threshold=1.5):
    # Calculamos el primer y tercer cuartil
    q1 = np.percentile(columna, 25)
    q3 = np.percentile(columna, 75)

    # Calculamos el rango intercuartil (IQR)
    iqr = q3 - q1

    # Calculamos los límites inferior y superior
    lower_bound = q1 - threshold * iqr
    upper_bound = q3 + threshold * iqr

    # Reemplazamos los valores atípicos por la media trimming (solo para valores por
encima del upper_bound)
    outliers_index = columna > upper_bound
    trimmed_mean = np.mean(columna[~outliers_index])
```

```
    columna[outliers_index] = trimmed_mean

    return columna

# Aplicamos el tratamiento de valores atípicos a la columna 'Valor_casa_apuesta' del
DataFrame 'datos1'
datos1['Valor_casa_apuesta'] = tratar_outliers(datos1['Valor_casa_apuesta'],
threshold=1.5)

apuestas2023 = datos1
numeric_vars = apuestas2023.select_dtypes(include=['float', 'int']).columns

# Análisis univariante para variables numéricas
for column in numeric_vars:
    variable = apuestas2023[column]

    # Resumen estadístico de la variable
    summary = variable.describe()
    print(f"Resumen estadístico de {column}:")
    print(summary)

    # Histograma de la variable
    plt.hist(variable, bins=10)
    plt.xlabel('Valor')
    plt.ylabel('Frecuencia')
    plt.title(f'Histograma de {column}')
    plt.show()

    # Gráfico de caja y bigotes de la variable
    plt.boxplot(variable)
    plt.ylabel('Valor')
    plt.title(f'Diagrama de caja y bigotes de {column}')
    plt.show()

# Exportamos los parámetros de spotify
path = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/Eurovisión
Definitivo/ParamSpotifyEuro.csv'
datos2 = pd.read_csv(path, encoding='utf-8')
columnas_eliminar = [datos2.columns[0], datos2.columns[3], datos2.columns[4],
datos2.columns[5], datos2.columns[7]]
datos2 = datos2.drop(columnas_eliminar, axis=1)

# Exportamos los tópicos de las canciones
path = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/Eurovisión
Definitivo/TopicosLyricsEurovision.csv'
datos3 = pd.read_csv(path, encoding='utf-8')
```

```
columnas_eliminar2 = [datos3.columns[2], datos3.columns[3]]
datos3 = datos3.drop(columnas_eliminar2, axis=1)

# Juntamos la base de datos entera
features2023 = pd.merge(datos2, datos3, how='left', on=['Artist', 'Title'])

detector = gender_detector.Detector()

# Iteramos sobre cada fila del DataFrame
for index, row in features2023.iterrows():
    # Obtener el valor de 'Artista' en la fila actual
    artista = row['Artist']

    # Realizamos una búsqueda en la página web para obtener información relacionada con
el artista
    url_busqueda = "https://es.wikipedia.org/wiki/" + artista

    # Realizamos la solicitud HTTP y obtener el contenido de la página
    respuesta = requests.get(url_busqueda)
    contenido = respuesta.text

    # Analizamos el contenido HTML con BeautifulSoup
    soup = BeautifulSoup(contenido, 'html.parser')

    # Buscamos la presencia de las palabras "hijo" y "hija" en el texto de la página
    texto_pagina = soup.get_text()
    if 'Años activo' in texto_pagina:
        sexo = 'Masculino'
    elif 'Años activa' in texto_pagina:
        sexo = 'Femenino'
    else:
        nombre = artista.split(' ')[0]
        gender = detector.get_gender(nombre)
        if gender == 'male':
            sexo = 'Masculino'
        elif gender == 'female':
            sexo = 'Femenino'

    # Asignamos el valor obtenido al DataFrame en una nueva columna 'Sexo'
    features2023.loc[index, 'Sexo'] = sexo

    # Mostramos el progreso
    print("Procesando fila", index)

features2023.loc[5, 'Sexo'] = 'Masculino'
features2023.loc[7, 'Sexo'] = 'Femenino'
```

