ITBA
Instituto Tecnológico de Buenos Aires

02/05/2024

82.20 Análisis Predictivo Avanzado - Primer Trabajo Práctico

Azul de los Angeles Makk (61589) y Bruno Soifer (62423)

Agenda

01 **Introducción**
Problema de negocio a resolver

02 **Dataset a trabajar**
Introducción a la base y a sus variables

03 **EDA**
Gráficos exploratorios

04 **Desarrollo del problema**
Aplicación de técnicas de *feature engineering* y manipulación de variables

05 **Conclusión**
Acciones de negocio que se podrían tomar

INTRODUCCIÓN

Introducción

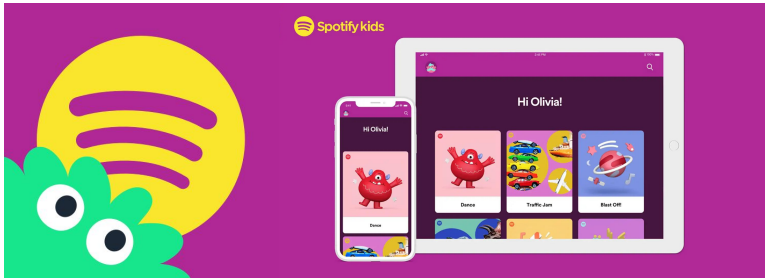
Problema de negocio a resolver: identificar qué canciones pueden ser catalogadas con “contenido explícito” en función de distintas características, para el desarrollo de Spotify Kids.

El mismo implica:

1. Un entorno seguro y adecuado
2. Contenido apropiado para el grupo etario

Spotify Kids update lets parents check listening history and block songs

Oddly enough, a song can only be blocked if it's already in the kid's Listening History



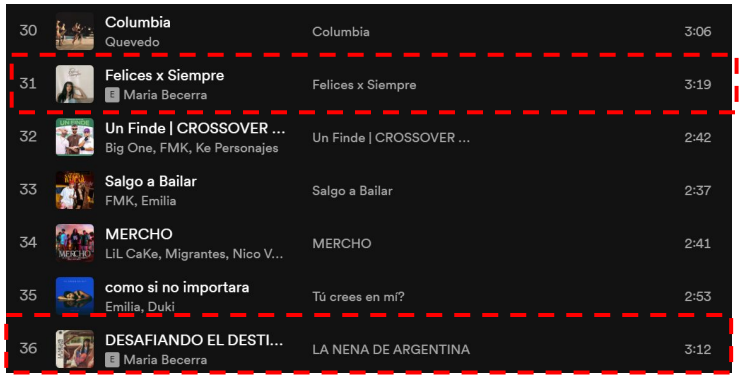
Introducción

¿Qué es *contenido explícito*?

Se considera que una canción contiene contenido explícito cuando cumple con alguno de los siguientes criterios:

1. Contiene lenguaje soez
2. Referencia de violencia, abuso físico o sexual
3. Referencias al comportamiento sexual explícito
4. Lenguaje discriminatorio a cualquier grupo, etnia o género

"Spotify incluye contenido explícito porque lo ofrecemos de la forma en que el artista pretende que se escuche".



DATASET

Dataset

- El dataset a trabajar contiene **114.000** canciones de Spotify.
  - Hay 1.000 canciones por cada género musical (114 en total).
- Hay **21** columnas en la base, con información correspondiente a cada canción.

Unnamed: 0	1
track_id	4qPNDBW1i3p13qLCt0Ki3A
artists	Ben Woodward
album_name	Ghost (Acoustic)
track_name	Ghost - Acoustic
popularity	55
duration_ms	149610
explicit	False
danceability	0.42
energy	0.166
key	1
loudness	-17.235
mode	1

speechiness	0.0763
acousticness	0.924
instrumentalness	0.000006
liveness	0.101
valence	0.267
tempo	77.489
time_signature	4
track_genre	acoustic

