Clasificación de Canciones Pop: Aplicando Minería de Datos en el Género Musical

Autor:
Patricio Abarca

Facultad de Ingeniería
Departamento de Informática y Computación
Minería de Datos
Profesor: Pablo Figueroa Plaza

Enero del 2025

Índice

1. Introducción

En la era digital, el análisis de datos se ha convertido en una herramienta esencial para resolver problemas complejos y mejorar procesos en diversas industrias, incluida la musical. Plataformas como Spotify no solo han transformado la forma en que los usuarios consumen música, sino que también han generado grandes volúmenes de datos sobre características musicales y preferencias de los oyentes. Este contexto ha abierto nuevas oportunidades para aplicar técnicas de análisis avanzado y aprendizaje automático, particularmente en el estudio y clasificación de géneros musicales.

La música pop, como uno de los géneros más populares e influyentes a nivel global, destaca por su capacidad de conectar con audiencias diversas y por ocupar un lugar predominante en las plataformas de streaming. Este género presenta características únicas que pueden ser aprovechadas para su identificación mediante modelos predictivos, lo que resulta especialmente útil en aplicaciones como la personalización de recomendaciones, la creación de listas de reproducción y la optimización de experiencias de usuario.

El presente proyecto aborda la identificación del género pop desde un enfoque basado en el análisis de datos. Utilizando información recopilada de Spotify, se exploran patrones y características que permiten diferenciar este género de otros, mostrando cómo el procesamiento y modelado de datos pueden contribuir al entendimiento y clasificación musical.

2. Marco Teórico

2.1. Metodologías

El proyecto sigue una metodología basada en el aprendizaje automático supervisado para la clasificación de canciones en categorías como ”popz ”no pop”. La metodología se basa en los siguientes pasos:

- **Preprocesamiento de Datos**: Antes de entrenar los modelos, se realiza un proceso de limpieza de los datos, que incluye la eliminación de variables irrelevantes, la gestión de valores faltantes, y la transformación de las características categóricas utilizando codificación one-hot.
- **División del Conjunto de Datos**: El conjunto de datos se divide en conjuntos de entrenamiento y prueba mediante técnicas de partición, como la validación cruzada estratificada, para asegurar que los modelos se entrenen y evalúen con representaciones balanceadas de ambas clases.
- **Balanceo de Clases**: Dado el desbalanceo de clases, se implementan técnicas como el SMOTE (Synthetic Minority Over-sampling Technique) para generar datos sintéticos de la clase minoritaria, mejorando así la capacidad de los modelos para clasificar correctamente la clase ”pop”.
- **Entrenamiento y Evaluación de Modelos**: Se entrenan múltiples modelos de clasificación y se evalúan mediante métricas de rendimiento, como la precisión, la matriz de confusión, y el área bajo la curva ROC (AUC). Además, se aplica validación cruzada para evaluar la robustez de los modelos y evitar sobreajuste.

2.2. Modelos

Los modelos utilizados en este proyecto son:

- **Regresión Logística**: Un modelo lineal ampliamente utilizado para clasificación binaria. Este modelo predice la probabilidad de que una observación pertenezca a una de las dos clases (en este caso, ’pop’ y ’no pop’). A pesar de ser relativamente simple, la regresión logística ha demostrado ser eficaz para conjuntos de datos con características numéricas y categóricas.

- **XGBoost (eXtreme Gradient Boosting)**: XGBoost es una biblioteca de aprendizaje automático de código abierto y distribuida que implementa el algoritmo de boosting, utilizando árboles de decisión potenciados por gradiente. Este enfoque de aprendizaje supervisado emplea técnicas de descenso por gradiente para mejorar la precisión del modelo mediante la combinación secuencial de árboles de decisión. XGBoost es ampliamente reconocido por su eficiencia, velocidad y capacidad para manejar grandes volúmenes de datos, lo que lo convierte en una opción destacada en tareas de clasificación y regresión, especialmente en conjuntos de datos desbalanceados.

2.3. Herramientas

Para implementar los modelos y metodologías anteriores, se utilizan diversas herramientas de programación y bibliotecas de Python, entre las cuales destacan:

- **Google Colab**: Un entorno de desarrollo basado en la nube que permite ejecutar código Python de manera eficiente sin necesidad de configuración local. Colab ofrece acceso a recursos de hardware como GPUs, lo que facilita el entrenamiento de modelos complejos.
- **Python**: El lenguaje de programación principal utilizado en este proyecto, conocido por su flexibilidad y amplia gama de bibliotecas de ciencia de datos.
- **Pandas**: Se utiliza para la manipulación y análisis de datos. Permite trabajar con estructuras de datos como DataFrames, facilitando la limpieza, transformación y exploración de datos.
- **Scikit-learn**: Una biblioteca clave para la implementación de modelos de aprendizaje automático. En este proyecto, se utiliza para entrenar la regresión logística, la validación cruzada y el escalado de las características. También se emplea para evaluar la precisión de los modelos y generar la matriz de confusión y los informes de clasificación.
- **Matplotlib**: Biblioteca para la visualización de datos. Se utiliza para generar gráficos como la curva ROC, la matriz de confusión y otros gráficos que ayudan a interpretar los resultados de los modelos.
- **Imbalanced-learn**: Esta biblioteca ofrece implementaciones de técnicas de re-muestreo como SMOTE y SMOTEENN (SMOTE con edición de ejemplos negativos), que son fundamentales para manejar el desbalanceo de clases en el conjunto de datos.

3. Comprensión del Negocio

3.1. Definición del Problema

En la industria de la música, las plataformas de streaming como Spotify contienen un enorme volumen de canciones de diversos géneros. El desafío radica en identificar de manera automática y precisa si una canción pertenece al género pop o no, dado que este es uno de los géneros más populares y tiene una gran influencia en las listas de reproducción y las recomendaciones algorítmicas.

El problema específico que se aborda en este proyecto es la clasificación de canciones en dos categorías: pop y no pop. Esto se puede utilizar para mejorar los sistemas de recomendación, optimizar listas de reproducción y mejorar la experiencia de los usuarios de las plataformas de streaming, lo que indirectamente incrementaría la satisfacción y la retención.

3.2. Solución Propuesta

La solución propuesta consiste en aplicar técnicas de minería de datos, específicamente el uso de modelos de clasificación basados en algoritmos de machine learning, para predecir si una canción es pop o no. La metodología incluye los siguientes pasos:

- **Preprocesamiento de datos:** Preparar los datos para el entrenamiento de los modelos, eliminando columnas irrelevantes, manejando valores nulos y normalizando las características.
- **Creación del modelo:** Implementar varios algoritmos de clasificación para entrenar el modelo de clasificación.
- **Balanceo de clases:** Aplicar técnicas para manejar el desbalance en las clases y asegurar que el modelo no se sesgue hacia la clase mayoritaria.
- **Evaluación y optimización:** Utilizar métricas para evaluar el rendimiento del modelo, y ajustar parámetros según sea necesario.

3.3. Objetivos

3.3.1. Objetivo General

Desarrollar un modelo de clasificación para predecir la categoría musical de las canciones como pop o no pop.

3.3.2. Objetivos Específicos

- Identificar y seleccionar las características más relevantes que influyen en la clasificación de las canciones.
- Desarrollar un modelo de clasificación que permita distinguir entre canciones pop y no pop.
- Evaluar el desempeño del modelo mediante métricas de clasificación estándar, como precisión y recall.
- Mejorar la capacidad del modelo para generalizar a nuevos datos, reduciendo posibles sesgos.
- Ajustar el modelo para maximizar la precisión en la predicción de canciones pop y no pop.
- Analizar los resultados obtenidos para identificar patrones clave que definan el género pop.

