

Thaïs Rocafull  
github: Thaisbcn/

# Proyecto Bootcam

IT Academy (Barcelona Activa)

Análisis y Predicción  
de la popularidad de una canción

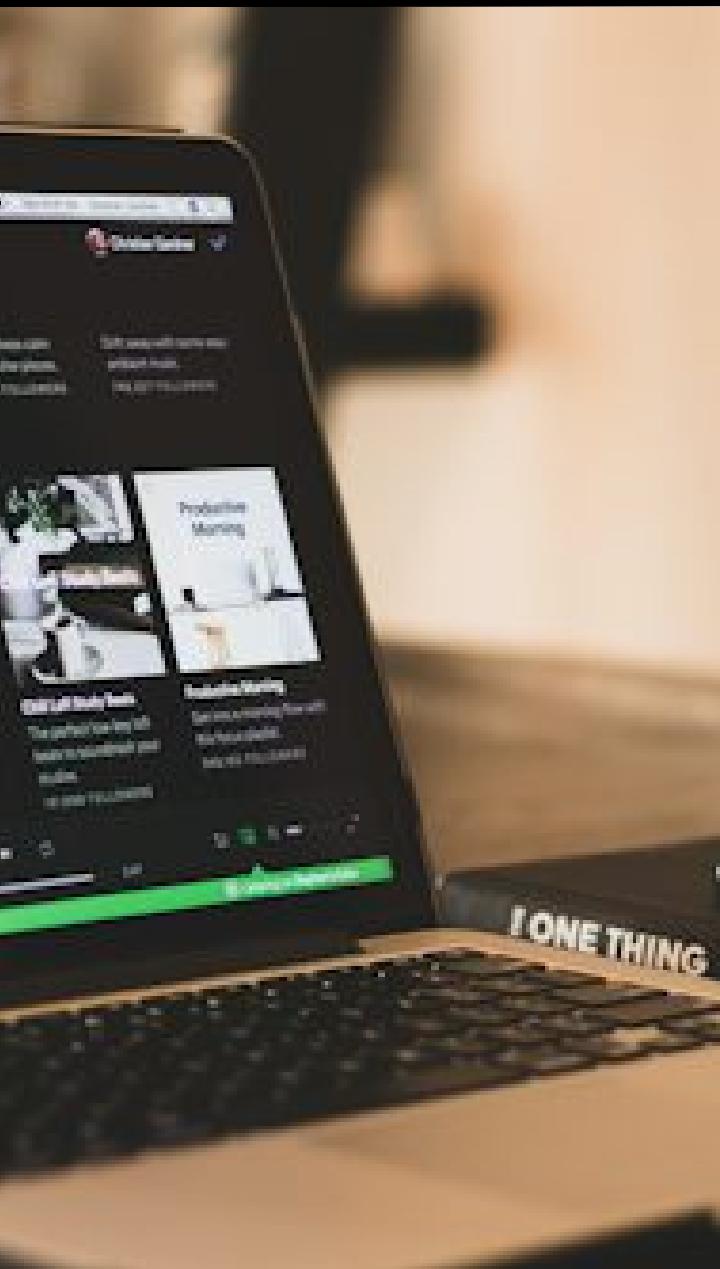
# ÍNDICE

---

1. Objetivos del proyecto
2. Conjunto de datos
3. Limpieza de datos
4. EDA
5. Preprocesado
6. Modelado
7. Mejoras y Predicción
8. Conclusión

## 1. Objetivos

# Objetivo principal



“Construir y evaluar modelos de aprendizaje automático para aplicaciones específicas, en la industria de la música y el entretenimiento”

# 1. Objetivos

## Objetivos específicos



### Limpieza de datos

- Detección de Nulls, Nans y duplicados.
- Eliminar datos no fiables

### Transformaciones

- Adaptar las categorías de los datos para poder ser tratados correctamente.

### EDA

- Análisis Exploratorio de los datos

### Machine Learning

- Entrenar modelos de aprendizaje automático.
- Evaluar rendimiento de los modelos.
- Búsqueda de mejores hiperparámetros.
- Predicción con el mejor modelo.

### Conclusiones

- Obtención de Conclusiones sobre los modelos

## 2. Conjunto de datos

# Presentación Datasets



Fuente: [www.kaggle.com](http://www.kaggle.com)

- Dos datasets obtenidos de la fuente “Kaggle” y que proceden de la extracción de la base de datos mediante un API de “Spotify”.
- Son dos bases de datos referentes a artistas musicales de todo el mundo y reproducciones en Estados Unidos durante el mes de diciembre del año 2023.



Dataset ‘Artist’

contiene información detallada sobre varios artistas, incluidos aspectos como género, edad, país, géneros musicales, popularidad y seguidores.



Dataset ‘Top songs’

Contiene información detallada sobre canciones, álbumes y artistas. Incluye información sobre la reproducibilidad, si la canción es explícita, la duración de la canción, etc.

## 2. Conjunto de datos

# Estructura DataSet “Artists”

---

- Número de Filas: 9488
- Número de Columnas: 9

### COLUMNAS Y SUS DESCRIPCIONES:

- Name: Nombre del artista (objeto).
- ID: Identificación del artista (objeto).
- Gender: Género del artista (objeto), encontramos 4 tipos y algunos valores faltantes.
- Age: Edad del artista (entero). Edades de 0 a 146 años, por lo que hay errores.
- Country: País del artista (objeto). De todo el mundo, aunque falta un 37% de datos.
- Genres: Géneros musicales asociados al artista (objeto), como pop, rock, latino...
- Popularity Artist: Puntuación de popularidad del artista (entero), del 1 al 99
- Followers: Número de seguidores del artista (entero).
- URI: Identificación única de recursos asociada al artista (objeto).

### OBSERVACIONES:

Algunas columnas tienen valores nulos, como "Gender" y "Country".

## 2. Conjunto de datos

# Estructura DataSet “Top Songs”

- Número de Filas: 37146
- Número de Columnas: 16

### COLUMNAS Y SUS DESCRIPCIONES:

- Album Type: Tipo de álbum (objeto). Se refiere a si es un “Single”, un álbum o una compilación.
- Artist ID: Identificación del artista (objeto).
- Artist Name: Nombre del artista (objeto).
- Artist Song Rank: Rango de la canción del artista (entero). Dentro de su mismo ranking.
- Track Name: Nombre de la canción (objeto).
- Is Playable: Indicador de si la canción es reproducible (booleano).
- Album Name: Nombre del álbum (objeto).
- Release Date: Fecha de lanzamiento del álbum (objeto).
- Total Album Tracks: Número total de pistas en el álbum (entero).
- Is Explicit: Se refiere a si incluye lenguaje fuerte, contenido sexual, violencia o temas controvertidos.(booleano).
- ISRC: Código de grabación estándar internacional (objeto).
- Song Duration: Duración de la canción en milisegundos (entero).
- Track Number: Número de la pista en el álbum (entero).
- Popularity Song: Puntuación de popularidad de la canción (entero). Del 1 al 99
- Track Id: Identificación de la pista (objeto).
- Track URI: Identificación única de recursos asociada a la pista (objeto).

OBSERVACIONES: Algunas columnas tienen duplicados

## 2. Conjunto de datos

# Visualización DF

	a.head()											
	Name	ID	Gender	Age	Country	Genres	Popularity	Followers				URI
0	Drake	3TVXtAsR1Inumwj472S9r4	male	33	CA	['canadian hip hop', 'canadian pop', 'hip hop'...]			95	83298497	spotify:artist:3TVXtAsR1Inumwj472S9r4	
1	Post Malone	246dkjvS1zLTtiykXe5h60	male	25	US	['dfw rap', 'melodic rap', 'pop', 'rap']			86	43130108	spotify:artist:246dkjvS1zLTtiykXe5h60	
2	Ed Sheeran	6eUKZXaKkcvih0Ku9w2n3V	male	29	GB	['pop', 'singer-songwriter pop', 'uk pop']			87	115998928	spotify:artist:6eUKZXaKkcvih0Ku9w2n3V	
3	J Balvin	1vyhD5VmyZ7KMfW5gqLgo5	male	35	CO	['reggaeton', 'reggaeton colombiano', 'trap la...']			83	38028010	spotify:artist:1vyhD5VmyZ7KMfW5gqLgo5	
4	Bad Bunny	4q3ewBCX7sLwd24euuV69X	male	26	PR	['reggaeton', 'trap latino', 'urbano latino']			95	77931484	spotify:artist:4q3ewBCX7sLwd24euuV69X	

	b.head(5)															
	Album Type	Artist ID	Artist Name	Artist Song Rank	Track Name	Is Playable	Album Name	Release Date	Total Album Tracks	Is Explicit	ISRC	Song Duration	Track Number	Popularity	Track Id	Track URI
0	album	3TVXtAsR1Inumwj472S9r4	Drake	1	IDGAF (feat. Yeat)	True	For All The Dogs	2023-10-06	23	True	USUG12306072	260111	7	93	2YSzYUF3jWqb9YP9VXmpjE	spotify:track:2YSzYUF3jWqb9YP9VXmpjE
1	album	3TVXtAsR1Inumwj472S9r4	Drake	2	First Person Shooter (feat. J. Cole)	True	For All The Dogs	2023-10-06	23	True	USUG12306071	247444	6	91	7aqfrAY2p9BUsiupwk3svU	spotify:track:7aqfrAY2p9BUsiupwk3svU
2	album	3TVXtAsR1Inumwj472S9r4	Drake	3	Rich Baby Daddy (feat. Sexy Red & SZA)	True	For All The Dogs	2023-10-06	23	True	USUG12306085	319191	20	89	1yeB8MUNeLo9Ek1UEpsyz6	spotify:track:1yeB8MUNeLo9Ek1UEpsyz6
3	album	3TVXtAsR1Inumwj472S9r4	Drake	4	Jimmy Cooks (feat. 21 Savage)	True	Honestly, Nevermind	2022-06-17	14	True	USUG12204897	218364	14	89	3F5CgOj3wFIRv51JsHbxhe	spotify:track:3F5CgOj3wFIRv51JsHbxhe
4	album	3TVXtAsR1Inumwj472S9r4	Drake	5	One Dance	True	Views	2016-05-06	20	False	USCM51600028	173986	12	89	1zi7xx7UVEFkmKfv06H8x0	spotify:track:1zi7xx7UVEFkmKfv06H8x0