```
features2023.loc[18, 'Sexo'] = 'Femenino'
features2023.loc[25, 'Sexo'] = 'Masculino'
features2023.loc[32, 'Sexo'] = 'Mixto'
features2023.loc[33, 'Sexo'] = 'Masculino'
features2023.loc[35, 'Sexo'] = 'Masculino'
features2023.loc[36, 'Sexo'] = 'Masculino'

features2023 = features2023.drop(features2023.columns[15:199], axis=1)
features2023 = features2023.drop(features2023.columns[4], axis=1)

categorical_vars = features2023.select_dtypes(include='object').columns

# Análisis univariante para variables categóricas
for column in categorical_vars:
   variable = features2023[column]

   # Conteo de frecuencias de la variable
   frequencies = variable.value_counts()
   print(f"Frecuencias de {column}:")
   print(frequencies)

   # Gráfico de barras de la variable
   plt.bar(frequencies.index, frequencies.values)
   plt.xlabel('Categoría')
   plt.ylabel('Frecuencia')
   plt.title(f'Gráfico de barras de {column}')
   plt.show()

data_numeric = features2023.select_dtypes(include='number')

# Calculamos la matriz de correlación
correlation_matrix = data_numeric.corr()

# Visualizamos la matriz de correlación utilizando un mapa de calor
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
annot_kws={"size": 8})
plt.title('Matriz de correlación')
plt.show()

features2023 = features2023.drop(features2023.columns[4], axis=1)

# 2023
url =
'https://es.wikipedia.org/wiki/Festival_de_la_Canci%C3%B3n_de_Eurovisi%C3%B3n_2023
'
page = requests.get(url)
```

```
# Parseamos el contenido HTML con BeautifulSoup
soup = BeautifulSoup(page.content, 'html.parser')

# Buscamos el encabezado de la tabla "Participants and results"
result_header = soup.find('span', id='Final')
result_header1 = soup.find('span', id='Semifinal_1')
result_header2 = soup.find('span', id='Semifinal_2')

# Navegamos hasta el elemento "table" correspondiente
table = result_header.find_next('table')
table1 = result_header1.find_next('table')
table2 = result_header2.find_next('table')

# Extraemos los datos de la tabla y los almacenamos en un DataFrame
rows = table.find_all('tr')
data = []
for row in rows[1:]:
    cols = row.find_all('td')
    cols = [col.text.strip() for col in cols]
    data.append(cols)

rows1 = table1.find_all('tr')
data1 = []
for row in rows1[1:]:
    cols = row.find_all('td')
    cols = [col.text.strip() for col in cols]
    data1.append(cols)

rows2 = table2.find_all('tr')
data2 = []
for row in rows2[1:]:
    cols = row.find_all('td')
    cols = [col.text.strip() for col in cols]
    data2.append(cols)

df = pd.DataFrame(data)
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

# Transformamos las bases de datos a nuestro gusto, eliminando las filas y columnas
innecesarias
columnas_eliminar = df.columns[[0]]

df = df.drop(columnas_eliminar, axis=1)
df1 = df1.drop(columnas_eliminar, axis=1)
```

```
df2 = df2.drop(columnas_eliminar, axis=1)

df = df.rename(columns={1: 'País', 2: 'Interprete', 3: 'Canción', 4: 'Posición', 5:
'Puntos_Totales'})
df1 = df1.rename(columns={1: 'País', 2: 'Interprete', 3: 'Canción', 4: 'Posición', 5:
'Puntos_Totales'})
df2 = df2.rename(columns={1: 'País', 2: 'Interprete', 3: 'Canción', 4: 'Posición', 5:
'Puntos_Totales'})