4. Comprensión de los Datos

4.1. Descripción del Dataset

El conjunto de datos proviene de la API de Spotify, obteniendo canciones populares y no populares, junto con sus características tanto descriptivas como de audio. Las características descriptivas incluyen información sobre el nombre de la canción, el nombre del artista y el álbum, así como la fecha de lanzamiento del álbum. Las características de audio son resultado del análisis realizado por Spotify, como la energía, el tempo, la bailabilidad y la valencia. A continuación se describen las características utilizadas en la fase de experimentación:

4.2. Descripción de Variables

4.2.1. Características de Audio

- **Energy**: Medida de la intensidad y actividad de la canción. Las canciones energéticas suelen ser rápidas, ruidosas y fuertes.
- **Tempo**: Velocidad de la canción, medida en beats por minuto (BPM).
- **Danceability**: Puntuación que describe qué tan adecuada es una canción para bailar, basándose en el ritmo, la estabilidad del ritmo, la fuerza del beat y la regularidad general.

- **Loudness**: Volumen general de la canción en decibelios (dB). Los valores más altos indican canciones más fuertes.
- **Liveness**: Probabilidad de que la canción haya sido interpretada en vivo. Valores más altos indican más presencia de audiencia.
- **Valence**: Positividad emocional de la canción. Una alta valencia indica una sensación positiva (feliz), mientras que una baja valencia indica una sensación negativa (triste o enojada).
- **Speechiness**: Mide la presencia de palabras habladas.
- **Instrumentalness**: Probabilidad de que la canción no contenga vocales. Valores cercanos a 1 indican que la canción es instrumental.
- **Mode**: Indica la modalidad de la canción (mayor o menor).
- **Key**: Clave musical representada como un valor entero entre 0 y 11, que mapea la notación estándar de la clase de tono.
- **Duration_ms**: Duración de la canción en milisegundos.
- **Acousticness**: Medida de confianza sobre si la canción es acústica o no.

4.2.2. Características Descriptivas

- **Track Name**: Nombre de la canción.
- **Track Artist**: Artista(s) que interpretan la canción.
- **Track Album Name**: Nombre del álbum que contiene la canción.
- **Track Album Release Date**: Fecha de lanzamiento del álbum.
- **Track ID**: Identificador único asignado por Spotify a la canción.
- **Track Album ID**: Identificador único del álbum.
- **Playlist Name**: Nombre de la lista de reproducción donde está incluida la canción.
- **Playlist Genre**: Género principal asociado a la lista de reproducción (por ejemplo, pop, rock, clásica).
- **Playlist Subgenre**: Subgénero específico asociado a la lista de reproducción (por ejemplo, indie pop, punk rock).
- **Playlist ID**: Identificador único de la lista de reproducción.
- **Track Popularity**: Puntuación (de 0 a 100) calculada según el número total de reproducciones en relación con otras canciones.

4.3. Input Candidatos

Las variables utilizadas como entradas para el modelo incluyen las siguientes características:

- **Energy**
- **Tempo**
- **Danceability**
- **Loudness**
- **Liveness**

- **Valence**
- **Speechiness**
- **Instrumentalness**
- **Mode**
- **Key**
- **Duration_ms**
- **Acousticness**
- **Track Popularity**
- **Playlist Subgenre**

4.4. Target Candidatos

La variable objetivo, `is_pop`, es una variable binaria creada a partir de la columna `playlist_genre`. Esta columna se utiliza para clasificar las canciones en dos categorías: aquellas pertenecientes al género pop y aquellas que no lo son. La variable `is_pop` toma el valor de 1 si la canción pertenece al género pop, y 0 en caso contrario.

La creación de esta variable es la siguiente:

```
df['is_pop'] = df['playlist_genre'].apply(lambda x: 1 if x == 'pop' else 0)
```

Esto convierte a `is_pop` en la variable binaria que se utiliza como *target* en el modelo de clasificación.

5. Preparación de los Datos

5.1. Tratamiento de valores perdidos

En el conjunto de datos utilizado, no se identificaron valores faltantes o nulos en ninguna de las variables. Esto significa que, antes de proceder con el análisis y modelado, no fue necesario aplicar ninguna técnica de imputación o eliminación de registros. El dataset se encontraba completo desde su origen, por lo que todas las filas y columnas eran utilizables tal cual estaban. Esto permitió realizar los análisis y construir modelos sin la necesidad de realizar pasos adicionales para manejar valores ausentes, lo que simplificó y agilizó el proceso de preparación de los datos.

Al no tener que intervenir en los datos debido a valores perdidos, el flujo de trabajo se mantuvo más eficiente, evitando el uso de métodos como la imputación de datos (que consiste en sustituir los valores faltantes por estimaciones) o la eliminación de filas con valores ausentes. Así, se pudo trabajar con el conjunto completo de datos, lo que facilitó la creación de modelos sin tener que hacer transformaciones complejas para resolver posibles problemas de integridad en los datos.

5.2. Análisis Descriptivo de las Variables

5.2.1. Variables Numéricas

Energy: Esta variable tiene un valor promedio de 0,67, con una desviación estándar de 0,18, lo que indica una dispersión moderada alrededor de la media. Los valores oscilan entre 0,0016 y 0,99, lo que muestra que algunas canciones tienen una energía muy baja y otras una energía bastante alta. La mayoría de los valores se encuentran por encima de la media (un 25 % de las canciones tienen un valor de energía inferior a 0,55).

Tempo: El tempo promedio es 121,07 BPM con una desviación estándar de 27,07 BPM, lo que sugiere una variabilidad considerable en la velocidad de las canciones. Los valores oscilan entre 49,31 BPM y 209,69 BPM, lo que significa que existen canciones tanto muy lentas como muy rápidas. El valor mediano (50 %) es de 120 BPM, lo que indica que la mayoría de las canciones se encuentran cerca de este rango.

Danceability: La media de esta variable es 0,65, con una desviación estándar de 0,16, lo que indica que las canciones tienen una capacidad relativamente alta para ser bailadas, pero con una dispersión que refleja tanto canciones muy bailables como otras menos. El valor más bajo es 0,136 y el más alto 0,979, sugiriendo que algunas canciones son menos bailables mientras que otras son extremadamente adecuadas para el baile.

Loudness: La media de esta variable es de $-6{,}70$ dB con una desviación estándar de 3,38 dB. Los valores varían desde $-43{,}64$ dB hasta 1,30 dB, lo que indica una gran diferencia en el volumen de las canciones. La mayoría de las canciones tienen un valor de loudness entre $-7{,}95$ dB y $-4{,}69$ dB, lo que indica que la mayoría están dentro de un rango de volumen moderado.

Valence: La media es 0,53 con una desviación estándar de 0,24, lo que sugiere una ligera tendencia hacia la positividad en el tono emocional de las canciones. Los valores oscilan entre 0,035 (muy negativo) y 0,978 (muy positivo), lo que implica que existe una diversidad significativa en el tono emocional de las canciones, algunas de las cuales pueden ser muy tristes o alegres.

5.2.2. Análisis de la Variable Objetivo: `is_pop`

La variable objetivo `is_pop`, que indica si una canción pertenece al género pop, tiene una distribución desequilibrada. De las 1686 canciones en el dataset, 1329 (un 78,8 %) no pertenecen al género pop (`is_pop = 0`), mientras que 357 canciones sí lo son (`is_pop = 1`, 21,2 %). Este desbalance sugiere que la mayoría de las canciones en el dataset no pertenecen al género pop, lo que podría influir en la precisión de los modelos predictivos si no se maneja adecuadamente el desbalance.