### 3. Limpieza de Datos

# Limpieza general de Datos

```
1 b.drop_duplicates(inplace=True)
```

```
1 a['Gender'] = a['Gender'].fillna('desconocido')
2 a['Country'] = a['Country'].fillna('otros')
```

```
1 print(a['Gender'].unique()) # falta un 17%
2 print(a['Country'].unique()) # falta un 34%
```

```
['male' 'female' 'mixed' 'other' 'desconocido']
```

```
1 a.rename(columns={'Popularity': 'Popularity Artist'}, inplace=True)
```

```
1 b.rename(columns={'Popularity': 'Popularity Song'}, inplace=True)
```

- Eliminación de duplicados
- Conversión de valores Nulos

- Renombrar columnas de Popularidad de cada DataSet
- Eliminación de la Columna edad por contener demasiados datos erróneos

```
# Filtrar el DataFrame para obtener las filas con edades menores de 15 años
young_artists = a[a['Age'] < 15]
young_artists[['Name', 'Age']].T
```

	14	27	45	48	51	54	58	68	73	79	...	9464	9465	9469	9471	9472	9473	9474	9475
Name	The Chainsmokers	Imagine Dragons	Swae Lee	Major Lazer	Migos	Manuel Turizo	Anne-Marie	Clean Bandit	Offset	Darell	...	Martina Paz	Menchu Lauchengco-Yulo	英仁合唱團	Boujee	Winterkind	Bombotunes	Panama	Pablo Paz
Age	8	12	0	12	12	0	0	12	0	0	...	0	0	5	0	0	0	10	0

2 rows x 4777 columns

```
1 # Comprobación de valores nulos en df a
2 print("Valores nulos en df a:", a.isnull().sum().sum())
3 print("Valores NaN en df a:", a.isna().sum().sum())
4
5 # Comprobación de valores duplicados en df a
6 duplicados_a = a.duplicated()
7 print("Duplicados en df a:", duplicados_a.sum())
8
9 # Comprobación de valores nulos en df b
10 print("Valores nulos en df b:", b.isnull().sum().sum())
11 print("Valores NaN en df b:", b.isna().sum().sum())
12
13 # Comprobación de valores duplicados en df b
14 duplicados_b = b.duplicated()
15 print("Duplicados en df b:", duplicados_b.sum())
```

```
Valores nulos en df a: 4842
Valores NaN en df a: 4842
Duplicados en df a: 0
Valores nulos en df b: 0
Valores NaN en df b: 0
Duplicados en df b: 33109
```

### 3. Limpieza de Datos

# Limpieza detallada de Datos

- Valoración de la importancia de algunas columnas como [Is Playable] o [Is Explicit]
- Detección de Valores Atípicos como album de 165 pistas
- Conversión de categorías erróneas como gentilicio y genero musical de Pavarotti.

```
1 b['Is Playable'].value_counts()
```

```
True    4037  
Name: Is Playable, dtype: int64
```

```
1 b['Is Explicit'].value_counts()
```

```
False   2522  
True    1515  
Name: Is Explicit, dtype: int64
```

```
1 # Cambiar país a "IT" para todas las filas relacionadas con Luciano Pavarotti  
2 a.loc[a['Name'] == 'Luciano Pavarotti', 'Country'] = 'IT'  
3  
4 # Cambiar género a "Opera" en el DataFrame a  
5 a.loc[a['Name'] == 'Luciano Pavarotti', 'Genres'] = 'Opera'
```

```
1 pistas_pavarotti = b[b['Artist Name'] == 'Luciano Pavarotti']  
2 pistas_pavarotti
```

```
1 b_menor_igual_25 = b[b['Total Album Tracks'] <= 25]  
2 b_entre_25_y_53 = b[(b['Total Album Tracks'] > 25) & (b['Total Album Tracks'] <= 53)]  
3 b_entre_54_y_87 = b[(b['Total Album Tracks'] > 53) & (b['Total Album Tracks'] <= 87)]  
4 b_entre_88_y_165 = b[(b['Total Album Tracks'] > 87) & (b['Total Album Tracks'] <= 165)]  
5  
6 print(f'Filas con 25 o menos tracks: {len(b_menor_igual_25)}')  
7 print(f'Filas con entre 25 y 53 tracks: {len(b_entre_25_y_53)}')  
8 print(f'Filas con entre 54 y 87 tracks: {len(b_entre_54_y_87)}')  
9 print(f'Filas con entre 88 y 165 tracks: {len(b_entre_88_y_165)}')
```

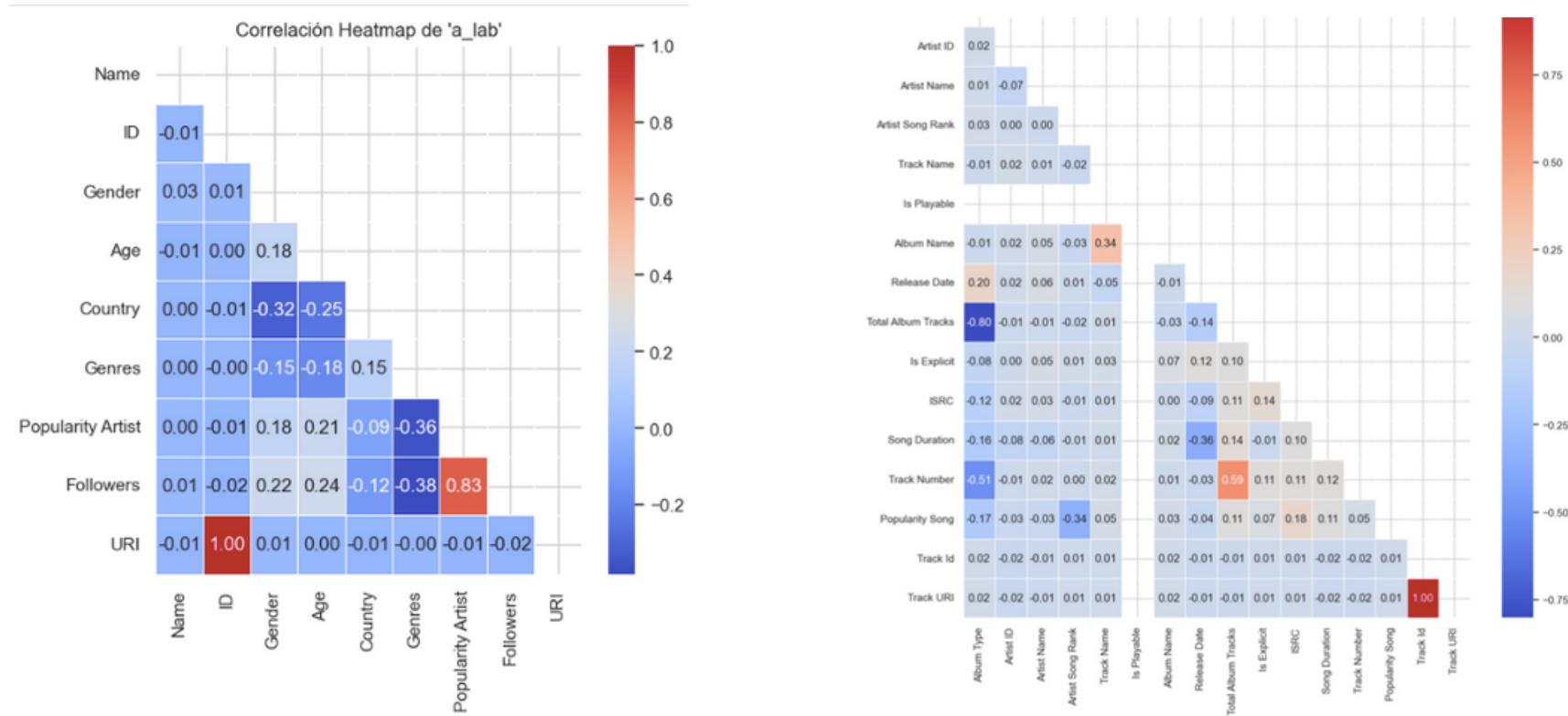
```
Filas con 25 o menos tracks: 3908  
Filas con entre 25 y 53 tracks: 124  
Filas con entre 54 y 87 tracks: 4  
Filas con entre 88 y 165 tracks: 1
```

## 3. Limpieza de Datos

# Identificar columnas iguales

## Visualización Rápida de correlaciones de los dos Datasets con LabelEncoder para localizar duplicados en los ID

- ID y URI corresponden al mismo identificador, al igual que Track ID y Trak URI
  - También podemos observar mucha correlación con Followers y Popularity Artist, al igual que Total Album tracks y Album Type, pero esta correlación por el momento no es fiable.



### 3. Limpieza de Datos

## Definir nuevo DF

Consolidamos los dos datasets con un Merge, relacionando el ID del Artista de ambos DF.

```
In [ ]: merged.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4037 entries, 0 to 4036
Data columns (total 25 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name              4037 non-null    object  
 1   ID                4037 non-null    object  
 2   Gender             4037 non-null    object  
 3   Age               4037 non-null    int64  
 4   Country            4037 non-null    object  
 5   Genres             4037 non-null    object  
 6   Popularity Artist 4037 non-null    int64  
 7   Followers          4037 non-null    int64  
 8   URI               4037 non-null    object  
 9   Album Type         4037 non-null    object  
 10  Artist ID          4037 non-null    object  
 11  Artist Name        4037 non-null    object  
 12  Artist Song Rank  4037 non-null    int64  
 13  Track Name         4037 non-null    object  
 14  Is Playable        4037 non-null    bool   
 15  Album Name          4037 non-null    object  
 16  Release Date       4037 non-null    object  
 17  Total Album Tracks 4037 non-null    int64  
 18  Is Explicit         4037 non-null    bool   
 19  ISRC               4037 non-null    object  
 20  Song Duration      4037 non-null    int64  
 21  Track Number        4037 non-null    int64  
 22  Popularity Song    4037 non-null    int64  
 23  Track Id            4037 non-null    object  
 24  Track URI           4037 non-null    object  
dtypes: bool(2), int64(8), object(15)
memory usage: 764.8+ KB
```

Definimos las Columnas prescindibles:

Eliminación de los ID, porque ya tenemos el nombre del artista y de la canción como referencia.:

- 'ID'
  - 'URI'
  - 'Artist ID'
  - 'ISRC'
  - 'Track Id'
  - 'Track URI'
- 
- 'Is Playable' (ya que todas las canciones son reproducibles)
  - 'Age' (por falta de fiabilidad)
  - 'Album Type' y 'Release Date' (información prescindible para el análisis)

## 4. EDA

# Análisis Exploratorio del Conjunto

Presentación del DF obtenido:

	Gender	Country	Genres	Popularity	Artist	Followers	Artist Name	Artist Song Rank	Track Name	Album Name	Total Album Tracks	Is Explicit	Song Duration	Popularity Song
0	male	CA	['canadian hip hop', 'canadian pop', 'hip hop'...]	95	83298497	Drake	1		IDGAF (feat. Yeat)	For All The Dogs	23	True	260111	93
1	male	CA	['canadian hip hop', 'canadian pop', 'hip hop'...]	95	83298497	Drake	2		First Person Shooter (feat. J. Cole)	For All The Dogs	23	True	247444	91
2	male	CA	['canadian hip hop', 'canadian pop', 'hip hop'...]	95	83298497	Drake	3		Rich Baby Daddy (feat. Sexyy Red & SZA)	For All The Dogs	23	True	319191	89
3	male	CA	['canadian hip hop', 'canadian pop', 'hip hop'...]	95	83298497	Drake	4		Jimmy Cooks (feat. 21 Savage)	Honestly, Nevermind	14	True	218364	89
4	male	CA	['canadian hip hop', 'canadian pop', 'hip hop'...]	95	83298497	Drake	5		One Dance	Views	20	False	173986	89

Después de completar la limpieza procedemos al EDA (Análisis Exploratorio de Datos), que encontramos en el archivo (2\_EDA).

Observación Dtype:

Hay dos columnas que deberé cambiar de formato antes de aplicar el preprocesado:

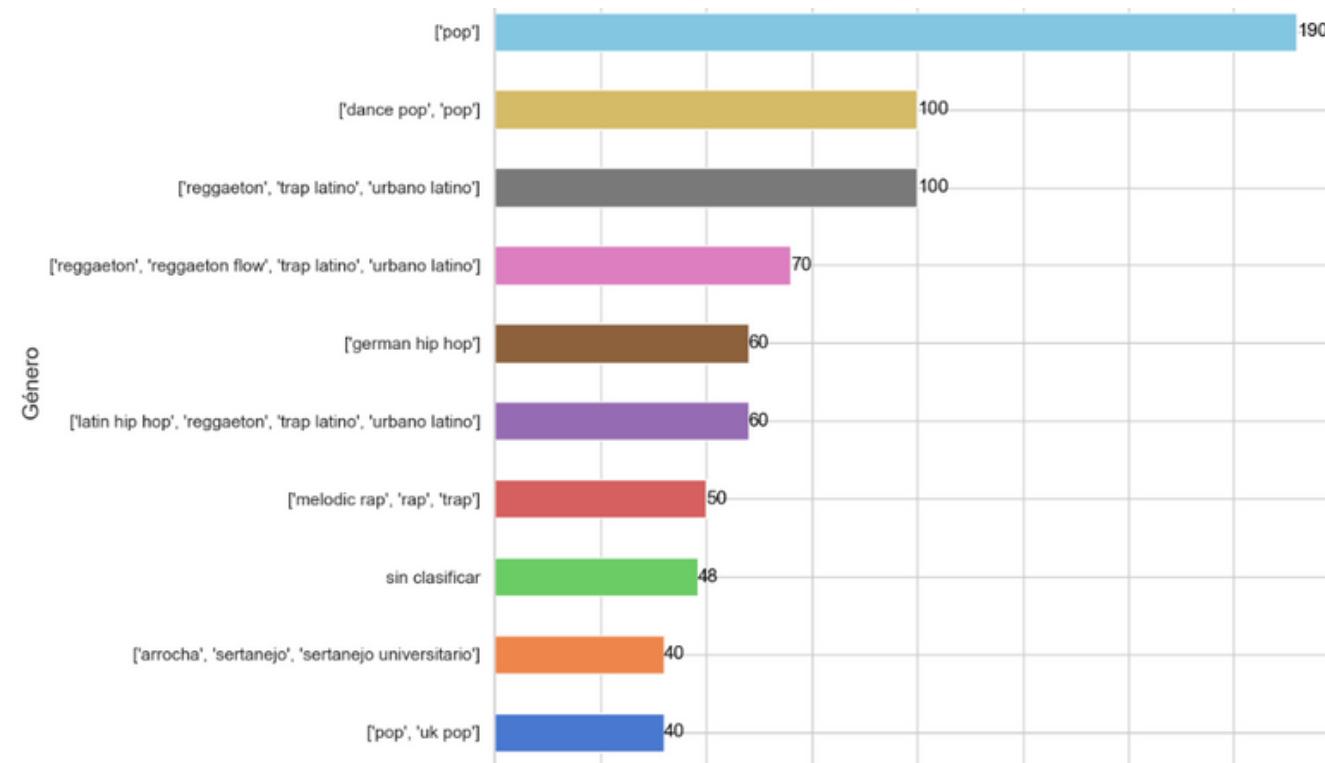
- Artist Song Rank, puesto que actualmente está en formato (int64) pero debe ser tratado como una categoría al ser ordinal. Aquí simplemente se aplicará 'Ordinal Encoder'.
- Is Explicit, porque es un booleano (true y false), que lo convertiré en categoría para usarlo como binario numérico (true=1, false= 0).

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4037 entries, 0 to 4036
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Gender          4037 non-null    object 
 1   Country         4037 non-null    object 
 2   Genres          4037 non-null    object 
 3   Popularity Artist 4037 non-null  int64  
 4   Followers       4037 non-null    int64  
 5   Artist Name     4037 non-null    object 
 6   Artist Song Rank 4037 non-null  int64  
 7   Track Name      4037 non-null    object 
 8   Album Name      4037 non-null    object 
 9   Total Album Tracks 4037 non-null  int64  
 10  Is Explicit     4037 non-null    bool   
 11  Song Duration   4037 non-null    int64  
 12  Popularity Song 4037 non-null  int64  
dtypes: bool(1), int64(6), object(6)
memory usage: 382.5+ KB
```

# Top 10 Géneros Musicales

Estas barras representan los 10 estilos musicales más escuchados en Estados Unidos,



Estos géneros necesitan una reagrupación por categoría principal, puesto que como podemos observar, en cada uno de ellos hay estilos repetidos

# Reagrupación Géneros Principales

## categorize\_genre

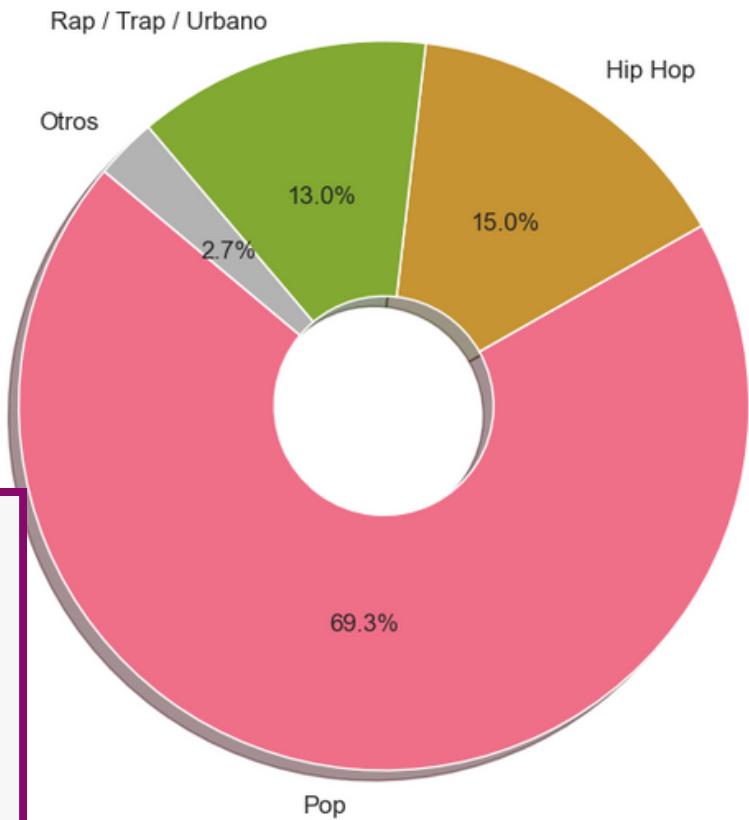
Podemos dividir la columna 'Genres' del DataFrame utilizando éste método. Divide una cadena en función de un delimitador y devuelve un DataFrame con las subcadenas resultantes.