# Juntamos las 2 bases de datos de semis y nos quedamos con los eliminados
dftotal = pd.concat([df1,df2])

columnas_numericas = ['Posición']
dftotal[columnas_numericas] = dftotal[columnas_numericas].apply(pd.to_numeric,
errors='coerce')
df[columnas_numericas] = df[columnas_numericas].apply(pd.to_numeric, errors='coerce')

elim = dftotal['Posición'] > 10
datos_filtrados = dftotal[elim]
datos_filtrados['Posición'] = datos_filtrados['Puntos_Totales'].rank(ascending=False)

# Damos mejor posición a San Marino por gusto propio, al tener ambos 0 puntos
datos_filtrados.loc[datos_filtrados['Interprete'] == 'Piqued Jacks', 'Posición'] = 10
datos_filtrados.loc[datos_filtrados['Interprete'] == 'Theodor Andrei', 'Posición'] = 11

datos_filtrados['Posición'] = datos_filtrados['Posición'] + 26

dftotal1 = pd.concat([df,datos_filtrados])
columnas = ['País', 'Interprete', 'Canción', 'Posición']
var_resp_2023 = dftotal1[columnas]

subset2023 = apuestas2023[['Pais', 'Artista', 'Canción']]
subset2023_unicos = subset2023.drop_duplicates(['Pais', 'Artista', 'Canción'])
indicador2023.loc[indicador2023['label'] == 'Czech Republic', 'label'] = 'Czechia'
previo2023 = pd.merge(subset2023_unicos, indicador2023, how='inner', left_on=['Pais'],
right_on=['label'])
previo2023 = previo2023.rename(columns={'cluster': 'Amistad'})
previo2023 = previo2023.drop(previo2023.columns[[3, 4, 6, 7]], axis=1)

features2023.loc[features2023['Artist'] == 'ALIKA', 'Artist'] = 'Alika'
previo2023.loc[previo2023['Artista'] == 'Albina & Familja Kelmendi', 'Artista'] = 'Albina
Kelmendi'
features2023.loc[features2023['Artist'] == 'BLANKA', 'Artist'] = 'Blanka'
features2023.loc[features2023['Artist'] == 'Mia Nicolai', 'Artist'] = 'Mia Nicolai & Dion
Cooper'
features2023.loc[features2023['Artist'] == 'Various Artists', 'Artist'] = 'Mimicat'
```

```
previo2023.loc[previo2023['Artista'] == 'Monika Linkytė', 'Artista'] = 'Monika Linkyte'
features2023.loc[features2023['Artist'] == 'TEYA', 'Artist'] = 'Teya & Salena'
features2023.loc[features2023['Artist'] == 'TVORCHI', 'Artist'] = 'Tvorchi'

features2023_2 = pd.merge(features2023, previo2023, how='left', left_on=['Artist'],
right_on=['Artista'])
features2023_2 = features2023_2.drop(features2023_2.columns[[17, 18]], axis=1)

apuestas2023.to_csv("C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de
datos/Eurovisión Predicción/apuestas2023.csv")
features2023_2.to_csv("C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de
datos/Eurovisión Predicción/features2023.csv")
var_resp_2023.to_csv("C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de
datos/Eurovisión Predicción/var_resp_2023.csv")
```

Último preprocesamiento antes de la creación de modelos para Eurovisión entero

```
# Cargamos las librerias necesarias
## Librerias de tratamiento de los datos
import numpy as np
import pandas as pd
import re

## Tratamiento de fechas
from datetime import datetime

#
==========================================================================
====
# Creamos los path
#INICIO = 'E:/DOCENCIA/TFG Alumnos/2022-2023/POL SATÓ/syntax/Bases de datos/'
INICIO = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/'
festival = 'Eurovisión Predicción/'

# Leemos la base de datos
apuestas21 = pd.read_csv(INICIO + festival + "apuestas2021.csv")
apuestas22 = pd.read_csv(INICIO + festival + "apuestas2022.csv")
apuestas23 = pd.read_csv(INICIO + festival + "apuestas2023.csv")

apuestas23.rename(columns={'Artista': 'Interprete'}, inplace=True)
apuestas23.rename(columns={'Pais': 'País'}, inplace=True)