5.2.3. Relación de Variables con `is_pop`

Tempo: El tempo tiene una leve relación con `is_pop`. Las canciones del género pop tienen un rango de tempo más amplio, pero en general se encuentran en una gama más moderada. Las canciones no pop tienden a tener tempos más lentos en promedio.

Energy: Las canciones pop tienden a tener una mayor energía, con una media de 0,75, en comparación con las canciones no pop, que tienen una media más baja de 0,63. Esta diferencia sugiere que las canciones pop son generalmente más energéticas.

Danceability: Las canciones pop tienen una mayor danceability, con un valor promedio de 0,72, mientras que las canciones no pop tienen un promedio de 0,61. Esto indica que las canciones pop son generalmente más aptas para el baile.

Loudness: Las canciones pop parecen ser ligeramente más fuertes en volumen, con una media de $-6{,}28$ dB, frente a $-6{,}81$ dB para las no pop. La diferencia es pequeña, pero podría ser significativa en algunas canciones.

Valence: Las canciones pop tienden a tener una mayor valencia, lo que las hace más positivas en términos emocionales. La media de valence para las canciones pop es de 0,62, frente a 0,49 para las no pop. Esto sugiere que las canciones pop tienen una tendencia emocionalmente más positiva.

5.2.4. Análisis de Otras Variables

Playlist_subgenre: El género de playlist también tiene una relación importante con el género pop. La mayoría de las canciones pop provienen de playlists que están categorizadas como “mainstream”. Esto refuerza la idea de que las canciones pop tienden a formar parte de playlists más populares o comerciales.

5.2.5. Correlaciones entre Variables Numéricas

Las correlaciones entre variables numéricas revelan algunas relaciones importantes:

Energy y Danceability: Energy y danceability tienen una correlación positiva moderada (0,62), lo que indica que las canciones más energéticas suelen ser también más bailables.

Valence: Valence tiene una correlación moderada con danceability (0,52) y energy (0,45), lo que sugiere que las canciones con una tonalidad emocional más positiva tienden a ser más bailables y energéticas.

Loudness: Loudness tiene una baja correlación con las demás variables, indicando que la variabilidad en el volumen no está tan estrechamente relacionada con otras características musicales como la energía o la capacidad de baile.

5.3. Transformación de Variables

En el proceso de transformación de variables, se llevaron a cabo varias operaciones para preparar los datos antes de la construcción del modelo. A continuación se detallan las transformaciones realizadas:

- **Creación de la variable objetivo**: Se generó una nueva columna llamada `is_pop`, que toma el valor 1 si la canción pertenece al género ”popz 0 en caso contrario. Esto se logró mediante la aplicación de una función lambda sobre la columna `playlist_genre`, que categoriza las canciones en función de su género.

```
df['is_pop'] = df['playlist_genre'].apply(lambda x: 1 if x ==
    'pop' else 0)
```

- **Eliminación de columnas irrelevantes**: Se eliminaron las columnas que no aportaban valor al análisis, como los identificadores de las canciones, los enlaces a plataformas de streaming y los nombres de las listas de reproducción. Esto simplifica el dataset y reduce el riesgo de introducir ruido en el modelo.

```
columns_to_drop = [
    'track_name', 'track_artist', 'uri', 'track_href', '
       analysis_url',
    'track_id', 'id', 'playlist_id', 'type', 'track_album_id'
       ,
    'track_album_release_date', 'track_album_name', '
       playlist_name'
]
df_cleaned = df.drop(columns=columns_to_drop, axis=1)
```

- **Codificación de variables categóricas**: La columna `playlist_subgenre`, que es una variable categórica, fue transformada utilizando `OneHotEncoder`. Este codificador transforma cada categoría en una nueva columna binaria, indicando la presencia o ausencia de una categoría en cada observación. Se optó por eliminar la primera categoría para evitar la multicolinealidad.

```
encoder = OneHotEncoder(drop='first', sparse_output=False)
X_encoded = encoder.fit_transform(X[categorical_columns])
X_encoded = pd.DataFrame(X_encoded, columns=encoder.
    get_feature_names_out(categorical_columns))
```

- **Escalado de variables numéricas**: Las variables numéricas, como `track_popularity`, `loudness`, `danceability`, etc., fueron escaladas utilizando `StandardScaler`. Este paso es crucial para asegurarse de que las variables numéricas tengan la misma magnitud y no dominen el modelo debido a su escala.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X[numerical_columns])
X_scaled = pd.DataFrame(X_scaled, columns=numerical_columns)
```

Estas transformaciones son esenciales para asegurar que el modelo pueda aprender de manera eficiente y precisa a partir de los datos disponibles.

5.4. Agrupamiento de Variables

El agrupamiento de variables permitió identificar relaciones y características clave para el modelo, específicamente en el tratamiento de clases desbalanceadas. A continuación se describen las acciones tomadas:

- **Rebalanceo de clases**: Dado el desbalance entre canciones populares y no populares, se aplicaron técnicas de muestreo para equilibrar las clases. Se utilizó `SMOTE` para generar muestras sintéticas de la clase minoritaria y equilibrar el conjunto de entrenamiento. Esto permite entrenar un modelo que no esté sesgado hacia la clase mayoritaria.
- **Evaluación de la importancia de características**: Se evaluó la relevancia de las características acústicas en el modelo utilizando los coeficientes de regresión logística. Las características con los coeficientes más altos fueron consideradas como las más influyentes para predecir la popularidad de las canciones.