```

1 def categorize_genre(genre):
2     genre_upper = genre.upper()
3
4     if 'POP' in genre_upper:
5         return 'Pop'
6     elif 'HIP HOP' in genre_upper:
7         return 'Hip Hop'
8     elif any(keyword in genre_upper for keyword in ['RAP', 'TRAP', 'URBANO']):
9         return 'Rap / Trap / Urbano'
10    elif any(keyword in genre_upper for keyword in ['REGGAETON', 'REGGAEON FLOW']):
11        return 'Reggaetón'
12    elif 'HOUSE' in genre_upper:
13        return 'House'
14    elif any(keyword in genre_upper for keyword in ['EDM', 'ELECTRO', 'ELECTRONIC']):
15        return 'EDM Electro'
16    elif 'ROCK' in genre_upper:
17        return 'Rock'
18
19    return 'Otros'
20
21 df['Main Genre'] = df['Genres'].apply(categorize_genre)

```

Distribución de Géneros Principales



# Reagrupación Género Secundario

## categorize\_genre

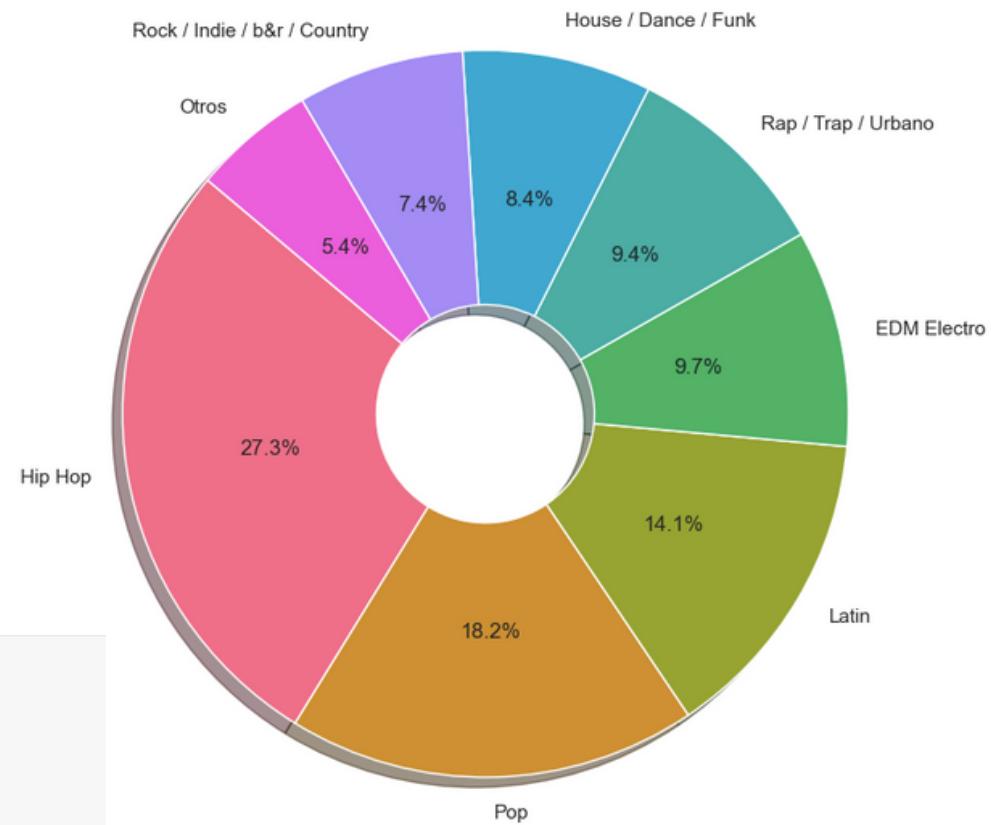
Volvemos a reagrupar los estilos, dejando el género POP en último plano, para encontrar los secundarios.

