**Elizabeth Lorena Porras Ortiz**
**Jorge Alberto Gómez Vigoya**
**Carol Johana Peña Pico**
**Juan Sebastian Pascuas**

---

# Proyecto 1: Predicción de popularidad de canciones

## 1. Preprocesamiento de Datos

El proyecto tiene como objetivo crear un modelo que realice la predicción de la popularidad de una canción considerando un set de variables que incluye el nombre del artista, álbum, género y demás datos relacionados a la instrumentalidad. Dentro de los análisis descriptivos se encontró que las variables categóricas contenían un gran número de clases por lo cual las sintetizamos en un set más pequeño. A continuación se relacionan las variables creadas:

- Género musical: Se agrupan dentro de 11 clases que ahora incluyen rock, electrónica, pop, hip-hop, clásica, látina, folklore, indie, temáticos, niños y otros.
- Nombre de la canción: La variable contiene textos de los cuales es difícil extraer un provecho significativo. Por ello, se genera un código que realiza un conteo de las palabras más repetidas, con el objetivo de identificar términos frecuentes que permitan crear categorías basadas en las palabras más utilizadas. Cabe aclarar que no se tendrán en cuenta palabras conectoras o aquellas que no se consideren relevantes para el ejercicio, ya que no aportan valor a la clasificación del nombre de la canción.
- Álbum: Se genera el mismo tratamiento de la variable ‘canción’
- Nombre del artista: Se tomó como base el conteo de apariciones de los artistas. Aquellos con un conteo superior a 100 se ubicaron en la categoría Top 20; si el conteo se encuentra entre 80 y menos de 100, se clasificaron como Top 40, y así sucesivamente. Para este ejercicio se consideró que el conteo máximo de apariciones es 139 y el mínimo es 1, por lo que se definieron los rangos en función de estos valores. Posteriormente, estos rangos se aplicaron a la variable original, generando categorías que representan los artistas más escuchados.

En cuanto a la distribución de la variable objetivo se observa que tiene un rango de 0 a 100 con una alta concentración en los valores más pequeños, también hay presencia de outliers pero como se desarrollaran modelos basados en árboles de decisión, no los vamos a eliminar ni a imputar pues son datos que no afectan a los modelos de este tipo.

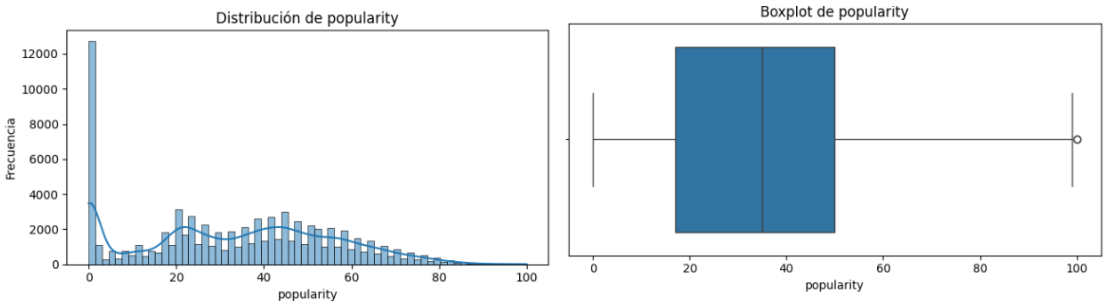



**Figura 1.** Distribución de la variable objetivo.

### 1.1. Análisis de Correlaciones

Inicialmente, se realiza un análisis de correlación entre todas las variables numéricas presentes en el conjunto de datos de entrenamiento. El objetivo principal de este paso es identificar variables altamente correlacionadas entre sí, las cuales pueden aportar información redundante que afecta negativamente la estabilidad y generalización de los modelos predictivos.

Primero, se contaba con 21 variables numéricas originales antes de la creación de variables dummies. Posteriormente, al transformar las variables categóricas en variables dummies y ampliar el conjunto de predictores, el número de variables aumentó. Se calcula la matriz de correlación y se eliminan aquellas variables cuya correlación absoluta es mayor al 80%. Así, se conservan únicamente las variables que aportan información relevante y no redundante.

Resultados tras creación de dummies:

- Variables iniciales (tras creación de dummies): 63
- Variables eliminadas por alta correlación: 13
- Variables finales tras eliminación: 50

En la matriz de correlaciones se observó que algunas variables como energy y loudness presentaban altos niveles de correlación, lo cual justificó su eliminación para evitar redundancia en el modelo.