Por cuestiones de procesamiento, se han tomando únicamente los géneros musicales 'classical', 'metal', 'jazz', 'punk-rock', 'techno', 'reggae', 'sleep', 'trance','study' y 'hip-hop'

Dataset: variables

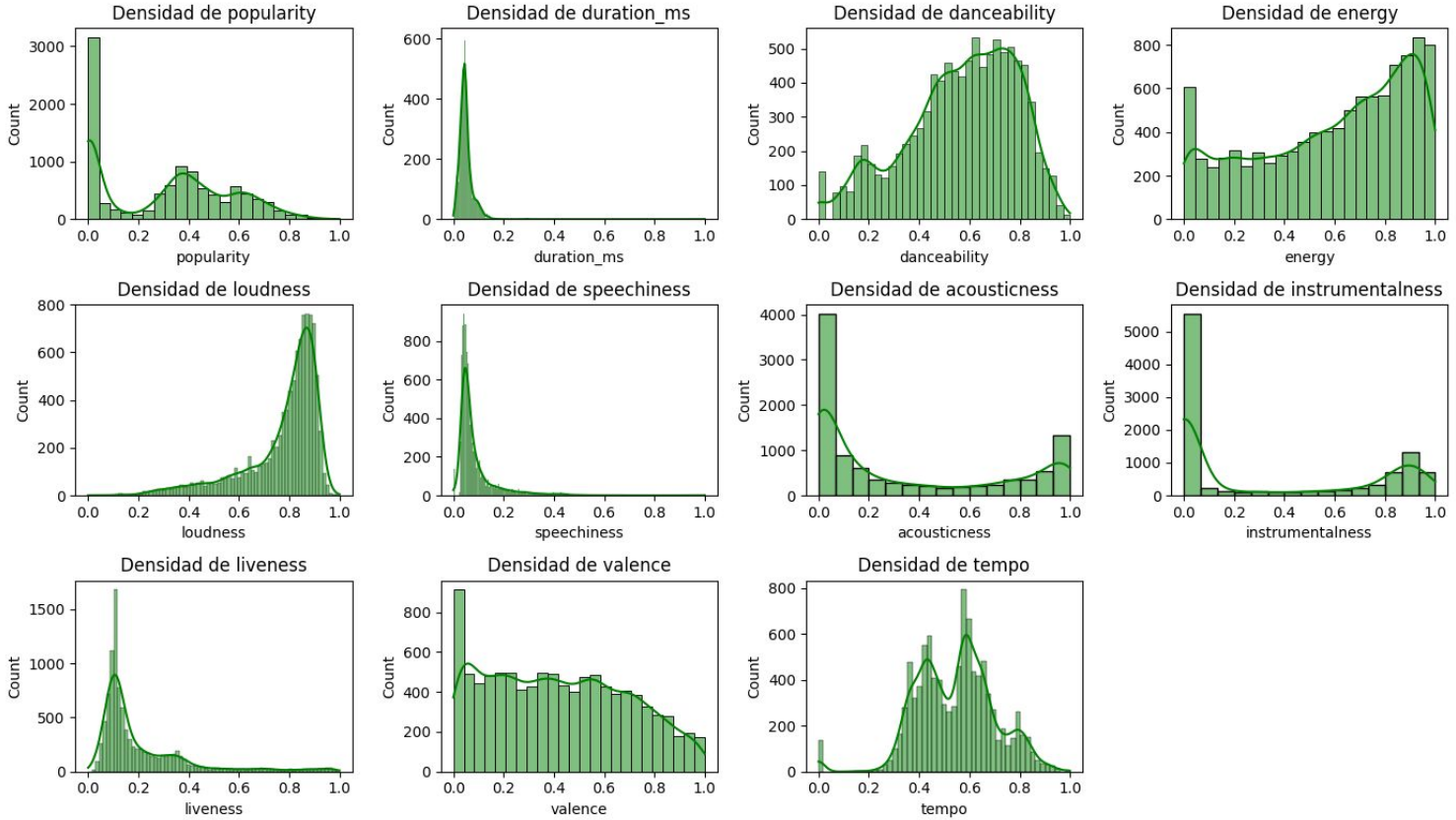
- **Numéricas**
  - Speechiness [0-1]
  - Acousticness [0-1]
  - Instrumentalness [0-1]
  - Liveness [0-1]
  - Valence [0-1]
  - Tempo
  - Popularidad [0-100]
  - Duración en milisegundos
  - Danceability [0-1]
  - Energy [0-1]
  - Loudness (dB)
- **Categóricas**
  - Explicit T/F
  - Key
  - Genre
  - Mode 0/1
  - Time Signature
- **Extras**
  - Track_id
  - Artista
  - Album Name
  - Track Name