```

1 def categorize_genre(genre):
2     genre_upper = genre.upper()
3
4     if 'HIP HOP' in genre_upper:
5         return 'Hip Hop'
6     elif 'LATIN' in genre_upper:
7         return 'Latin'
8     elif any(keyword in genre_upper for keyword in ['EDM', 'ELECTRO', 'ELECTRONIC']):
9         return 'EDM Electro'
10    elif any(keyword in genre_upper for keyword in ['RAP', 'TRAP', 'URBANO','URBAN']):
11        return 'Rap / Trap / Urbano'
12    elif any(keyword in genre_upper for keyword in ['REGGAETON', 'REGGAEON FLOW']):
13        return 'Reggaetón'
14    elif any(keyword in genre_upper for keyword in ['HOUSE', 'DANCE', 'FUNK']):
15        return 'House / Dance / Funk'
16    elif any(keyword in genre_upper for keyword in ['ROCK', 'INDIE', 'B&R','SOUL','COUNTRY','b&r']):
17        return 'Rock / Indie / b&r / Country'
18    elif 'POP' in genre_upper:
19        return 'Pop'
20    return 'Otros'
21
22 df['Main Genre'] = df['Genres'].apply(categorize_genre)

```

Distribución de Categorías Principales de Géneros Musicales

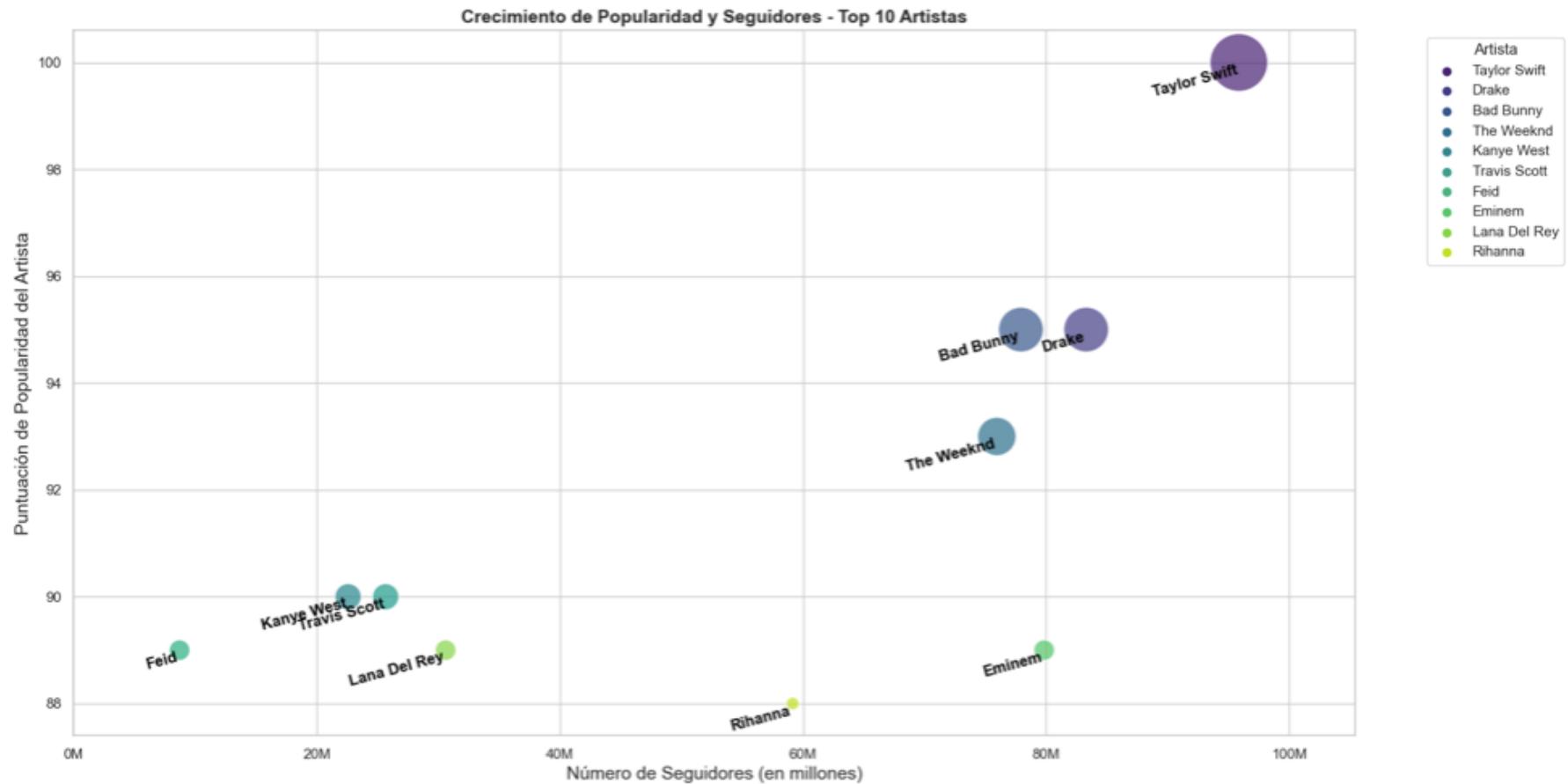


# Top 10 Países con más Artistas

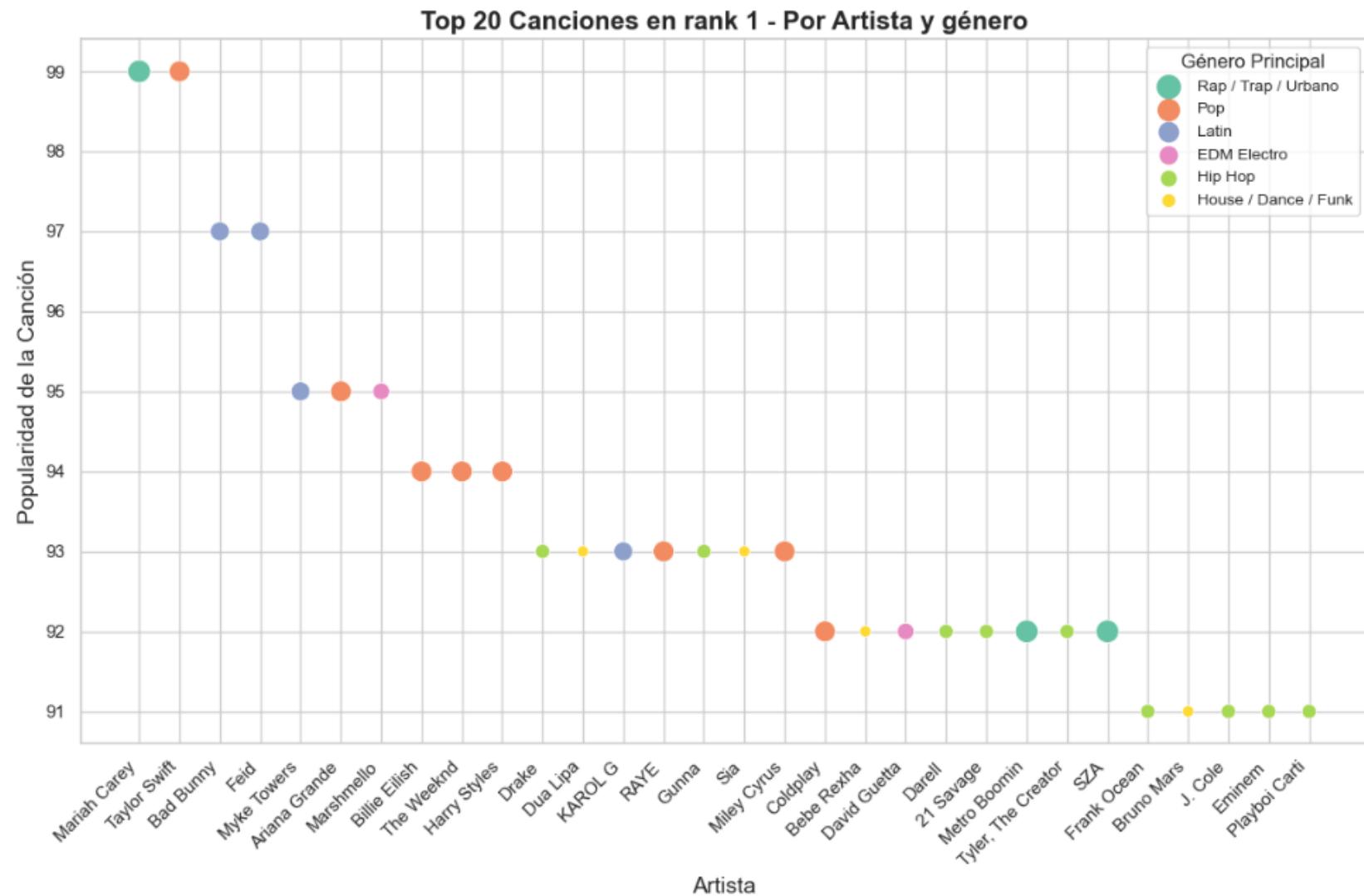