# Convertimos la cadena a un objeto datetime
apuestas21['Fecha'] = pd.to_datetime(apuestas21['Fecha'])
```

```
apuestas22['Fecha'] = pd.to_datetime(apuestas22['Fecha'])
apuestas23['Fecha'] = pd.to_datetime(apuestas23['Fecha'])

# ------------------------------------------------------------------------
# Calculamos la diferencia que hay entre el dia de la gala y el dia de la casa de
# apuestas
fechaFinal21 = 20210522
apuestas21['delay'] = [datetime.strptime(str(fechaFinal21), '%Y%m%d') - d for d in
apuestas21['Fecha']]
apuestas21['delay'] = apuestas21['delay'].astype(str)

fechaFinal22 = 20220514
apuestas22['delay'] = [datetime.strptime(str(fechaFinal22), '%Y%m%d') - d for d in
apuestas22['Fecha']]
apuestas22['delay'] = apuestas22['delay'].astype(str)

fechaFinal23 = 20230513
apuestas23['delay'] = [datetime.strptime(str(fechaFinal23), '%Y%m%d') - d for d in
apuestas23['Fecha']]
apuestas23['delay'] = apuestas23['delay'].astype(str)

# Eliminamos aquellas columnas que no necesitaremos usar
colsEliminar = ['Fecha']
apuestas21 = apuestas21.drop(colsEliminar, axis = 1)
apuestas22 = apuestas22.drop(colsEliminar, axis = 1)
apuestas23 = apuestas23.drop(colsEliminar, axis = 1)

# Creamos una columna de año, para poder diferenciar en cada año
apuestas21['Anyo'] = '2021'
apuestas22['Anyo'] = '2022'
apuestas23['Anyo'] = '2023'

# Juntamos las dos bases de datos para tener una única
apuestas = pd.concat([apuestas21, apuestas22, apuestas23])

# Pivotamos la tabla para tener:
## filas: Las canciones individuos
## columnas: el retardo del indicador

indices = ['Interprete', 'Canción', 'Casa_apuesta', 'Anyo']
apuestas = apuestas.pivot(index = indices, columns = 'delay', values =
'Valor_casa_apuesta')

# Ordenamos las columnas por nombre
apuestas = apuestas[sorted(apuestas.columns, key = lambda x: int(x.split(' ')[0]) if
x[0].isdigit() else float('inf'))]
```

```
apuestas = apuestas.reset_index()

# Obtener el índice de la columna inicial y final
indice_inicial = apuestas.columns.get_loc('58 days')
indice_final = apuestas.columns.get_loc('79 days')

# Eliminar las columnas en el rango determinado
apuestas.drop(apuestas.columns[indice_inicial:indice_final+1], axis=1, inplace=True)

# Añadimos la variable respuesta. Sera necesaria para poder hacer la red neuronal
# que nos permita predecir el número de votos correspondientes
resp21 = pd.read_csv(INICIO + festival + "var_resp_2021.csv")
resp22 = pd.read_csv(INICIO + festival + "var_resp_2022.csv")
resp23 = pd.read_csv(INICIO + festival + "var_resp_2023.csv")

# Agregamos las tablas en una misma
respuestas = pd.concat([resp21, resp22, resp23])

# Le quitamos los signos extraños y espacios de más a los datos
respuestas['Canción'] = [re.sub("«|»|» ", "", c) for c in respuestas['Canción']]

# Añadimos a las casas de apuestas la información correspondiente a la variable
# respuesta
varRespuesta = ['Canción', 'Posición']
apuestas = pd.merge(apuestas, respuestas[varRespuesta], on = 'Canción', how = 'inner')