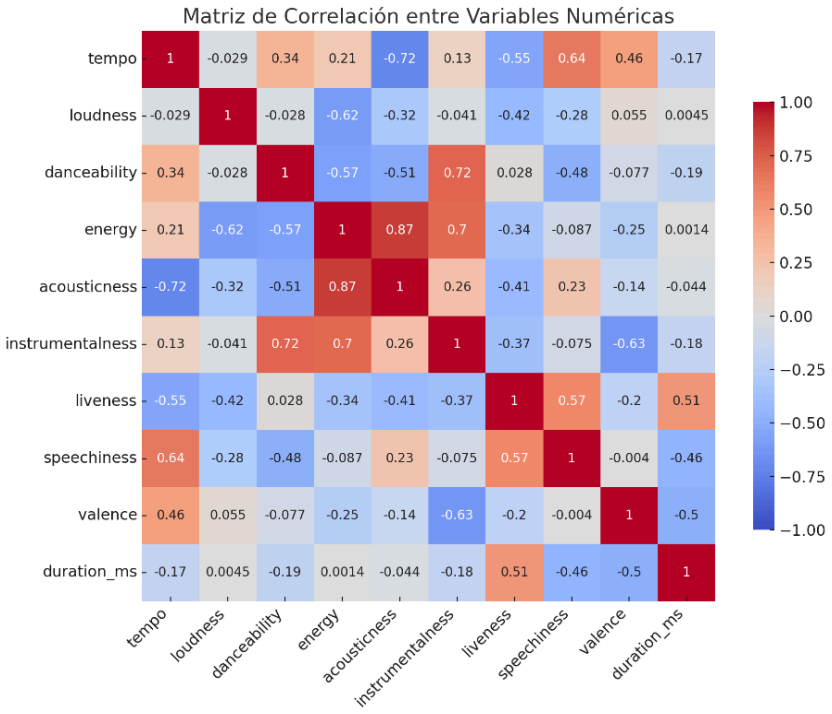



**Figura 2.** Matriz de correlaciones entre las variables numéricas del conjunto de datos.

### 1.2. VIF

Realizando la prueba VIF se identificaron 5 variables con un factor alto, que fueron eliminadas para evitar la distorsión de las estimaciones de los coeficientes, la inflación de errores estándar y la afectación de la interpretación del modelo.

### 1.3. Selección de variables

Se aplicó RFE (Eliminación Recursiva de Características) para seleccionar automáticamente las variables más relevantes. La selección se hizo con validación cruzada, ajustando modelos de regresión.

Considerando que el proceso es costoso en términos computacionales, se tomó una muestra aleatoria de la base (30%) para evaluar de forma más eficiente las mejores variables. Se emplearon modelos como Gradient Boosting, XGBoost y Random Forest por su alta capacidad predictiva, manejo de datos complejos y su robustez frente al sobreajuste, además de facilitar la interpretación de la importancia de las variables. Una vez ejecutado, tomamos las variables señaladas por los 3 modelos, dejando la selección final en 40 predictores.

**1.4. División de Datos Entrenamiento y Validación:** Para evaluar de manera más precisa el rendimiento de los modelos, se realiza una partición del conjunto de datos original:

- 80% de los datos se utilizan para el entrenamiento del modelo (X_train_final, y_train_final): 79,580 observaciones.

- 20% de los datos se reservan para una validación interna antes de aplicar el modelo al conjunto de test definitivo.  (X_val, y_val): 19,895 observaciones.

La división se realiza de forma aleatoria, asegurando una representación adecuada del conjunto de datos.

---

**2. Calibración del Modelo**

Vamos a tomar la misma muestra del dataframe original para calibrar los hiperparametros de cada modelo, se tendrán en cuenta sólo las variables indicadas en el punto anterior.

**2.1 Modelos calibrados**

Se calibraron los siguientes modelos:

- Gradient Boosting Regressor
- XGBoost Regressor
- Random Forest Regressor

**2.2 Método de calibración**

Se utilizó Optuna, un optimizador automático de hiper parámetros basado en búsqueda bayesiana.

Justificación del método seleccionado:

- Eficiencia: Optuna requiere menos tiempo de búsqueda comparado con métodos tradicionales como GridSearchCV o RandomizedSearchCV.
- Búsqueda adaptativa: Optuna utiliza estrategias de exploración-explotación inteligentes, lo que permite encontrar combinaciones óptimas de hiper parámetros de manera dinámica.
- Optimización automática: Optuna optimiza directamente la métrica de interés (en este caso, el Error Cuadrático Medio - MSE) sin necesidad de definir manualmente el espacio de búsqueda completo.
- Flexibilidad: Permite controlar de manera más precisa el proceso de calibración, ajustar funciones objetivo personalizadas y utilizar paralelización si es necesario.