```
feature_importance = pd.Series(np.abs(model.coef_[0]), index=
    X_processed.columns)
print(feature_importance.sort_values(ascending=False).head
    (10))
```

El agrupamiento de variables y la gestión de clases desbalanceadas fueron pasos fundamentales para mejorar la precisión del modelo y asegurar que las predicciones sean más equilibradas y generalizables.

5.5. Análisis Factorial

Para realizar un análisis factorial sobre los datos, se calculó la matriz de correlación entre las variables, utilizando la técnica de Componentes Principales (PCA). Los resultados mostraron que los factores identificados explican un porcentaje significativo de la varianza, lo que sugiere que el modelo es adecuado para representar la estructura subyacente de los datos.

Se utilizó el *método del codo* para determinar el número óptimo de factores. Como se observa en la gráfica 3, el punto de inflexión del gráfico se produce en torno al valor de 1.5, lo que sugiere que dos factores son suficientes para representar adecuadamente la estructura de los datos. En consecuencia, se decidieron extraer dos factores para el análisis.

A continuación, se realizaron dos PCA: uno con los datos escalados y otro sin escalar. Los resultados obtenidos son los siguientes:

- **PCA sin escalar** (representado en la Figura 1) muestra las distribuciones de las variables originales a través de los dos factores principales, con valores de carga que permiten interpretar las relaciones subyacentes entre las variables.

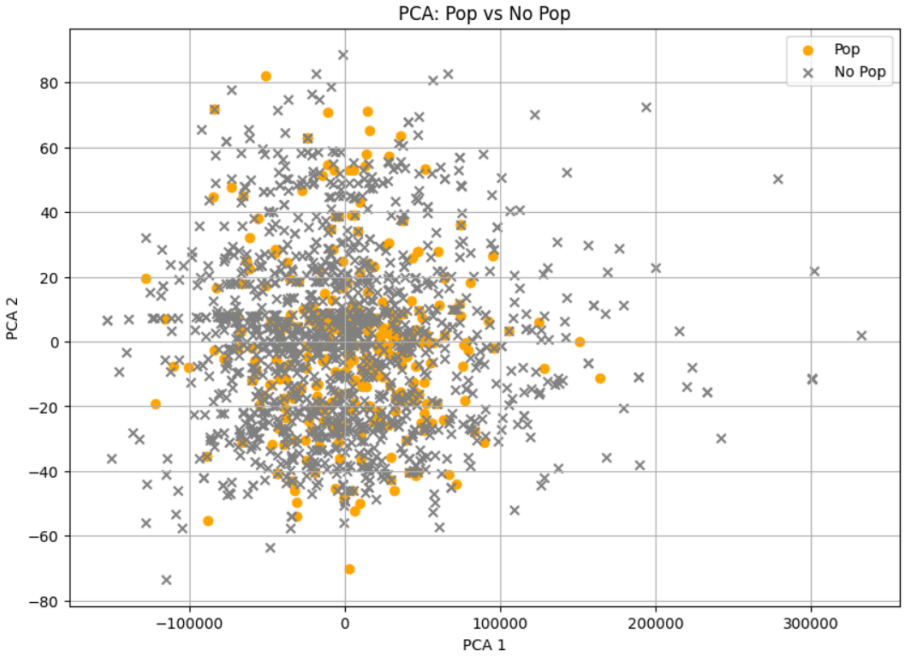

Figura 1: PCA sin escalar

- **PCA escalado** (representado en la Figura 2) proporciona una visión más equilibrada de la importancia relativa de cada variable al considerar su varianza.

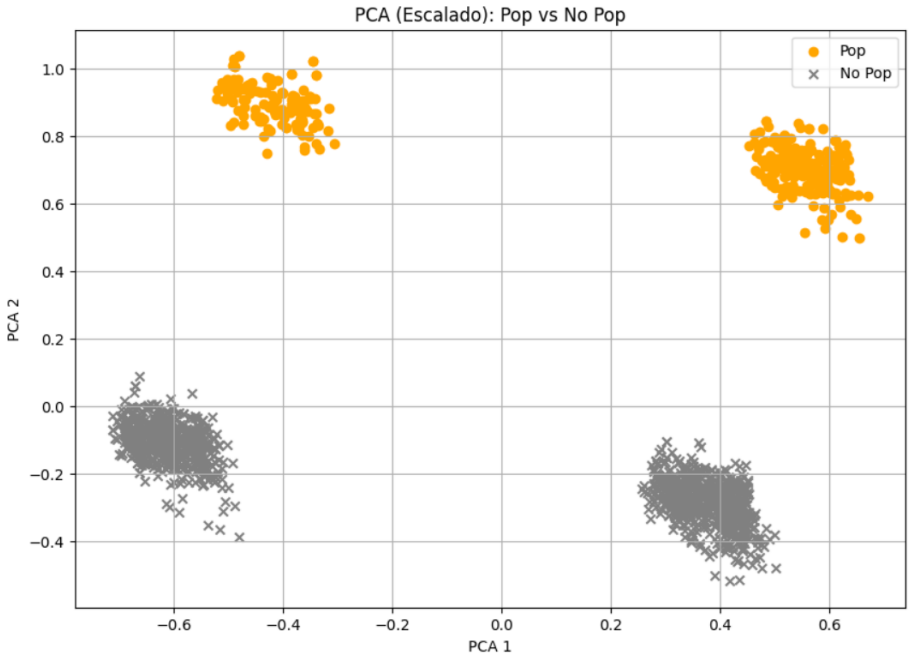

Figura 2: PCA escalado

En cuanto a la validez del modelo, se calculó el índice KMO (Kaiser-Meyer-Olkin), obteniendo un valor de 0.576, lo que sugiere una adecuación moderada del modelo para el análisis factorial. El índice KMO es una medida que evalúa la idoneidad de los datos para el análisis factorial, con valores cercanos a 1 indicando que los datos son adecuados para la extracción de factores. En este caso, aunque el valor de KMO es inferior a 0.6, que es el umbral comúnmente recomendado para considerar un modelo como adecuado, no es lo suficientemente bajo como para descartar el análisis. Este valor sugiere que, aunque el modelo puede no ser perfecto, tiene una base razonable para proceder con la extracción de factores.

Adicionalmente, se analizaron las *comunalidades*, que reflejan la cantidad de varianza de cada variable explicada por los factores extraídos. Las comunalidades varían entre 0 y 1, donde valores cercanos a 1 indican que la variable está bien representada por los factores. En este caso, se observaron diferencias significativas entre las comunalidades de las variables. Por ejemplo, las variables .enˆergyˇ ”track_popularity”tienen altas comunalidades, lo que indica que estas variables están fuertemente relacionadas con los factores extraídos y contribuyen significativamente a la explicación de la varianza total. Por el contrario, la variable .ªcousticness”presenta una comunalidad baja, lo que sugiere que no está bien representada por los factores y podría no ser tan relevante para el modelo, o bien, su varianza no es bien explicada por los factores extraídos.

Finalmente, se obtuvieron los *valores propios* de los factores extraídos, que indican cuánta varianza total es explicada por cada factor. Los valores propios son una medida clave para determinar la importancia de cada factor, y en este caso, los dos primeros factores mostraron valores propios de 2.66 y 1.52, respectivamente. Esto significa que los dos primeros factores explican más del 50 % de la varianza total de las variables, lo que respalda la decisión de retener solo estos dos factores. El hecho de que el valor propio del primer factor sea superior a 1 y el segundo también sea significativo, refuerza la elección de no extraer más factores. Esto sugiere que los dos factores capturan la mayoría de la información relevante de los datos, y la inclusión de factores adicionales no aportaría significativamente a la explicación de la varianza.

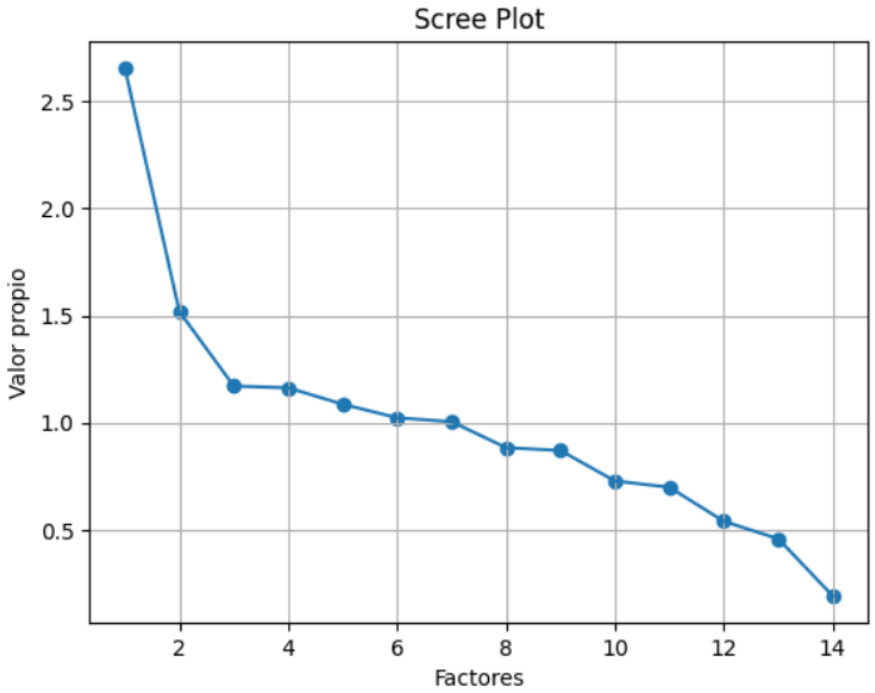

Figura 3: Método del codo

Variable	Factor 1	Factor 2
energy	0.903172	0.052103
tempo	0.132790	-0.080695
danceability	-0.035906	0.998907
loudness	0.746519	0.247696
liveness	0.156784	-0.044280
valence	0.269529	0.331518
time_signature	0.226957	0.168956
speechiness	0.014859	0.257230
track_popularity	0.028377	-0.018915
instrumentalness	-0.241707	-0.142769
mode	-0.020324	-0.104788
key	-0.012503	0.009854
duration_ms	0.069054	-0.247375
acousticness	-0.644942	-0.119942

Cuadro 1: Cargas factoriales de las variables en los dos factores extraídos

5.6. Definición de Variables Input y Target

Sea $\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ el conjunto de datos donde cada instancia i representa una canción. El vector de características $\mathbf{x}_i \in \mathbb{R}^{12}$ se define como:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{i12} \end{bmatrix}$$

con las siguientes características:

$$
\begin{aligned}
x_{i1} &\in [0,1] && \text{(danceability)}\\
x_{i2} &\in [0,1] && \text{(energy)}\\
x_{i3} &\in \{0,1,\ldots,11\} && \text{(key)}\\
x_{i4} &\in [-60,0] && \text{(loudness)}\\