Ninguna de las variables presenta valores nulos (NAs)

EDA

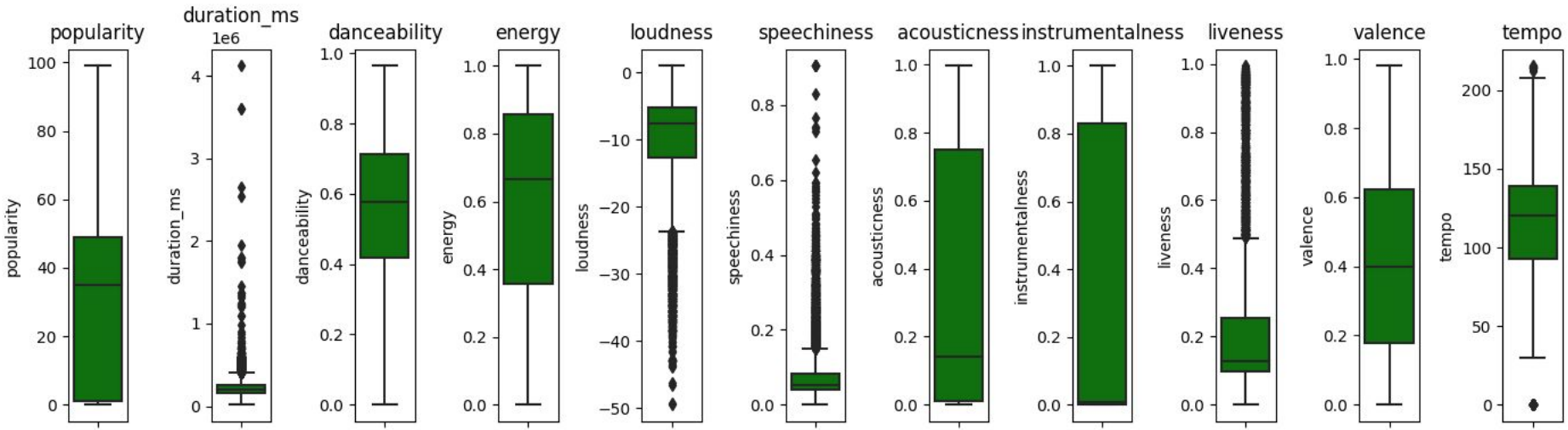
EDA

1. Densidad de variables numéricas



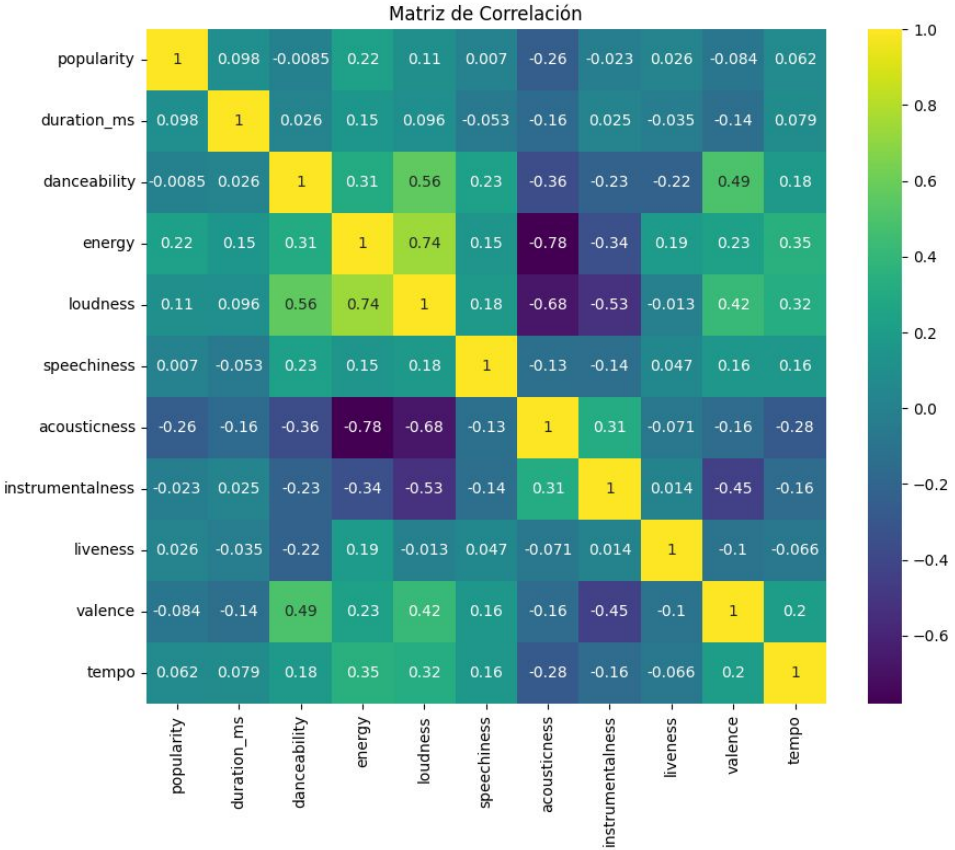
EDA

2. Análisis de outliers