Estas barras representan la procedencia geográfica de los artistas más escuchados en Estados Unidos.



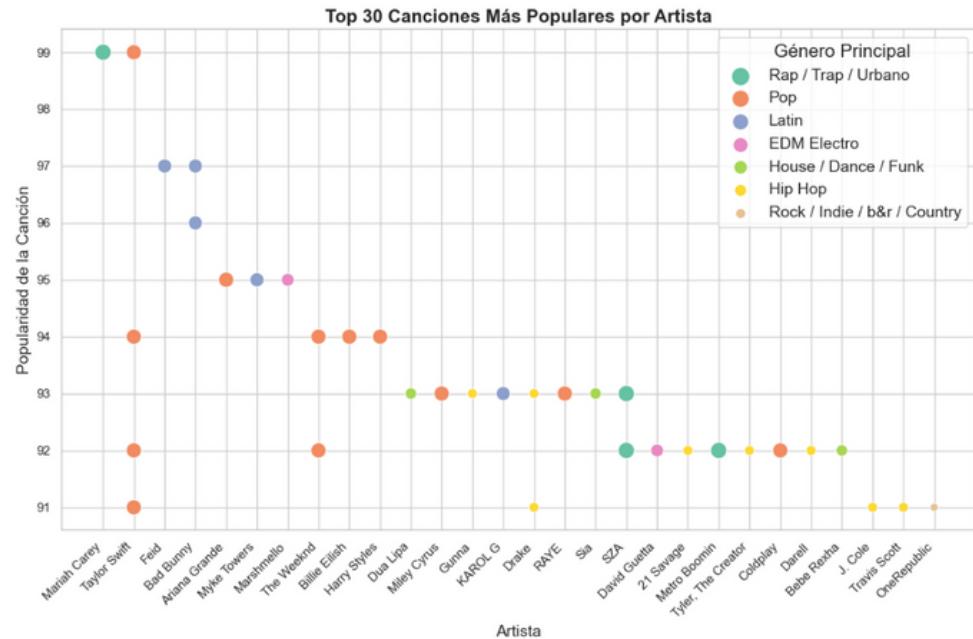
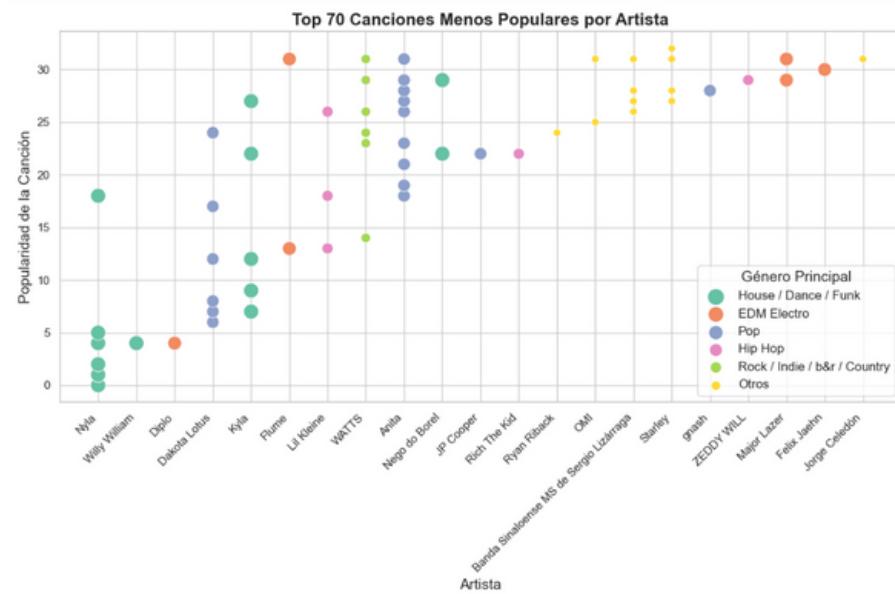
# Top 10 Artistas-Popularidad-Seguidores



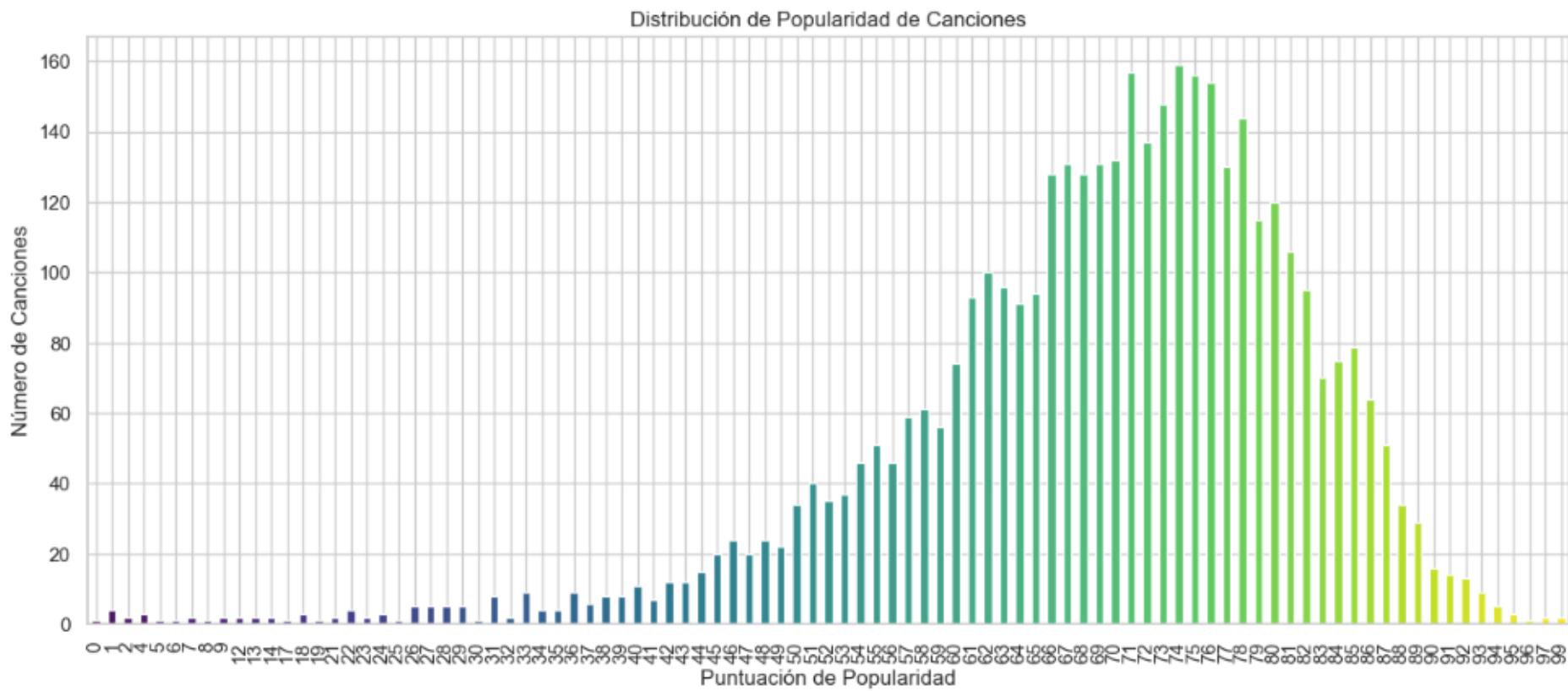
# Top Canciones Ranking 1



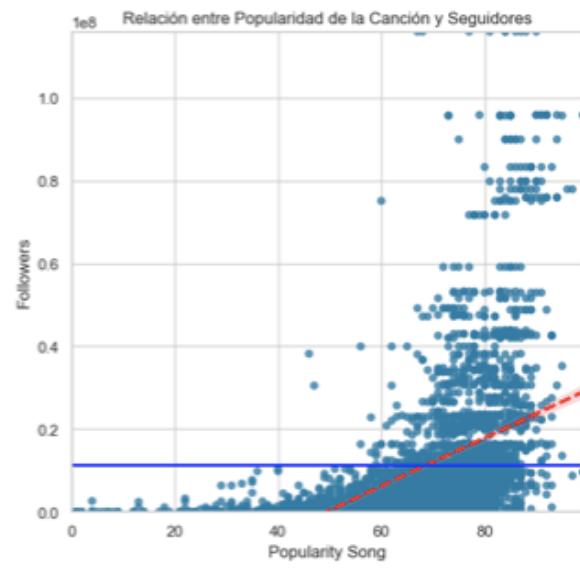
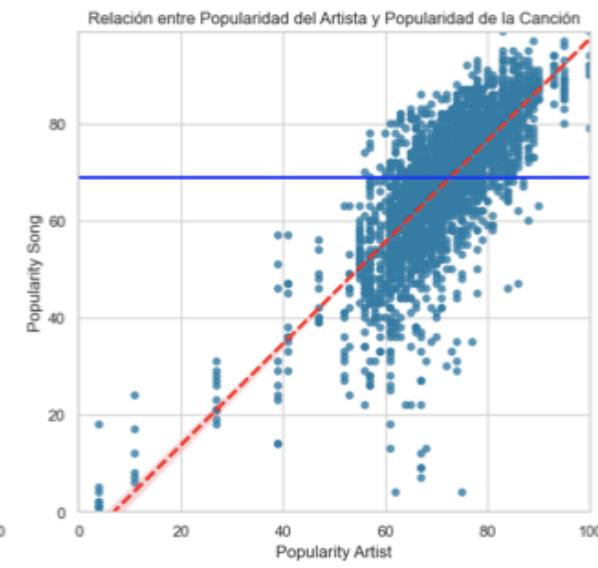
# Top Máxima y Mínima Popularidad



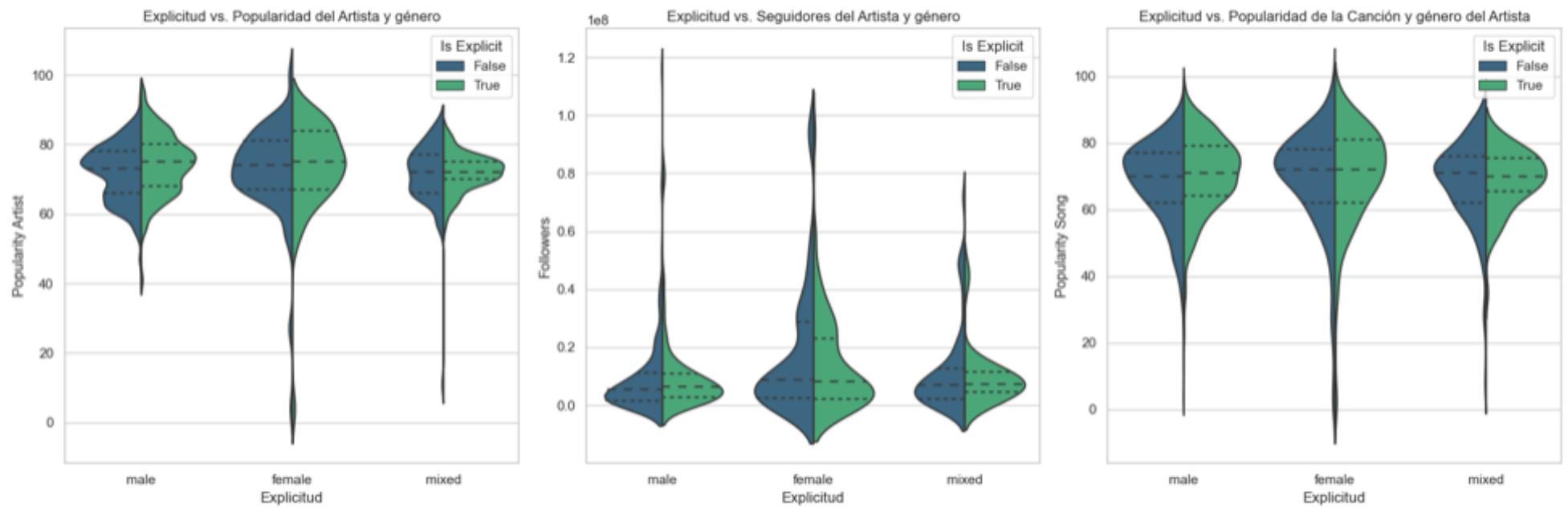
# Índices de Popularidad por canciones



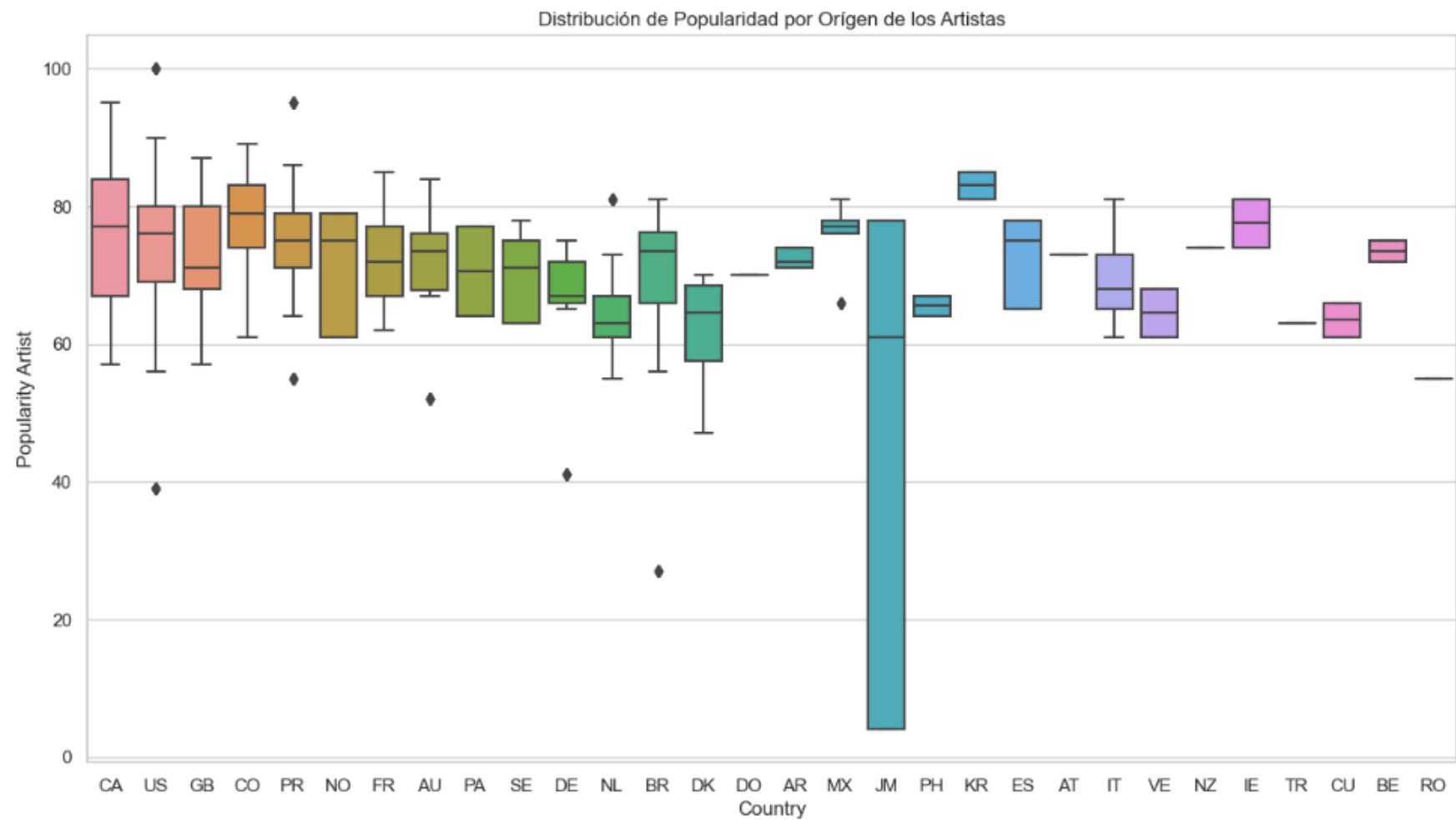
# Relación Popularidad- Seguidores



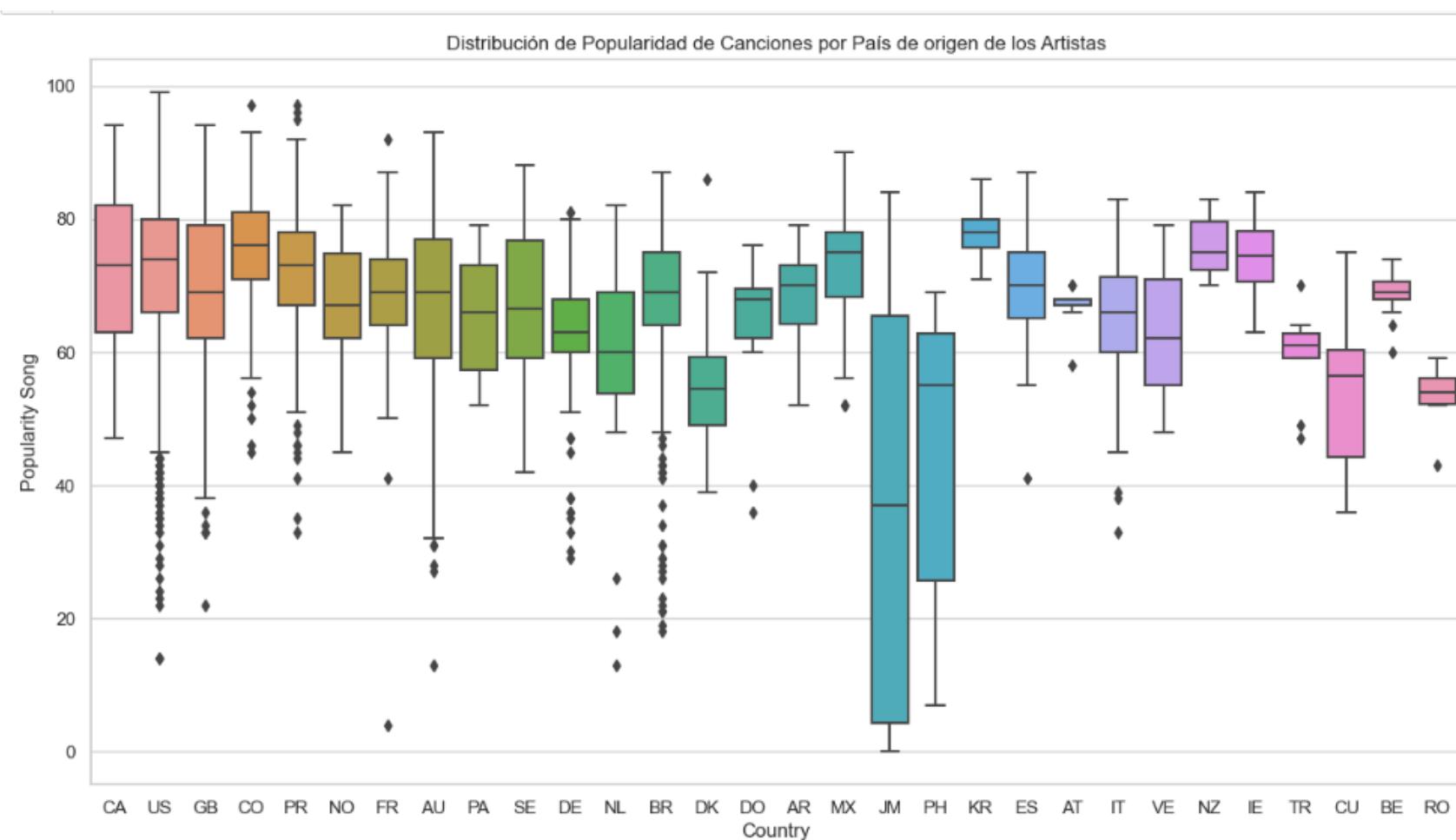
# Relación Explicitud



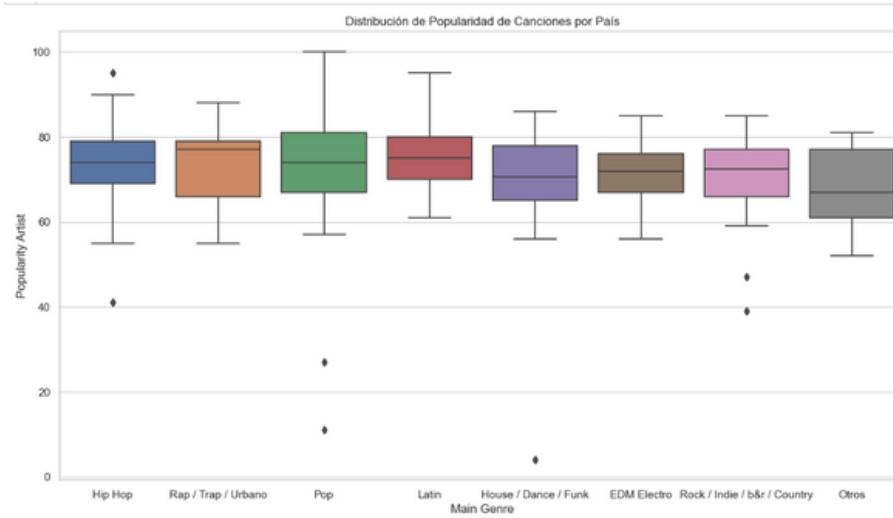
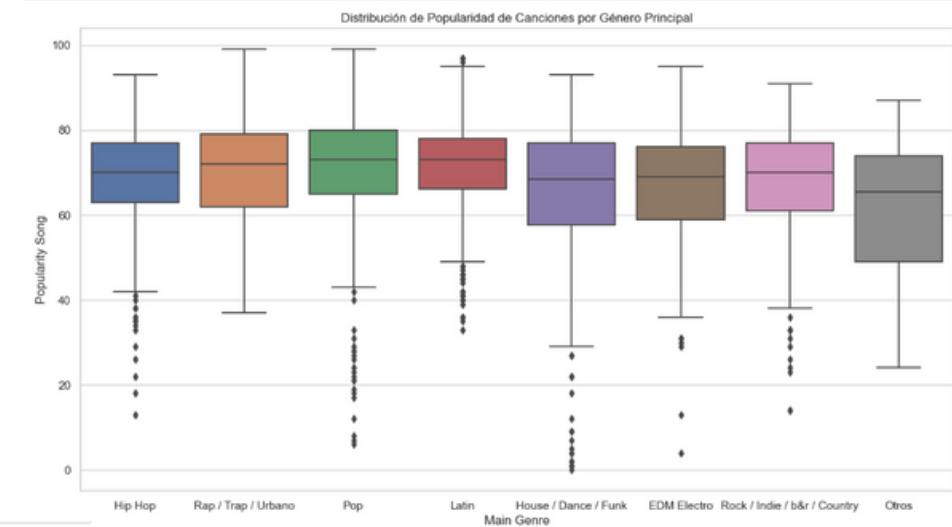
# Relación Popularidad Artistas- País



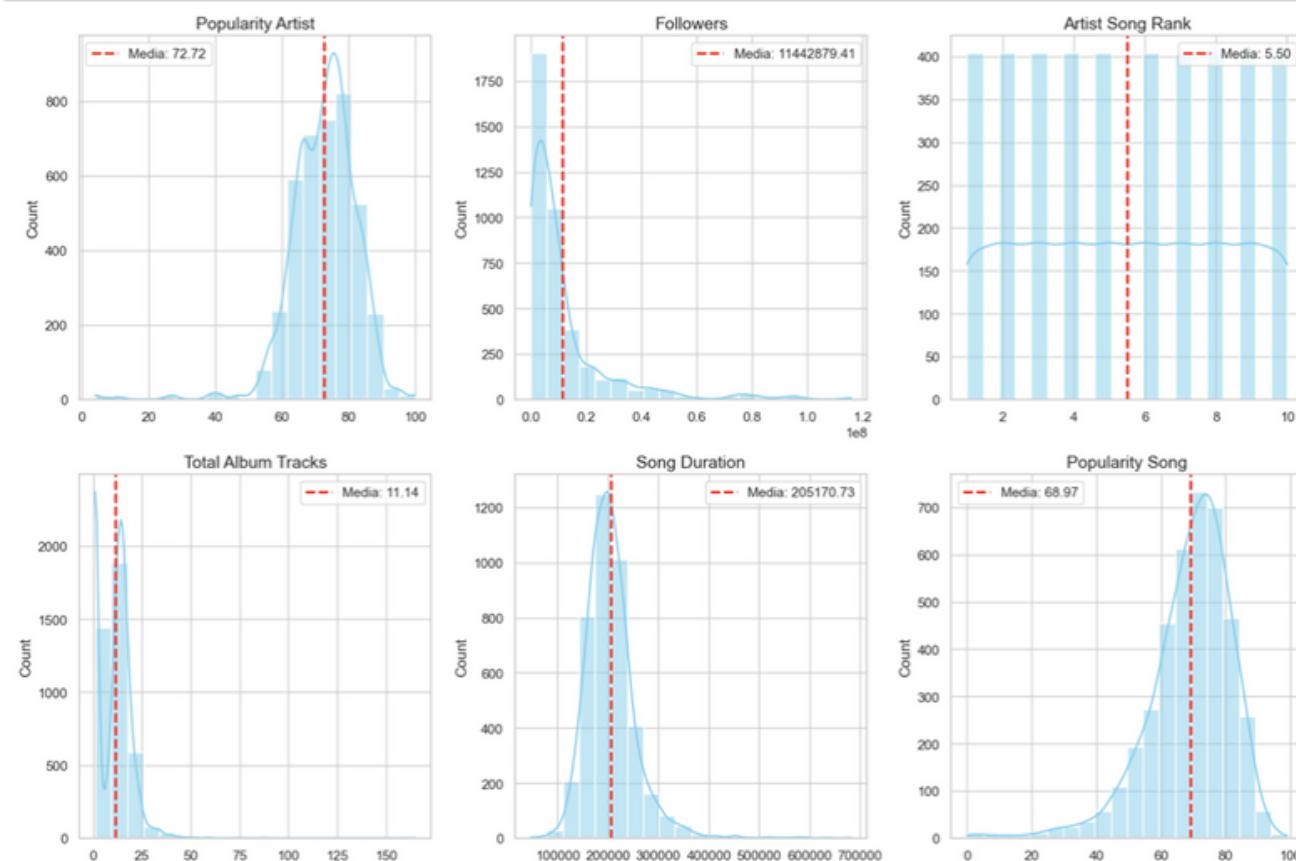
# Relación Popularidad Canción- País



# Distribución popularidad\_género



# Visualización de HISTOGRAMAS



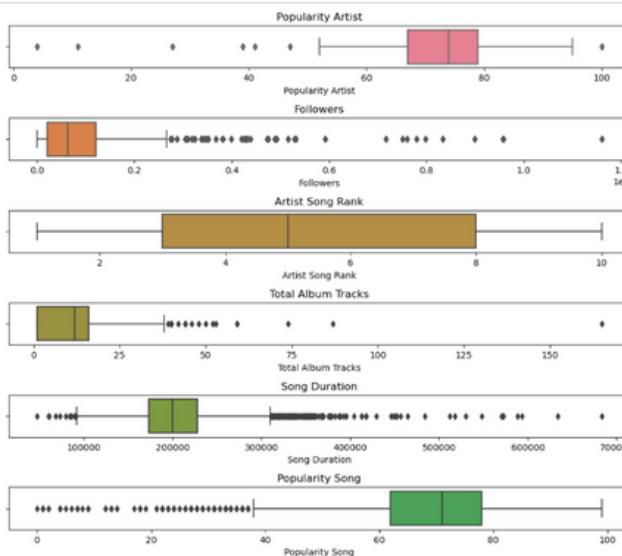
## 5. Preprocesado

# Detección Outliers y Preparación

---

## entrenamiento

- Detección Valores Atípicos / Outliers



- Eliminación de Track Name y Album Name para el Entrenamiento