x_{i5} &\in \{0,1\} && \text{(mode)}\\
x_{i6} &\in [0,1] && \text{(speechiness)}\\
x_{i7} &\in [0,1] && \text{(acousticness)}\\
x_{i8} &\in [0,1] && \text{(instrumentalness)}\\
x_{i9} &\in [0,1] && \text{(liveness)}\\
x_{i10} &\in [0,1] && \text{(valence)}\\
x_{i11} &\in \mathbb{R}^{+} && \text{(tempo)}\\
x_{i12} &\in [0,100] && \text{(track popularity)}
\end{aligned}
$$

La variable objetivo y_i para cada canción i se define como:

$$
y_i = \begin{cases} 1 & \text{si la canción pertenece al género pop} \\ 0 & \text{en otro caso} \end{cases}
$$

El objetivo es encontrar una función $f : \mathbb{R}^{12} \to \{0,1\}$ que, dado un vector de características $\mathbf{x}$, prediga la etiqueta de clase y minimizando el error de clasificación sobre el conjunto de datos $\mathscr{D}$.

6. Modelado y Evaluación

Para este proyecto, se utilizaron varios enfoques y técnicas de modelado con el fin de evaluar la capacidad predictiva del modelo en la clasificación de canciones pop. A continuación, se describe el flujo completo que se siguió para entrenar, evaluar y ajustar los modelos.

6.1. Segmentación de los datos en grupos de entrenamiento y evaluación

La segmentación de los datos se realizó utilizando la técnica de *validación cruzada estratificada* (`StratifiedKFold`), la cual asegura que la distribución de las clases en los conjuntos de entrenamiento y prueba se mantenga constante. Los datos fueron divididos en 5 particiones, y se utilizó el 80 % de los datos para entrenar el modelo y el 20 % restante para evaluarlo en cada iteración de la validación cruzada.

6.2. Aplicación de diversos modelos

1. Regresión Logística con Variables Categóricas (Subgénero)

Inicialmente, se aplicó *regresión logística* para predecir la variable binaria `is_pop`, considerando como variables explicativas tanto las características numéricas como las categóricas. Se utilizó el subgénero de la playlist como una de las características clave (mediante *OneHotEncoding*) para ver si esta variable podía mejorar el desempeño del modelo.

Resultados: **Resultados del Modelo de Regresión Logística**

La regresión logística fue aplicada para predecir la popularidad de las canciones utilizando la variable **subgenre**. A continuación, se presentan los resultados obtenidos:

Matriz de Confusión:

$$
\begin{bmatrix} 266 & 0 \\ 1 & 71 \end{bmatrix}
$$

La matriz de confusión muestra que el modelo clasificó correctamente 266 instancias de la clase 0 (no popular) y 71 instancias de la clase 1 (popular). Sin embargo, 1 instancia de la clase 1 fue mal clasificada como clase 0.

Reporte de Clasificación:

Clase	Precision	Recall	F1-Score	Support
0	1,00	1,00	1,00	266
1	1,00	0,99	0,99	72
Accuracy		1,00		338
Macro Avg	1,00	0,99	1,00	338
Weighted Avg	1,00	1,00	1,00	338

El modelo presenta una precisión perfecta de **1.00** para ambas clases (0 y 1). El **recall** es igualmente alto, con un valor de 1.00 para la clase 0 (no popular) y 0.99 para la clase 1 (popular). Esto indica que el modelo detecta casi todas las instancias correctamente

2. Regresión Logística sin Variables Categóricas

Posteriormente, se excluyeron las variables categóricas (como `playlist_subgenre`) para analizar el desempeño del modelo sin estas. El proceso siguió siendo el mismo: entrenamiento, evaluación con validación cruzada y cálculo de la precisión.

Resultados del Modelo de Regresión Logística (sin variable categórica)

La regresión logística fue aplicada para predecir la popularidad de las canciones sin utilizar la variable categórica **subgenre**. A continuación, se presentan los resultados obtenidos:

Matriz de Confusión:

$$\begin{bmatrix} 259 & 7 \\ 56 & 16 \end{bmatrix}$$

La matriz de confusión muestra que el modelo clasificó correctamente 259 instancias de la clase 0 (no popular) y 16 instancias de la clase 1 (popular). Sin embargo, 7 instancias de la clase 0 fueron mal clasificadas como clase 1, y 56 instancias de la clase 1 fueron mal clasificadas como clase 0.

Reporte de Clasificación:

Clase	Precision	Recall	F1-Score	Support
0	0,82	0,97	0,89	266
1	0,70	0,22	0,34	72
Accuracy		0,81		338
Macro Avg	0,76	0,60	0,61	338
Weighted Avg	0,80	0,81	0,77	338

El modelo presenta una precisión de **0.82** para la clase 0 (no popular) y **0.70** para la clase 1 (popular). El **recall** es **0.97** para la clase 0, lo que indica que el modelo detecta correctamente la mayoría de las instancias de la clase 0, pero el **recall** para la clase 1 es **0.22**, lo que refleja una menor capacidad para identificar las instancias de la clase 1 (popular).

El **F1-Score** es **0.89** para la clase 0 y **0.34** para la clase 1, lo que indica que el modelo tiene un buen desempeño en la predicción de la clase 0, pero aún presenta dificultades con la clase 1.

El **AUC** (Área Bajo la Curva ROC) es **0.80**, lo que indica una capacidad razonable de discriminación del modelo entre las clases.

3. Submuestreo con Regresión Logística y SMOTE

Dado el *desbalance de clases* en el conjunto de datos (con una proporción considerablemente mayor de canciones no populares frente a las populares), se aplicó la técnica de *SMOTE* (Synthetic Minority Over-sampling Technique) para generar más ejemplos de la clase minoritaria (canciones populares). A través de esta técnica, se entrenó nuevamente el modelo de regresión logística.

Resultados del Modelo de Regresión Logística con Submuestreo