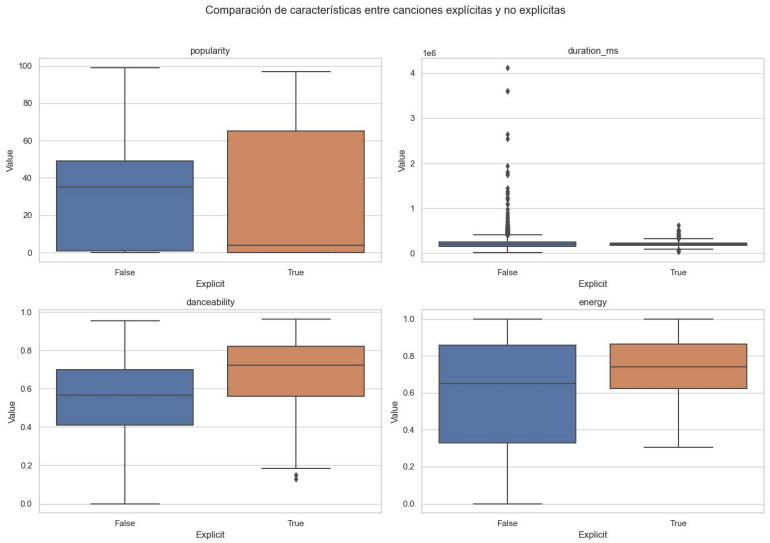
EDA

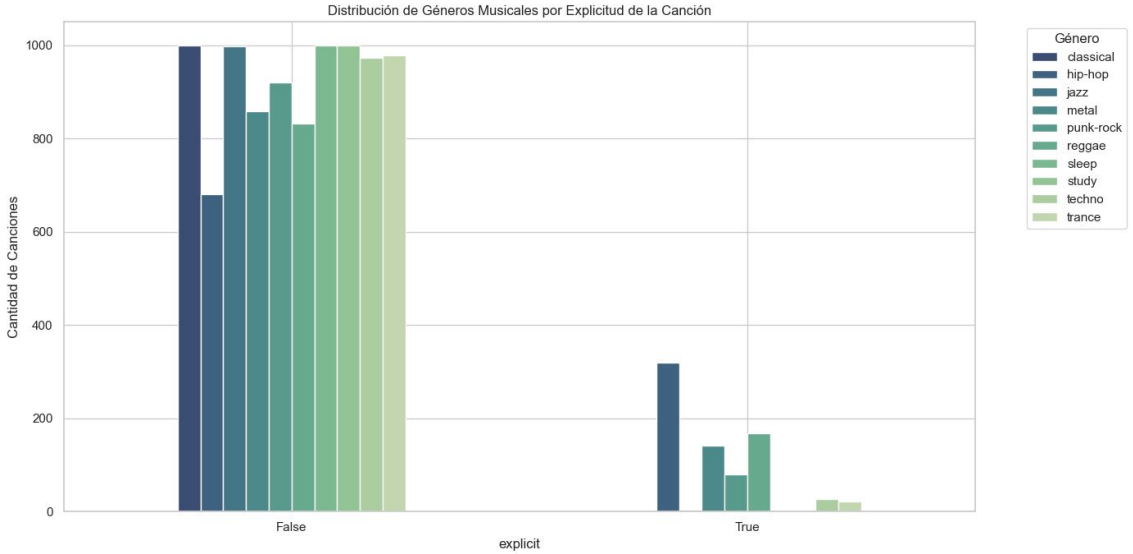
3. Matriz de correlación



EDA

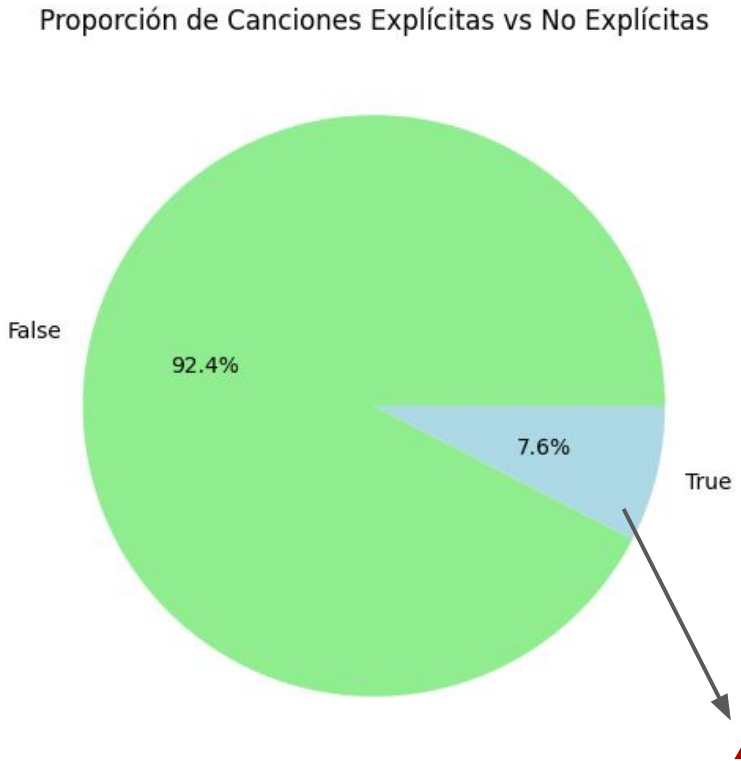
4. Canciones explícitas vs. no explícitas