## Features
### Leemos los datos
features21 = pd.read_csv(INICIO + festival + 'features2021.csv')
features21['Anyo'] = "2021"
features22 = pd.read_csv(INICIO + festival + 'features2022.csv')
features22['Anyo'] = "2022"
features23 = pd.read_csv(INICIO + festival + 'features2023.csv')
features23['Anyo'] = "2023"

features21.rename(columns={'Canción_x': 'Canción'}, inplace=True)
features22.rename(columns={'Canción_x': 'Canción'}, inplace=True)
features23.rename(columns={'Artist': 'Interprete'}, inplace=True)
features23.rename(columns={'Title': 'Canción'}, inplace=True)
features23.rename(columns={'Pais': 'País'}, inplace=True)

### Juntamos la información
features = pd.concat([features21, features22, features23])

### Creamos las dicotomicas faltantes de algunas de las variables
dummies = pd.get_dummies(features[['Sexo', 'Tópico']])
```

```
# Eliminar las columnas originales
features = features.drop(['Sexo', 'Tópico'], axis = 1)

# Concatenar las variables dummy al DataFrame original
features = pd.concat([features, dummies], axis = 1)
features = features.drop(['Unnamed: 0'], axis = 1)

# Concatenamos las dos bases de datos a partir de un merge
apuestas = apuestas.drop(['Interprete'], axis=1)
eurovision = features.merge(apuestas, on='Canción')

pd.crosstab(eurovision['Interprete'], eurovision['Casa_apuesta'])
eurovision = eurovision.loc[eurovision['Casa_apuesta'].isin(['BET365'])]

# Eliminamos algunas variables por no ser necesarias
colsEliminar = ["Anyo_x", "Anyo_y", "Casa_apuesta"]
eurovision = eurovision.drop(colsEliminar, axis = 1)

# Guardamos la base de datos creada en un pickle
#eurovision.to_pickle(INICIO + festival + 'eurovision.pkl')
```

Diagrama de flujos para algunas bases de datos

```
import graphviz

# Crear el gráfico
graph = graphviz.Digraph('ER Diagram', filename='er_diagram.gv', format='png',
node_attr={'shape': 'record'})

# Definir los nodos del diagrama y las variables dentro de cada nodo con títulos
graph.node('Cuotas y votaciones', 'Cuotas y votaciones|{<f0> Interprete|<f1>
Canción|<f2> Fecha|<f3> Probabilidad ganar|<f4> Casa de apuesta|<f5> Valor casa de
apuesta|<f6> Votos encuesta}')

# Guardar y mostrar el diagrama
graph.render(view=True)

# Crear el gráfico
graph = graphviz.Digraph('ER Diagram', filename='er_diagram.gv', format='png',
node_attr={'shape': 'record'})

graph.node('Parámetros Spotify', 'Parámetros Spotify|{<f0> Interprete|<f1> Canción|<f2>
Parámetros Spotify|...}')
graph.node('Topic Modelling', 'Topic Modelling|{<f0> Title|<f1> Artist|<f2> Lyrics|<f3>
Lemas|<f4> Tópico}')
```

```
graph.node('Indicador de Amistad', 'Indicador de Amistad|{<f0> País|<f1> Indicador|<f2> Modularidad|<f3> Año}')

# Definir las conexiones entre nodos
graph.edge('Parámetros Spotify:f0', 'Topic Modelling:f1')
graph.edge('Parámetros Spotify:f1', 'Topic Modelling:f0')

# Guardar y mostrar el diagrama
graph.render(view=True)

# Crear el gráfico
graph = graphviz.Digraph('ER Diagram', filename='er_diagram.gv', format='png', node_attr={'shape': 'record'})

graph.node('Parámetros Spotify', 'Parámetros Spotify|{<f0> País|<f1> Interprete|<f2> Canción|<f3> Parámetros Spotify|...}')
graph.node('Topic Modelling', 'Topic Modelling|{<f0> País|<f1> Title|<f2> Artist|<f3> Lyrics|<f4> Lemas|<f5> Tópico}')
graph.node('Indicador de Amistad', 'Indicador de Amistad|{<f0> País|<f1> Indicador|<f2> Modularidad|<f3> Año}')