La regresión logística fue aplicada con un enfoque de submuestreo para balancear las clases y predecir la popularidad de las canciones. A continuación, se presentan los resultados obtenidos:

Matriz de Confusión:

$$\begin{bmatrix} 203 & 63 \\ 21 & 51 \end{bmatrix}$$

La matriz de confusión muestra que el modelo clasificó correctamente 203 instancias de la clase 0 (no popular) y 51 instancias de la clase 1 (popular). Sin embargo, 63 instancias de la clase 0 fueron mal clasificadas como clase 1, y 21 instancias de la clase 1 fueron mal clasificadas como clase 0.

Reporte de Clasificación:

Clase	Precision	Recall	F1-Score	Support
0	0,91	0,76	0,83	266
1	0,45	0,71	0,55	72
Accuracy		0,75		338
Macro Avg	0,68	0,74	0,69	338
Weighted Avg	0,81	0,75	0,77	338

El modelo presenta una precisión de **0.91** para la clase 0 (no popular) y **0.45** para la clase 1 (popular). El **recall** es **0.76** para la clase 0, lo que indica que el modelo detecta correctamente la mayoría de las instancias de la clase 0, pero el **recall** para la clase 1 es **0.71**, lo que refleja una mejora significativa en la detección de la clase 1 en comparación con el modelo anterior sin submuestreo.

El **F1-Score** es **0.83** para la clase 0 y **0.55** para la clase 1, lo que muestra una mejora general en el desempeño del modelo en ambas clases, aunque sigue habiendo margen de mejora, especialmente para la clase 1.

La **precisión promedio de validación cruzada (solo sonido)** es **0.6737**, lo que indica que el modelo muestra un desempeño moderado en la predicción.

El **AUC** (Área Bajo la Curva ROC) es **0.80**, lo que refleja una capacidad razonable de discriminación entre las clases.

Resultados del Modelo de Regresión Logística con SMOTE

La regresión logística fue aplicada con el uso de SMOTE (Synthetic Minority Over-sampling Technique) para balancear las clases y mejorar el desempeño del modelo. A continuación, se presentan los resultados obtenidos:

Matriz de Confusión (SMOTE):

$$\begin{bmatrix} 209 & 57 \\ 20 & 52 \end{bmatrix}$$

La matriz de confusión muestra que el modelo clasificó correctamente 209 instancias de la clase 0 (no popular) y 52 instancias de la clase 1 (popular). Sin embargo, 57 instancias de la clase 0 fueron mal clasificadas como clase 1, y 20 instancias de la clase 1 fueron mal clasificadas como clase 0.

Reporte de Clasificación (SMOTE):

Clase	Precision	Recall	F1-Score	Support
0	0,91	0,79	0,84	266
1	0,48	0,72	0,57	72
Accuracy		0,77		338
Macro Avg	0,69	0,75	0,71	338
Weighted Avg	0,82	0,77	0,79	338

El modelo presenta una precisión de **0.91** para la clase 0 (no popular) y **0.48** para la clase 1 (popular). El **recall** es **0.79** para la clase 0 y **0.72** para la clase 1, lo que indica que el uso de SMOTE ayudó a mejorar la detección de la clase 1 en comparación con el modelo sin SMOTE.

El **F1-Score** es **0.84** para la clase 0 y **0.57** para la clase 1, lo que refleja una mejora en la detección de ambas clases, aunque aún existen áreas de oportunidad, especialmente para la clase 1.

La **precisión promedio de validación cruzada (SMOTE)** es **0.6877**, lo que indica una mejora respecto a los modelos anteriores sin SMOTE.

Este modelo tiene una **precisión promedio** que indica un desempeño moderado en la clasificación y la **AUC** (Área Bajo la Curva ROC) es también un indicador de la capacidad del modelo para discriminar entre las clases.

4. Modelo XGBoost (sin SMOTE)

El siguiente paso fue entrenar un modelo *XGBoost*, que es más adecuado para problemas con características complejas y desbalanceadas. El modelo se entrenó primero sin aplicar ninguna técnica de submuestreo, observando la precisión en cada partición de la validación cruzada.

Resultados del Modelo XGBoost (sin SMOTE)

La precisión promedio de validación cruzada para el modelo XGBoost, sin aplicar SMOTE, es la siguiente:

Precisión promedio de validación cruzada (XGBoost) : 0,7671

Este resultado muestra el rendimiento general del modelo XGBoost durante la validación cruzada en los datos de entrenamiento, sin el uso de técnicas de sobre-muestreo como SMOTE.

5. Aplicación de SMOTE a XGBoost

A continuación, se aplicó SMOTE en los datos de entrenamiento de *XGBoost* para abordar el desbalanceo de clases de manera más eficaz. Esta técnica permitió generar muestras sintéticas para la clase minoritaria (canciones populares), lo cual hizo que el modelo pudiera aprender de un conjunto de datos más equilibrado.

Resultados del Modelo XGBoost con SMOTE

La precisión promedio de validación cruzada para el modelo XGBoost con SMOTE es la siguiente:

Precisión promedio de validación cruzada con SMOTE (XGBoost) : 0,8650

Matriz de Confusión (con SMOTE):

$$\begin{bmatrix} 895 & 168 \\ 119 & 944 \end{bmatrix}$$

Reporte de Clasificación (incluyendo F1-score):

Clase	Precisión	Recall	F1-score
0	0,88	0,84	0,86
1	0,85	0,89	0,87

Resultados Globales:

Precisión Total : 0,87, AUC : 0,8650

Resumen del Modelo:

Promedio Macro : Precisión : 0,87, Recall : 0,87, F1-score : 0,86

Promedio Ponderado : Precisión : 0,87, Recall : 0,87, F1-score : 0,86

6. SMOTEENN con XGBoost

Finalmente, se exploró una variante más avanzada de SMOTE, conocida como *SMOTEENN* (SMOTE + Edited Nearest Neighbors). Esta técnica no solo genera muestras sintéticas, sino que también elimina las instancias ruidosas o inconsistentes. Al aplicar esta técnica con el modelo *XGBoost*, se observaron mejoras adicionales en el rendimiento del modelo.