EDA

5. Proporción de canciones explícitas

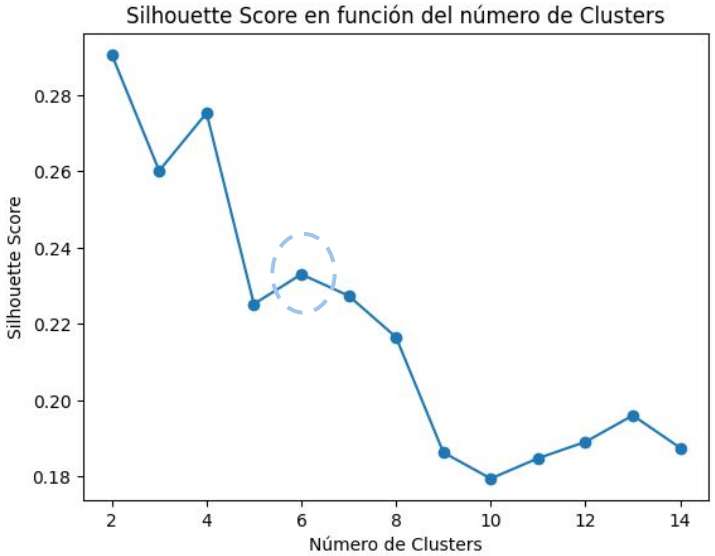


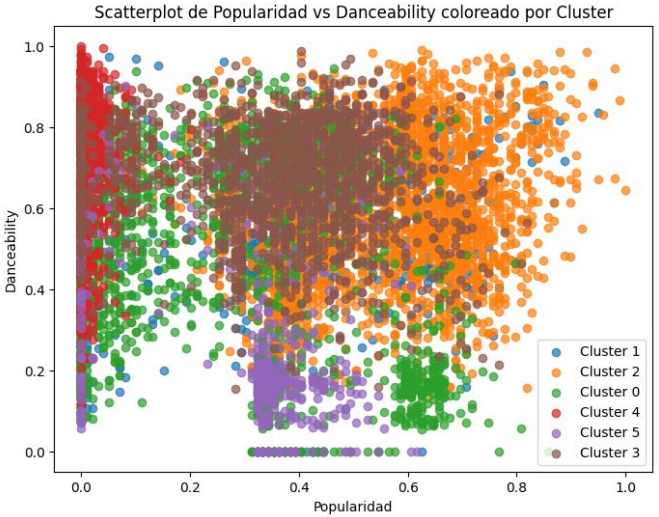
Desbalanceo moderado: la clase positiva representa un 7,6% del total del dataset