# Definir las conexiones entre nodos
graph.edge('Parámetros Spotify:f0', 'Topic Modelling:f0')
graph.edge('Parámetros Spotify:f1', 'Topic Modelling:f2')
graph.edge('Parámetros Spotify:f2', 'Topic Modelling:f1')
graph.edge('Topic Modelling:f0', 'Indicador de Amistad:f0')

# Guardar y mostrar el diagrama
graph.render(view=True)

# Crear el gráfico
graph = graphviz.Digraph('ER Diagram', filename='er_diagram.gv', format='png', node_attr={'shape': 'record'})

graph.node('Apuestas', 'Apuestas|{<f0> País*|<f1> Interprete|<f2> Canción|<f3> Casa_apuesta|<f4> Año|<f5> Delays}')
graph.node('Features', 'Features|{<f0> País*|<f1> Interprete|<f2> Canción|<f3> Parámetros Spotify|<f4> Indicador Amistad|<f5> Dummies Sexo|<f6> Dummies Tópico}')
graph.node('Variable respuesta', 'Variable respuesta|{<f0> País*|<f1> Interprete|<f2> Canción|<f3> Posición}')

# Definir las conexiones entre nodos
graph.edge('Apuestas:f0', 'Variable respuesta:f0')
graph.edge('Apuestas:f1', 'Variable respuesta:f1')
graph.edge('Apuestas:f2', 'Variable respuesta:f2')
graph.edge('Apuestas:f0', 'Features:f0')
```

```
graph.edge('Apuestas:f1', 'Features:f1')
graph.edge('Apuestas:f2', 'Features:f2')

# Guardar y mostrar el diagrama
graph.render(view=True)
```

Modelaje y evalución de modelos

```
# Cargamos las librerias necesarias
## Librerias de tratamiento de los datos
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

# Creamos los path
#INICIO = 'E:/DOCENCIA/TFG Alumnos/2022-2023/POL SATÓ/syntax/Bases de datos/'
INICIO = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/'
festival = 'Eurovisión Predicción/'

# Leemos la base de datos
## Eurovisión
eurovision = pd.read_pickle(INICIO + festival + 'eurovision.pkl')
colsEliminar = ["0 days"]
eurovision = eurovision.drop(colsEliminar, axis = 1)



#### Dividimos los datos en entrneamiento y test
from sklearn.model_selection import train_test_split
X = eurovision.drop(["Posición", "País", "Interprete", "Canción"], axis = 1)
y = eurovision["Posición"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)

from sklearn.ensemble import ExtraTreesRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score as acc
import seaborn as sns

# Generamos un gráfico con la importance matriz de variables importantes para el
# modelo
n = 20 ## Número de variables que queremos expresar
reg = ExtraTreesRegressor()
reg.fit(X_train,y_train)
reg.feature_importances_
feat_importances = pd.Series(reg.feature_importances_, index=X_train.columns)
fi = feat_importances.nlargest(n)
```

```
features = pd.DataFrame({'variable': fi.index.tolist(), 'valor': fi})
features = features.reset_index()

# create horizontal barplot
fig, axes = plt.subplots(nrows = 1, ncols = 1, dpi=600)
sns.barplot(x = features.valor, y = features.variable, orient='h');
fig.savefig(INICIO + festival + 'ImportanceMatrixDecisionTree.png')

# Vamos a buscar los parámetros a partir del tunning
from sklearn.tree import DecisionTreeRegressor
reg_decision_model = DecisionTreeRegressor()
## reg_decision_model.fit(X_train,y_train)
## reg_decision_model.score(X_train,y_train)
## reg_decision_model.score(X_test,y_test)
## prediction = reg_decision_model.predict(X_test)