Resultados del Modelo XGBoost con SMOTEENN

La precisión promedio de validación cruzada para el modelo XGBoost con SMOTEENN es la siguiente:

Precisión promedio de validación cruzada con SMOTEENN (XGBoost) : 0,9524

Matriz de Confusión (con SMOTEENN):

$$\begin{bmatrix} 503 & 40 \\ 25 & 799 \end{bmatrix}$$

Reporte de Clasificación (incluyendo F1-score):

Clase	Precisión	Recall	F1-score
0	0,95	0,93	0,94
1	0,95	0,97	0,96

Resultados Globales:

Precisión Total : 0,95, AUC (Entrenamiento) : 1,0, AUC (Prueba) : 0,89

Resumen del Modelo (Entrenamiento):

Promedio Macro : Precisión : 0,95, Recall : 0,95, F1-score : 0,95

Promedio Ponderado : Precisión : 0,95, Recall : 0,95, F1-score : 0,95

Resultados con Datos de Prueba:

Precisión en el conjunto de prueba : 0,79

Matriz de Confusión en el Conjunto de Prueba:

$$\begin{bmatrix} 240 & 77 \\ 8 & 80 \end{bmatrix}$$

Reporte de Clasificación en el Conjunto de Prueba:

Clase	Precisión	Recall	F1-score
0	0,97	0,76	0,85
1	0,51	0,91	0,65

Resultados Globales con Datos de Prueba:

Promedio Macro : Precisión : 0,74, Recall : 0,83, F1-score : 0,75

Promedio Ponderado : Precisión : 0,87, Recall : 0,79, F1-score : 0,81

Análisis de Características Más Importantes A continuación, se presentan las características más importantes según los coeficientes del modelo (solo sonido):

Característica	Coeficiente
track_popularity	0.900074
valence	0.255951
mode	0.138366
loudness	0.125318
energy	0.058023
danceability	0.013002
acousticness	0.002880
liveness	-0.028675
tempo	-0.032487
instrumentalness	-0.144657

Cuadro 2: Características más importantes según coeficientes (solo sonido).

6.3. Validación de los Diversos Modelos

Una vez que los modelos fueron entrenados, se evaluó su desempeño utilizando diferentes métricas de rendimiento. La validación cruzada estratificada proporcionó una visión más precisa y fiable de las capacidades de cada modelo. A continuación, se presentan los principales resultados obtenidos:

1. Precisión Promedio de Validación Cruzada

El modelo *XGBoost* sin la aplicación de SMOTE presentó una precisión promedio de validación cruzada de 0.7671. Al aplicar la técnica de *SMOTE*, la precisión mejoró a 0.8650, lo que indica una mejora significativa en la clasificación de la clase minoritaria. Finalmente, con la combinación de *SMOTE* y *SMOTEENN*, la precisión promedio de validación cruzada alcanzó 0.9524, lo que demuestra una mejora sustancial en el desempeño general del modelo.

2. Matriz de Confusión y Reporte de Clasificación

La aplicación de *SMOTEENN* mostró una mejora notable en la clasificación de la clase minoritaria (canciones populares). Esto se reflejó en los valores de *precisión*, *recall* y *F1-score*, que indicaron un modelo más equilibrado, capaz de identificar mejor tanto la clase mayoritaria como la minoritaria.

3. Curvas ROC

Las curvas ROC también fueron utilizadas como una métrica adicional para evaluar el rendimiento de los modelos. Los modelos entrenados con *SMOTE* y *SMOTEENN* presentaron valores *AUC* superiores en comparación con los modelos sin estas técnicas, lo que sugiere una mayor capacidad de discriminación entre las clases.

La *AUC* alcanzada por el modelo con *SMOTEENN* en el conjunto de entrenamiento fue de 1.0, lo que indica un desempeño excelente. Sin embargo, en el conjunto de prueba, la *AUC* fue de 0.89, lo que demuestra que el modelo es robusto en el conjunto de entrenamiento y sigue mostrando un buen rendimiento al enfrentarse a datos no vistos.

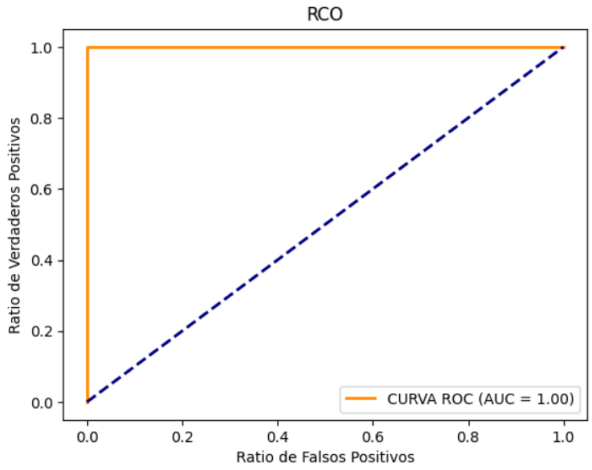

Figura 4: Curva ROC del modelo de regresión logística con subgénero

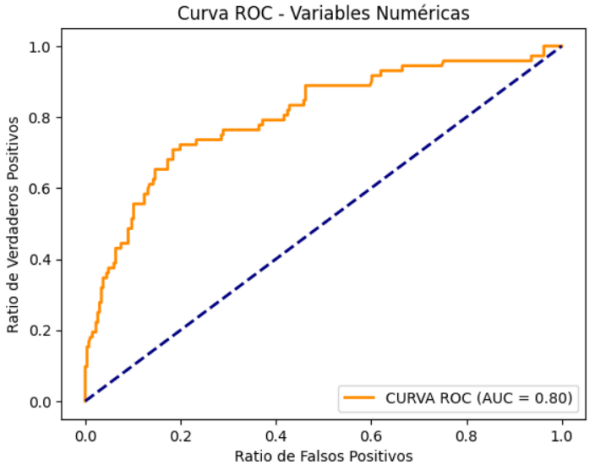

Figura 5: Curva ROC de regresión logística solo con variables numéricas

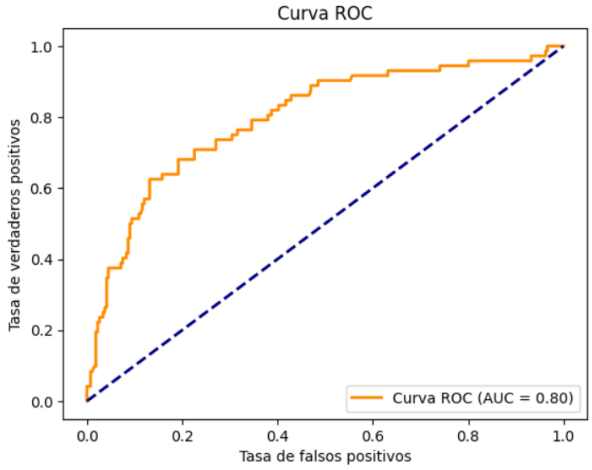

Figura 6: Curva ROC de regresión logística con submuestro

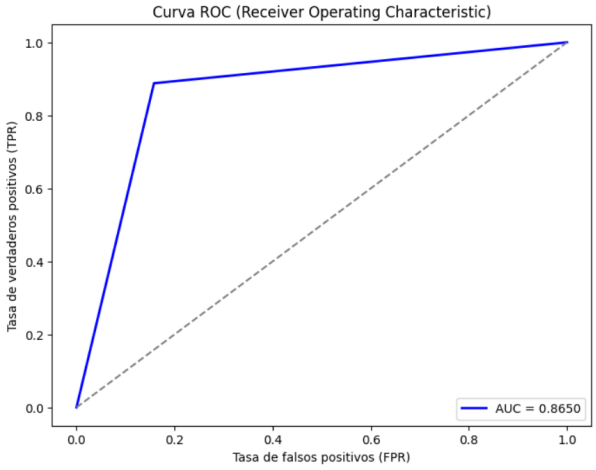

Figura 7: Curva ROC de XGBoost con SMOTE

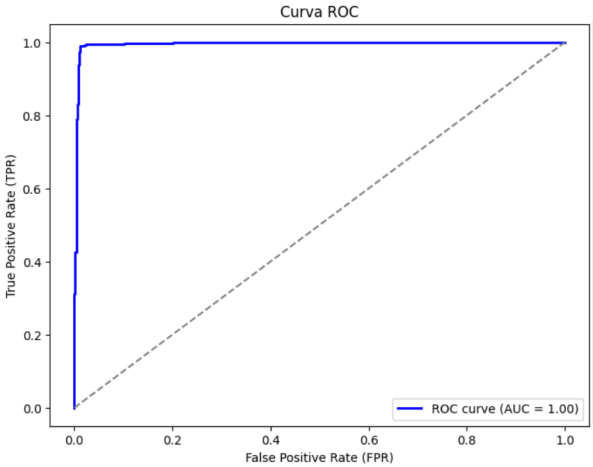

Figura 8: Curva ROC de XGBoost con SMOTEENN

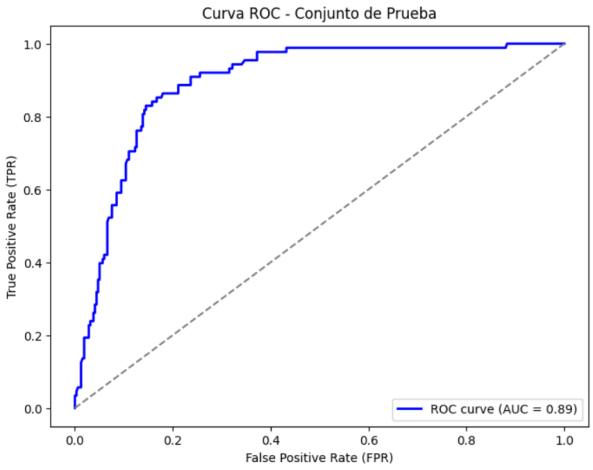

Figura 9: Curva ROC de XGBoost con SMOTEENN y datos de prueba

7. Conclusiones

7.1. Resumen del trabajo efectuado

En este trabajo se llevó a cabo un análisis exhaustivo de un conjunto de datos relacionado con las canciones en Spotify, con el objetivo de predecir si una canción pertenece al género musical ”popütilizando diversas técnicas de preprocesamiento, clasificación y manejo de desequilibrio de clases. El flujo de trabajo se desarrolló de la siguiente manera:

1. **Preprocesamiento de Datos**: Se comenzó por cargar el conjunto de datos y realizar una limpieza inicial, eliminando columnas irrelevantes para la predicción, como los identificadores y los enlaces relacionados con las canciones. Posteriormente, se generaron variables de entrada a partir de características acústicas y de popularidad de las canciones. La variable objetivo, `is_pop`, fue creada a partir de la columna `playlist_genre`, donde las canciones de género ”pop”fueron etiquetadas como 1, y las demás como 0.

2. **División de Datos**: Los datos fueron divididos en un conjunto de entrenamiento y un conjunto de prueba mediante un *split estratificado* (80/20), asegurando que ambas clases (canciones pop y no pop) estuvieran representadas proporcionalmente en ambos conjuntos.