EDA

6. Clusterización de los datos

Clustering **jerárquico**

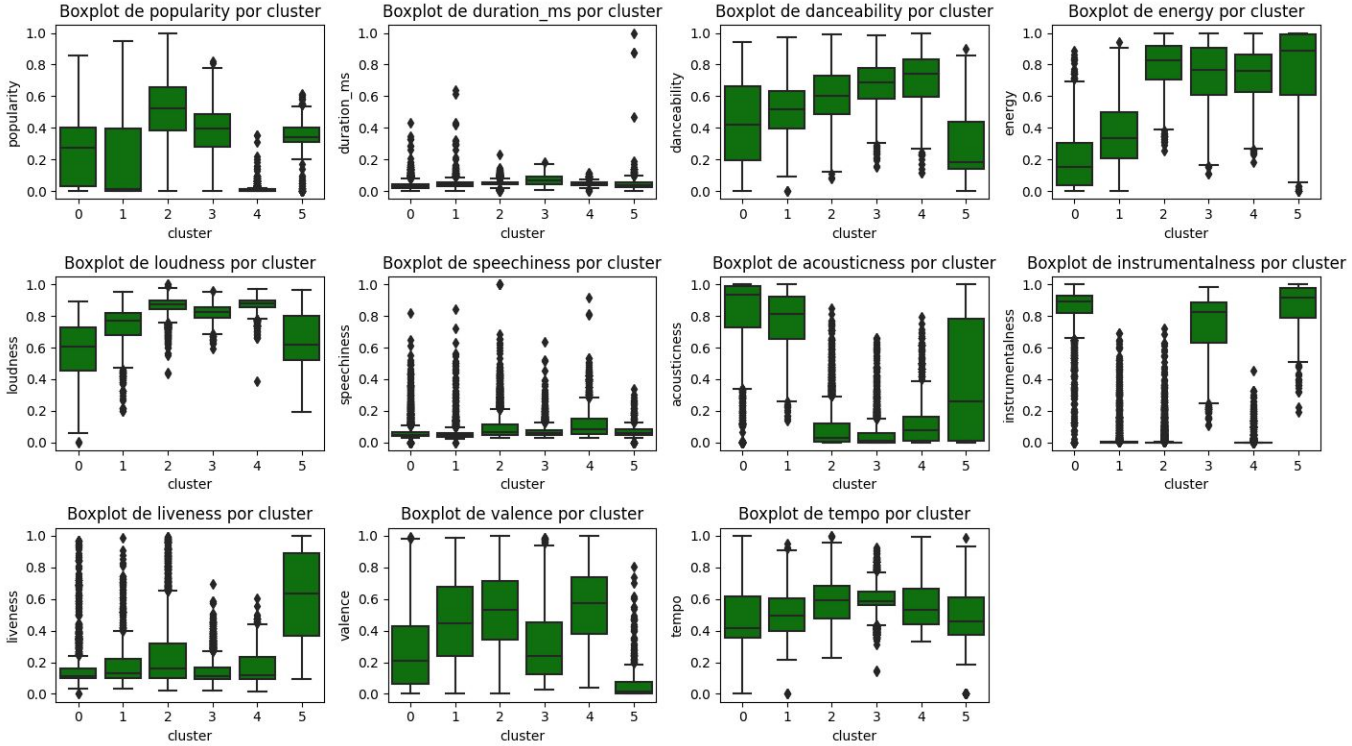




EDA

6. Clusterización de los datos

Clustering **jerárquico**



DESARROLLO DEL PROBLEMA

Desarrollo del problema

1. Encoders

a. Label Encoder: ordena las categorías de forma alfabética y les asigna un número. Útil para algoritmos como XGBoost (sklearn requiere que todas las variables input sean numéricas)

```python
def label_encoder(data):
    le = LabelEncoder()
    encoded_data = le.fit_transform(data)
    return encoded_data
```

```python
df['track_genre_encoded'] = label_encoder(df['track_genre'])

print(df[['track_genre', 'track_genre_encoded']])
df = df.drop(['track_genre'], axis=1)
```

	track_genre	track_genre_encoded
16000	classical	0
16001	classical	0
16002	classical	0