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=30)
```

**2.3 Parámetros calibrados**

Los parámetros a calibrar son los siguientes:

- n_estimators: Define el número de árboles a usar en el modelo; más árboles mejoran la capacidad predictiva pero aumentan el costo computacional. Es importante encontrar un valor que no vaya a generar sobreajuste.
- learning_rate: Se trata de la velocidad a la que el modelo aprende, tasas más bajas permiten un aprendizaje más preciso, pero se debe cuidar la convergencia.
- max_depth: Controla la profundidad de cada árbol. Tener una buena calibración evita el sobreajuste.
- subsample: Define la cantidad de datos usada en cada iteración, es bueno calibrarlo para tener diversidad en los árboles.
- min_samples_split: Define el número de muestras necesarias para dividir un nodo. Tener el parámetro correcto limita las divisiones innecesarias.
- min_samples_leaf: Define el número mínimo de muestras necesarias en cada hoja. Este parámetro es útil para evitar hojas demasiado pequeñas que capturen ruido en lugar de patrones significativos.
- min_child_weight: Evita que el modelo se enfoque demasiado en datos que son muy pequeños o específicos, reduciendo errores innecesarios.
- colsample_bytree: Especifica la fracción de características a considerar para construir cada árbol. Calibrarlo mejora la capacidad del modelo de manejar datos de alta dimensionalidad.
- gamma: Evita que el modelo haga divisiones a menos que sean realmente útiles.
- reg_alpha: Introduce penalizaciones para pesos grandes de las características.
- max_features: Determina cuántas variables puede considerar el modelo, tener el número adecuado ayuda a reducir el riesgo de sobreajuste.
- bootstrap: Indica si el modelo va a tomar muestras al azar con reemplazo para crear los árboles. En caso de hacerlo, los árboles serán más diversos y robustos.

### 2.4. Parámetros óptimos:

- Gradient Boosting:

```
Mejores hiperparámetros encontrados:
{'n_estimators': 400, 'learning_rate': 0.1, 'max_depth': 6, 'subsample': 0.8, 'min_samples_split': 2, 'min_samples_leaf': 1}
```

- XGBoost:

```
Mejores hiperparámetros encontrados:
{'n_estimators': 400, 'learning_rate': 0.2, 'max_depth': 6, 'subsample': 1.0, 'min_child_weight': 1, 'colsample_bytree': 1.0, 'gamma': 1, 'reg_alpha': 1}
```

- Random Forest

```
Mejores hiperparámetros encontrados:
{'n_estimators': 300, 'max_depth': None, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': None, 'bootstrap': True}
```

---

## 3. Entrenamiento del Modelo

El entrenamiento de los modelos seleccionados se realizó utilizando los conjuntos de datos preprocesados y los hiper parámetros calibrados mediante Optuna. El objetivo fue construir modelos de regresión capaces de predecir la popularidad de canciones con alta precisión.

### 3.1. Evaluación del desempeño de los modelos individuales

- Gradient Boosting

```
Resultados en conjunto de validación del modelo Gradient Boosting:
RMSE: 16.5880
R²: 0.4460
```

- XGBoost

```
Resultados en conjunto de validación del modelo XGB:
RMSE: 16.1908
R²: 0.4722
```

- Random Forest

```
Resultados en conjunto de validación del modelo Random Forest:
RMSE: 15.1642
R²: 0.5370
```

En la evaluación de los modelos se evidencia que el desempeño es muy similar en los tres. En promedio, se están equivocando en 16 puntos al predecir la popularidad de una canción; en cuanto al $R^2$ también tienen comportamientos similares logrando explicar alrededor del 50% de la variación. En vista de que no hay ningún modelo que sobresalga, se intentarán técnicas de ensamblaje para evaluar si hay una mejora en las métricas de desempeño.

### 3.2. Construcción de ensambles

- Ensamble promedio: Se promediaron las predicciones de los tres modelos base (Gradient Boosting, XGBoost y Random Forest) para obtener una predicción final.

```
Resultados en conjunto de validación del modelo Ensamble (promedio)
RMSE: 15.5606
R²: 0.5125
```

- Ensamble ponderado: Se combinaron las predicciones asignando mayor peso a los modelos con mejor desempeño individual en RMSE, mayor peso para Random Forest y XGBoost:

```
Resultados en validación del ensamble ponderado:
RMSE: 15.1181
R²: 0.5398
```

### 3.3. Comparación

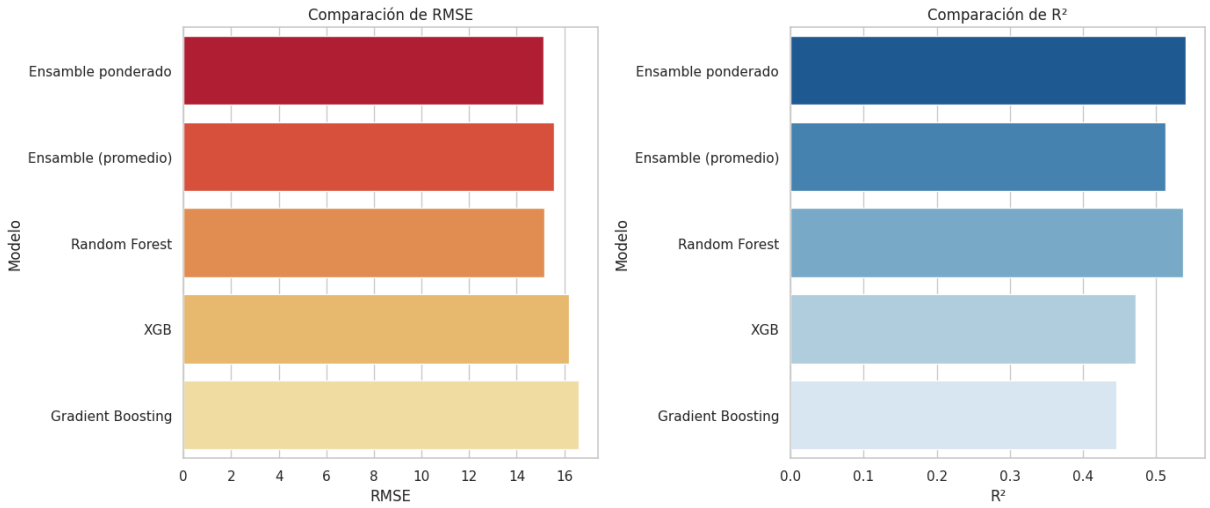



**Figura 3.** Comparación de los valores de (RMSE) y $R^2$ obtenidos por los diferentes modelos evaluados.

Teniendo la comparación del desempeño de los modelos entrenados, sigue sin haber uno significamente sobresaliente. Aún así, el que mejores métricas ofrece es el modelo ensamblado por promedio ponderado con un RMSE de 15.11 y un $R^2$ de 53%. Esto implica que, en promedio, las predicciones del modelo seleccionado tendrán una desviación de aproximadamente 15 puntos, considerando que logra explicar el 53% de la variabilidad en la popularidad de una canción.

---

## 4. Disponibilización del modelo

### 4.1. API

Para disponibilizar el modelo en una API se utiliza el servicio de AWS mediante una instancia de EC2. Considerando que el dataset original contiene un alto número de variables y observaciones simplificamos el modelo para que pudiera ser ejecutado sin problema en el servidor. De esta forma, entrenamos un modelo que predijera la popularidad de una canción utilizando dos variables y un árbol cuya máxima profundidad es 3. Se encuentra disponibilizado en la siguiente dirección:

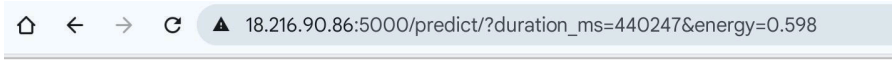
**18.216.90.86:5000**

### 4.2. Predicciones

Utilizando los datos del set de validaciones encontramos los siguientes predicciones para la popularidad de una canción:

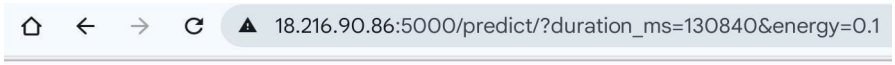
- track_id = 6KwkVtXm8OUp2XffN5k7lY

- duración(ms) = 440247
- energy = 0.598




{"result":35.6461534270459}

- track_id = 5lqigS1reMNzZDpLGSRsOH
  - duración(ms) = 130840
  - energy = 0.1




{"result":29.1673046251994}

---

**5. Conclusiones**

A lo largo del proyecto, se llevó a cabo un preprocesamiento de datos que incluyó la selección de variables mediante técnicas como correlación, VIF y RFE, así como la calibración de hiper parámetros con Optuna para optimizar el desempeño de los modelos. Se entrenaron diversos algoritmos basados en árboles de decisión, incluyendo Random Forest, XGBoost y Gradient Boosting, utilizando las variables más relevantes y los hiperparámetros que mejor se ajustabam al conjunto de datos. Como resultado, las métricas evaluadas en el set de test indicaron un desempeño aceptable de los modelos de manera individual.

La implementación de un ensamble ponderado permitió aprovechar las fortalezas de cada modelo, logrando una ligera  mejora en el rendimiento global. No obstante, aún es posible optimizar el modelo incluyendo nuevas variables que expliquen mejor la popularidad de las canciones, ya que persiste un margen de mejora en el $R^2$. Asimismo, la calibración de parámetros podría beneficiarse del uso de una mayor cantidad de datos, lo cual en este ejercicio fue una limitante, al solo poder incluir el 30%. Para estudios futuros, se sugiere integrar nuevas fuentes de información, aplicar transformaciones sobre los datos o explorar modelos alternativos como NLP, que podrían ayudar a definir la popularidad de las canciones a partir de análisis de texto.