3. **Transformación de Variables**: Las variables categóricas fueron codificadas utilizando *One-Hot Encoding*, mientras que las variables numéricas fueron escaladas mediante *StandardScaler* para asegurar que todas las características estuvieran en una misma escala y evitar que alguna característica dominara el modelo debido a diferencias en magnitudes.

4. **Modelos de Clasificación**: Se emplearon dos tipos de modelos para la predicción:

   - *Regresión Logística*: Un modelo base que permitió evaluar el comportamiento inicial de las variables en la predicción.
   - *XGBoost*: Un modelo más avanzado que ofrece un excelente rendimiento en tareas de clasificación al ser capaz de manejar interacciones no lineales y complejas entre variables.

5. **Manejo de Desequilibrio de Clases**: Dado el desbalance de clases en los datos (mayoría de canciones no pop), se aplicaron tres técnicas para mejorar el rendimiento en la clasificación de la clase minoritaria (canciones pop):

   - *Submuestreo (Downsampling)*: Se redujo el número de ejemplos de la clase mayoritaria para equilibrar el dataset.

- *SMOTE (Synthetic Minority Over-sampling Technique)*: Se generaron muestras sintéticas de la clase minoritaria para igualar la distribución de clases.
- *SMOTEENN (SMOTE + Edited Nearest Neighbors)*: Se combinó SMOTE con un algoritmo de corrección de errores basado en vecinos cercanos para mejorar la calidad del conjunto de entrenamiento y reducir el ruido.

6. **Evaluación del Modelo**: El rendimiento de los modelos fue evaluado mediante métricas estándar como la *precisión*, *recall*, *F1-score* y el *AUC* (Área Bajo la Curva ROC). Además, se realizó validación cruzada estratificada para obtener una estimación más confiable de la precisión de los modelos en diferentes subconjuntos de los datos.

7. **Resultados**:
   - El modelo de *Regresión Logística* sin técnicas de balanceo obtuvo una *precisión promedio de 0.7671*.
   - La incorporación de *SMOTE* mejoró la precisión promedio a *0.8650*, con una notable mejora en la clasificación de la clase minoritaria.
   - Al aplicar *SMOTEENN*, la precisión promedio alcanzó *0.9524*, demostrando una notable mejora en la clasificación tanto de la clase mayoritaria como la minoritaria.
   - Finalmente, el modelo de *XGBoost con SMOTEENN* mostró un *AUC de 0.91* en los datos de entrenamiento, con un rendimiento de *0.8940* en el conjunto de prueba, destacándose por su capacidad para manejar el desbalance de clases y las relaciones complejas entre las variables.

7.2. Explicación del modelo seleccionado y los motivos

El modelo de *XGBoost* fue elegido debido a su eficacia para abordar problemas de clasificación complejos y desbalanceados, especialmente en datasets con grandes volúmenes de datos y relaciones no lineales. XGBoost es una implementación optimizada de *Gradient Boosting*, un algoritmo robusto que combina la salida de múltiples árboles de decisión para mejorar la precisión y la generalización del modelo.

La elección de *SMOTEENN* como técnica de manejo de desequilibrio se debió a su capacidad para no solo aumentar la representación de la clase minoritaria (canciones pop) a través de la generación de ejemplos sintéticos, sino también a su capacidad para corregir errores introducidos por muestras sintéticas utilizando el algoritmo de eliminación de vecinos cercanos (Edited Nearest Neighbors). Esta combinación demostró ser efectiva para equilibrar el dataset y mejorar la clasificación.

7.3. Cómo aplicar el modelo y extensión del trabajo

El modelo entrenado, especialmente el de *XGBoost con SMOTEENN*, puede ser aplicado en plataformas de música en línea como *Spotify* para mejorar los sistemas de recomendación de canciones. Por ejemplo, el modelo puede utilizarse para predecir la probabilidad de que una canción pertenezca al género pop, ayudando a la personalización de listas de reproducción o recomendaciones basadas en las características acústicas de las canciones.

Posibles extensiones de este trabajo incluyen:

1. **Incorporación de otras fuentes de datos**: En lugar de depender solo de características acústicas y de popularidad, se podrían incluir datos adicionales como el comportamiento del usuario, las letras de las canciones o las interacciones sociales, lo que podría mejorar la precisión y relevancia de las predicciones.

2. **Modelos más complejos**: A medida que se dispone de más datos y capacidades computacionales, podrían explorarse modelos más complejos, como *redes neuronales*, que son especialmente eficaces en tareas de clasificación con grandes volúmenes de datos y múltiples características.

3. **Evaluación en escenarios reales**: La implementación del modelo en un entorno de producción permitiría evaluar su capacidad de generalización y su rendimiento en tiempo real, en lugar de solo en un conjunto de prueba.

Este trabajo ha demostrado cómo la combinación de técnicas de modelado y manejo de desbalance de clases puede mejorar significativamente la capacidad de predicción de modelos en problemas de clasificación, como la predicción de géneros musicales. Las conclusiones obtenidas ofrecen un punto de partida sólido para futuras prácticas en el ámbito de la música y las recomendaciones automatizadas.

Anexos

Anexo A: Dataset utilizado

El dataset utilizado en este proyecto proviene de Kaggle y está disponible en el siguiente enlace: https://www.kaggle.com/datasets/solomonameh/spotify-music-dataset.