```
1 # Crear copia del DataFrame original
2 df_procesado = df.copy()
3 df_procesado = df_procesado.drop(['Track Name', 'Album Name'], axis=1, errors='ignore')
4 df_procesado.head()
```

	Gender	Country	Popularity Artist	Followers	Artist Name	Artist Song Rank	Total Album Tracks	Is Explicit	Song Duration	Popularity Song	Main Genre
0	male	CA	95	83298497	Drake	1	23	True	260111	93	Hip Hop
1	male	CA	95	83298497	Drake	2	23	True	247444	91	Hip Hop
2	male	CA	95	83298497	Drake	3	23	True	319191	89	Hip Hop
3	male	CA	95	83298497	Drake	4	14	True	218364	89	Hip Hop
4	male	CA	95	83298497	Drake	5	20	False	173986	89	Hip Hop

## 5. Preprocesado

# Transformaciones

```
1 encoder = OrdinalEncoder()
2 df_procesado['Artist Song Rank'] = pd.to_numeric(df_procesado['Artist Song Rank'], errors='coerce')
3
4 # Reshape la columna 'Artist Song Rank' para que sea una matriz 2D (n_samples, n_features)
5 rank_column_reshaped = df_procesado['Artist Song Rank'].values.reshape(-1, 1)
6
7 # Aplica el encoder
8 df_procesado['Artist Song Rank'] = encoder.fit_transform(rank_column_reshaped)
```

- Convertimos 'Is Explicit' de booleano a int para que pueda ser tratada numéricamente (0/1) y no como (True / False)

```
1 # Cambio el tipo de datos a int
2 df_procesado['Is Explicit'] = df_procesado['Is Explicit'].astype(int)
```

```
df_procesado['Is Explicit'] = pd.to_numeric(df_procesado['Is Explicit'], errors='coerce')

# Selecciono columnas categóricas
columnas_categoricas = df_procesado[['Gender', 'Country', 'Main Genre', 'Artist Name', 'Is Explicit']]
df_procesado = pd.concat([df_procesado.drop(columnas_categoricas, axis=1), pd.get_dummies(columnas_categoricas, drop_first=True)], axis=1)