'''
## Declaramos los parámetros que vamos a intentar localizar los mejores
parameters={"splitter":["best","random"],
        "max_depth" : [1,3,5,7,9,11,12],
       "min_samples_leaf":[1,2,3,4,5,6,7,8,9,10],
       "min_weight_fraction_leaf":[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
       "max_features":["auto","log2","sqrt",None],
       "max_leaf_nodes":[None,10,20,30,40,50,60,70,80,90] }

# calculating different regression metrics
from sklearn.model_selection import GridSearchCV
tuning_model = GridSearchCV(reg_decision_model, param_grid = parameters,
                    scoring = 'neg_mean_squared_error', cv = 3,
                    verbose = 3)

# Entrenamos el modelo
tuning_model.fit(X_train,y_train)
# Localizamos el mejor parámetros
tuning_model.best_params_
# best model score
# tuning_model.best_score_
'''

# Fijamos la semilla para reproducibilidad
np.random.seed(100)

# Creamos el modelo con los mejores parámetros
tuned_hyper_model= DecisionTreeRegressor(max_depth = 3,
                            max_features = 'log2',
```

```
                            max_leaf_nodes = 50,
                            min_samples_leaf = 3,
                            min_weight_fraction_leaf = 0.1,
                            splitter = 'random')
# Entrenamos el modelo
tuned_hyper_model.fit(X_train,y_train)

# ------------------------------------------------------------------------
# Realizamos el árbol que nos ha salido de nuestros datos
from sklearn import tree
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
tree.plot_tree(tuned_hyper_model,
          feature_names = X_train.columns);
fig.savefig(INICIO + festival + 'arboldeDecision.png')

# ------------------------------------------------------------------------
# Realizamos la predicción del modelo en los datos de test
tuned_predtrain = tuned_hyper_model.predict(X_train)
tuned_predtest = tuned_hyper_model.predict(X_test)

# realizamos el gráfico de dispersión entre el valor real y el valor predicho
plt.scatter(y_test,tuned_predtest)

# Realizamos el gráfico de errores
sns.distplot(y_test-tuned_predtest)

# Vemos los valores obtenidos
from sklearn import metrics
print('MAE (train):', metrics.mean_absolute_error(y_train,tuned_predtrain))
print('MSE (train):', metrics.mean_squared_error(y_train, tuned_predtrain))
print('RMSE (train):', np.sqrt(metrics.mean_squared_error(y_train, tuned_predtrain)))
print("=============================================")
print('MAE (test):', metrics.mean_absolute_error(y_test,tuned_predtest))
print('MSE (test):', metrics.mean_squared_error(y_test, tuned_predtest))
print('RMSE (test):', np.sqrt(metrics.mean_squared_error(y_test, tuned_predtest)))

from sklearn.metrics import r2_score
print('R^2:', r2_score(y_train, tuned_predtrain))

# XGBOOST:
import xgboost
from xgboost import XGBRegressor

# create an xgboost regression model
model = xgboost.XGBRegressor()
```

```
# create an xgboost regression model
model = XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.7,
colsample_bytree=0.8)

#Creating the model on Training Data
XGB = model.fit(X_train, y_train)

# Generamos un gráfico con la importance matriz de variables importantes para el
# modelo
n = 20 ## Número de variables que queremos expresar
XGB.feature_importances_
feat_importances = pd.Series(XGB.feature_importances_, index=X_train.columns)
fi = feat_importances.nlargest(n)

features = pd.DataFrame({'variable': fi.index.tolist(), 'valor': fi})
features = features.reset_index()

# create horizontal barplot
fig, axes = plt.subplots(nrows = 1, ncols = 1, dpi=600)
sns.barplot(x = features.valor, y = features.variable, orient='h');
fig.savefig(INICIO + festival + 'ImportanceMatrixXGBoost.png')

from xgboost import plot_tree
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(12, 12), dpi=300)
plot_tree(XGB, num_trees=0, ax=ax, rounded=False)

# Guardar la imagen en alta resolución
plt.savefig('tree.png', dpi=300)

prediction = XGB.predict(X_test)