# 'Popularity Song' no se aplica transformación puesto que es el target.

# Variables numéricas con outliers (aplicar RobustScaler)
columnas_con_outliers = df_procesado[['Popularity Artist', 'Followers', 'Total Album Tracks', 'Song Duration']]

# Aplicar RobustScaler a las columnas con outliers
scaler_robust = RobustScaler()
df_procesado[columnas_con_outliers.columns] = scaler_robust.fit_transform(df_procesado[columnas_con_outliers.columns])
```

	Popularity Artist	Followers	Artist Song Rank	Total Album Tracks	Song Duration	Popularity Song	Is Explicit	Gender_male	Gender_mixed	Gender_other	...	Artist Name_Zara Larsson	Artist Name_Zion & Lennox	Artist Name_Zion.T	Artist Name_Zé Neto & Cristiano	Artist Name_bbno\$	Artist Name_beabadoor
count	4037.0	4037.0	4037.0	4037.0	4037.0	4037.0	4037.0	4037.0	4037.0	4037.0	...	4037.0	4037.0	4037.0	4037.0	4037.0	4037.0
mean	-0.0	1.0	4.0	-0.0	0.0	69.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
std	1.0	2.0	3.0	1.0	1.0	13.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
min	-6.0	-1.0	0.0	-1.0	-3.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
25%	-1.0	-0.0	2.0	-1.0	-0.0	62.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	4.0	0.0	0.0	71.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.0	1.0	7.0	0.0	1.0	78.0	1.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
max	2.0	11.0	9.0	10.0	9.0	99.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0

3 rows x 449 columns

## 6. Modelado

# Entrenamiento

Comprobación de multicolonialidad  
(correlación alta entre variables) con el test de VIF.

```
1 X = df_procesado.drop('Popularity Song', axis=1) #data
2 y = df_procesado['Popularity Song'] #target
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
1 X_train.describe().round(0)
```

	Variable	VIF
0	const	79.102739
1	Popularity Artist	3.395123
2	Followers	1.561414
3	Song Duration	1.028747
4	Popularity Song	2.689992
5	Total Album Tracks	1.075097

Ridge Regression Metrics:  
Mean Squared Error (MSE): 30.286014932665843  
R-squared (R2): 0.8238392622457211

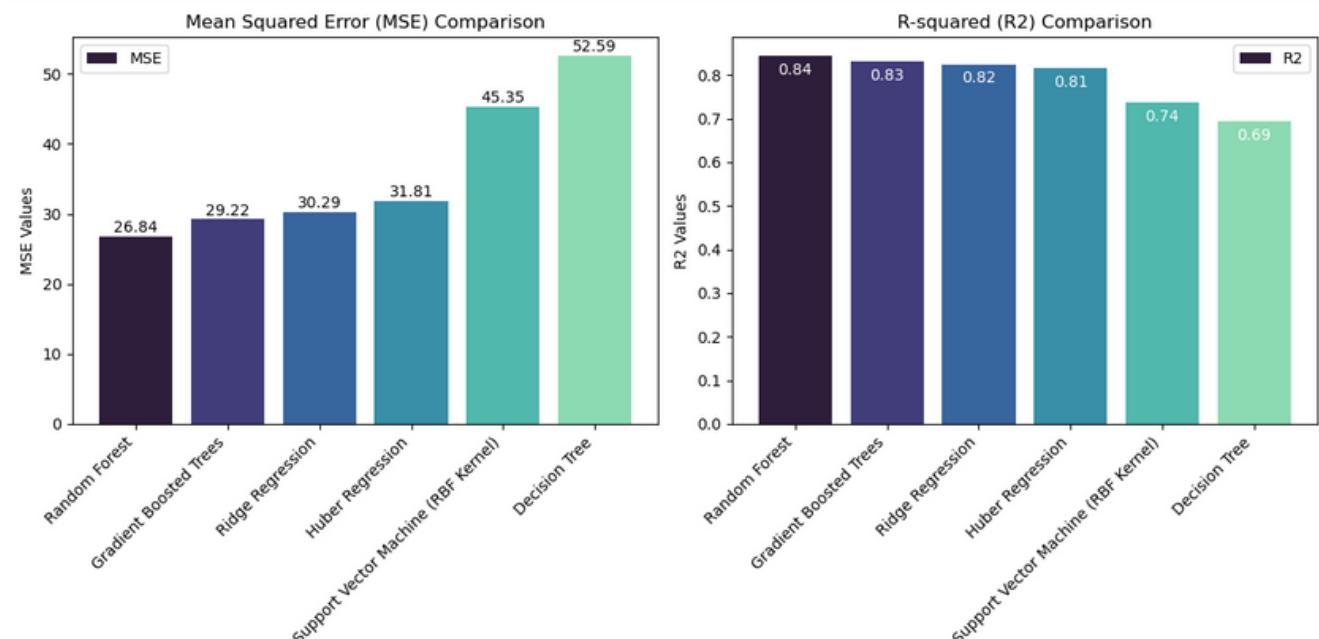
Random Forest Metrics:  
Mean Squared Error (MSE): 26.838974257425744  
R-squared (R2): 0.843900850118726

Support Vector Machine (RBF Kernel) Metrics:  
Mean Squared Error (MSE): 45.345549588288755  
R-squared (R2): 0.7362444188479091

Huber Regression Metrics:  
Mean Squared Error (MSE): 31.814099530804423  
R-squared (R2): 0.8149510506154514

Decision Tree Metrics:  
Mean Squared Error (MSE): 52.59158415841584  
R-squared (R2): 0.6940973487066397

Gradient Boosted Trees Metrics:  
Mean Squared Error (MSE): 29.21890279626615  
R-squared (R2): 0.830046195103433

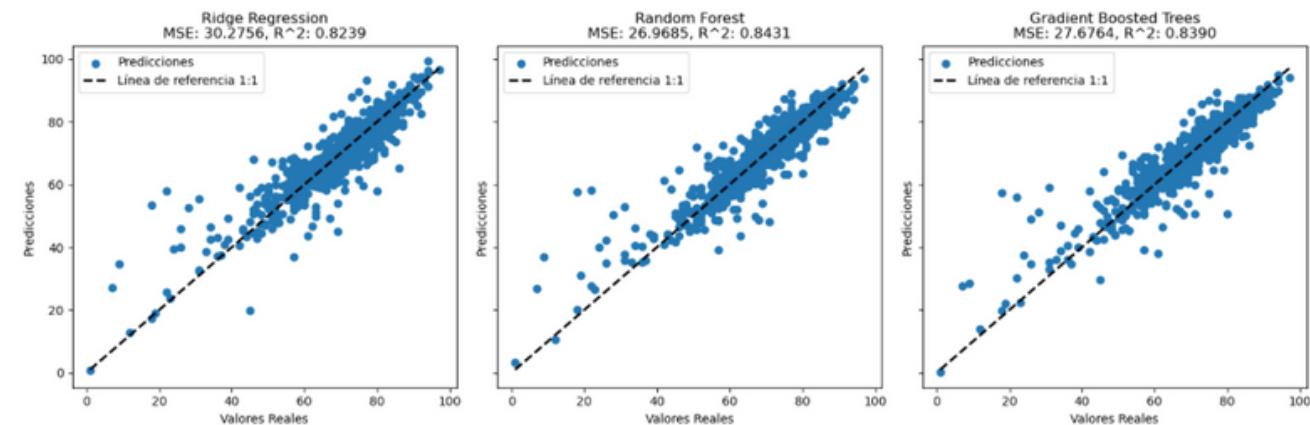


## 6. Modelado

# mejores hiperparámetros

XXXX

	Model	Metric	Best Parameters	Best Score
0	Ridge Regression	MSE	{'alpha': 1.0, 'solver': 'saga'}	-34.950311
1	Ridge Regression	R^2	{'alpha': 1.0, 'solver': 'saga'}	0.795107
2	Random Forest	MSE	{'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}	-36.007235
3	Random Forest	R^2	{'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}	0.787730
4	Gradient Boosted Trees	MSE	{'learning_rate': 0.2, 'max_depth': 4, 'n_estimators': 100}	-35.679397
5	Gradient Boosted Trees	R^2	{'learning_rate': 0.2, 'max_depth': 4, 'n_estimators': 100}	0.790223



# Mejores Hiperparámetros y Entrenamiento

```
selected_model_name = 'Random Forest'
selected_params_dict = best_models[selected_model_name]['MSE']
selected_model = selected_params_dict['best_model']

# Entrenar el modelo seleccionado
selected_model.fit(X_train, y_train)
predictions = selected_model.predict(X_test)

# Calcular métricas
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

# Imprimir métricas
print(f'Modelo {selected_model_name} con mejores hiperparámetros MSE - MSE: {mse:.4f}, R^2: {r2:.4f}')

# Guardar el modelo seleccionado
dump(selected_model, 'selected_model.joblib')
```

Modelo Random Forest con mejores hiperparámetros MSE - MSE: 26.5240, R<sup>2</sup>: 0.8457  
['selected\_model.joblib']

```
# Entrenar los modelos con los mejores hiperparámetros para MSE
for model_name, params_dict in best_models.items():
    model = params_dict['MSE']['best_model']
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    # Calcular métricas
    mse = mean_squared_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)

    print(f'Modelo {model_name} con mejores hiperparámetros MSE - MSE: {mse:.4f}, R^2: {r2:.4f}')
```

Modelo Ridge Regression con mejores hiperparámetros MSE - MSE: 30.2756, R<sup>2</sup>: 0.8239  
Modelo Random Forest con mejores hiperparámetros MSE - MSE: 26.9685, R<sup>2</sup>: 0.8431  
Modelo Gradient Boosted Trees con mejores hiperparámetros MSE - MSE: 27.6764, R<sup>2</sup>: 0.8390

## 8.- Conclusiones

# RANDOM FOREST

	Modelo	MSE_Antes	R2_Antes	MSE_Despues	R2_Despues
0	Ridge Regression	30.286015	0.823839	30.2767	0.8239
1	Random Forest	27.808511	0.838250	26.9622	0.8432
2	Gradient Boosted Trees	29.434486	0.828792	27.4993	0.8400

Nos quedamos con RANDOM FOREST como modelo por haber sido el más estable y con mejor puntuación, aunque los tres modelos han ofrecido un muy buen rendimiento y serían óptimos para obtener mi objetivo: "Construir y evaluar modelos de aprendizaje automático para aplicaciones específicas en la industria de la música y el entretenimiento."