# Definimo el método de evaluación del modelo
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import cross_val_score
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
scores = cross_val_score(model, X_train, y_train, scoring='neg_mean_absolute_error',
cv=cv, n_jobs=-1)

#Measuring Goodness of fit in Training data
# force scores to be positive
scores = np.absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (scores.mean(), scores.std()) )
```

```
from sklearn import metrics
print('R2 Value:',metrics.r2_score(y_train, XGB.predict(X_train)))

# make a prediction
yhat = model.predict(X_test)

# realizamos el gráfico de dispersión entre el valor real y el valor predicho
plt.scatter(y_test,yhat)

# Realizamos el gráfico de errores
sns.distplot(y_test-yhat)

print('MAE (train):', metrics.mean_absolute_error(y_train,XGB.predict(X_train)))
print('MSE (train):', metrics.mean_squared_error(y_train, XGB.predict(X_train)))
print('RMSE (train):', np.sqrt(metrics.mean_squared_error(y_train, XGB.predict(X_train))))
print("=============================================")
print('MAE (test):', metrics.mean_absolute_error(y_test,prediction))
print('MSE (test):', metrics.mean_squared_error(y_test, prediction))
print('RMSE (test):', np.sqrt(metrics.mean_squared_error(y_test, prediction)))

# Cargmamos los datos necesarios
INICIO = 'C:/Users/polgs/Desktop/TFG/Data_EurovisionAnalysis/Bases de datos/'
festival = 'Eurovisión Predicción/eurovision.csv'

eurovision = pd.read_csv(INICIO + festival)
colsEliminar = ['Unnamed: 0']
eurovision = eurovision.drop(colsEliminar, axis = 1)

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

#### Dividimos los datos en entrneamiento y test
from sklearn.model_selection import train_test_split
X = eurovision.drop(["Posición", "País", "Interprete", "Canción"], axis = 1)
y = eurovision["Posición"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)

# REDES NEURONALES:
# Regresion ejemplo con viviendalondres Dataset: Standardized y red mas amplia
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense

# Creamos el modelo
model = Sequential()
model.add(Dense(300, input_dim=len(X_train.columns), kernel_initializer='normal',
activation='relu'))
model.add(Dense(25, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))


# Imprimir la estructura de la red neuronal
model.summary()

# Visualizamos la red neuronal
import tensorflow as tf
from tensorflow import keras
import pydot

# tf.keras.utils.plot_model(
# model,
# to_file=path + festival + "model.png",
# show_shapes=True,
# show_dtype=False,
# show_layer_names=True,
# rankdir="TB",
# expand_nested=True,
# dpi=96,
# layer_range=None,
# show_layer_activations=True,
# )

import visualkeras
from PIL import ImageFont
visualkeras.layered_view(model, legend=True)


model.compile(loss='mean_squared_error',
          optimizer='adam')

historico = model.fit(X_train, y_train, epochs=30000)
```

```
# 6. Calculating the loss after training
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# realizamos el gráfico de dispersión entre el valor real y el valor predicho
plt.scatter(y_test,y_pred_test)

# Realizamos el gráfico de errores
import seaborn as sns
sns.distplot(y_test-np.concatenate(y_pred_test))

# Vemos los valores obtenidos
from sklearn import metrics
print('MAE (train):', metrics.mean_absolute_error(y_train,y_pred_train))
print('MSE (train):', metrics.mean_squared_error(y_train, y_pred_train))
print('RMSE (train):', np.sqrt(metrics.mean_squared_error(y_train, y_pred_train)))
print("==============================================")
print('MAE (test):', metrics.mean_absolute_error(y_test,y_pred_test))
print('MSE (test):', metrics.mean_squared_error(y_test, y_pred_test))
print('RMSE (test):', np.sqrt(metrics.mean_squared_error(y_test, y_pred_test)))

from sklearn.metrics import r2_score
print('R^2:', r2_score(y_train, y_pred_train))
```