16003	classical	0
16004	classical	0
...	...	...
110995	trance	9
110996	trance	9
110997	trance	9
110998	trance	9
110999	trance	9

	artists	artists_encoded
16000	Bombay Jayashri	536
16001	Shankar;Ehsaan;Loy;Alisha Chinai;Shankar Mahad...	2811
16002	Bombay Jayashri;DJ Aftab	537
16003	Bombay Jayashri	536
16004	Bombay Jayashri;Swattrex	540
...	...	...
110995	NG Rezonance;PHD	2162
110996	NG Rezonance;PHD	2162
110997	NG Rezonance;Begbie	2159
110998	NG Rezonance	2156
110999	NG Rezonance	2156

Elección de un modelo

2. XGBoost

En primer lugar, se ha particionado el dataset en train (80%) y test (20%)

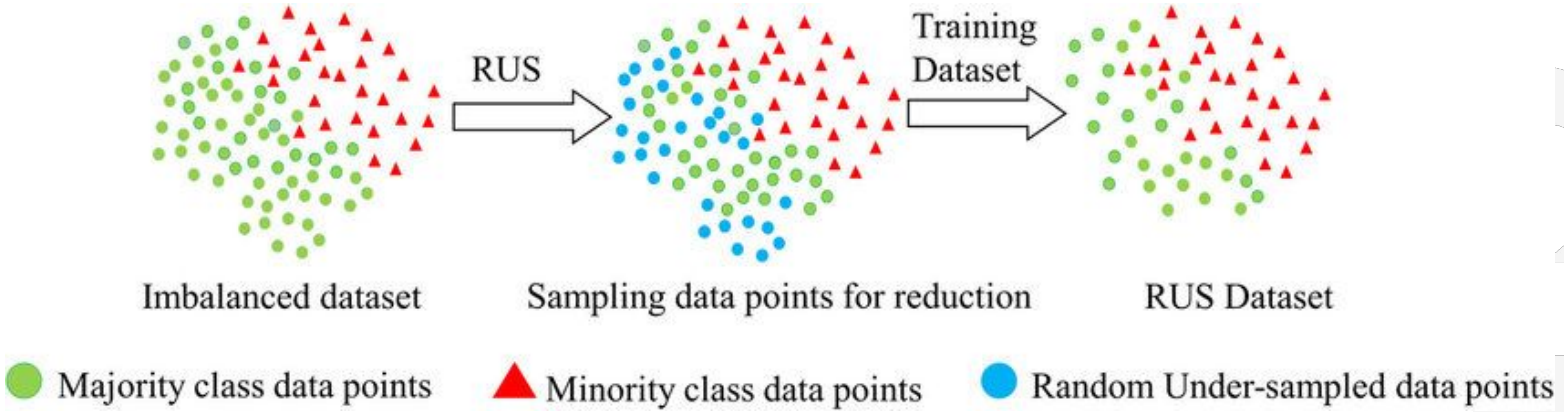
Para el modelo predictivo se ha utilizado XGBoost. Utilizando GridSearchCV se han seleccionado los siguientes parámetros como los mejores estimadores:

n_estimators	100	500	1000
learning_rate	0,05	0,1	0,2
max_depth	3	4	5
min_child_weight	1	2	3

Desarrollo del problema

3. Balanceo de clases

a. RandomUnderSampler: identifica la clase con más instancias en el conjunto de datos y las reduce de forma aleatoria hasta alcanzar un set de datos más equilibrado.



- ✓ Método fácil de implementar
- ✓ Puede ayudar a reducir el sesgo para la clase "no implícita"
- ✗ Pérdida de información
- ✗ Posibilidad de overfitting si la muestra es sesgada

Desarrollo del problema

3. Balanceo de clases

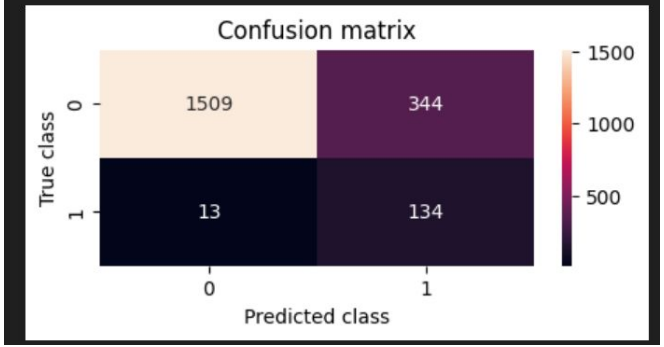
a. RandomUnderSampler

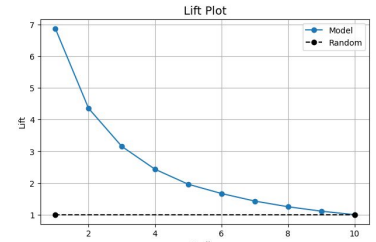
```
rus = RandomUnderSampler(random_state=42, replacement=True)
x_rus, y_rus = rus.fit_resample(X_train, y_train)

print('original dataset shape:', Counter(y_train))
print('Resample dataset shape', Counter(y_rus))
```

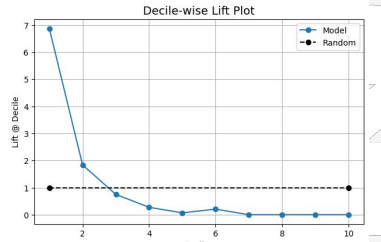
```
original dataset shape: Counter({0: 7388, 1: 612})
Resample dataset shape Counter({0: 612, 1: 612})
```

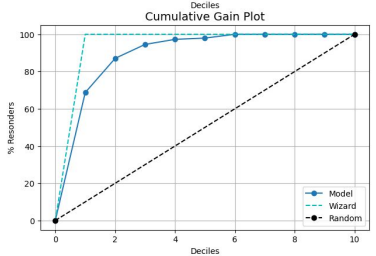
Recall: **0.9953**

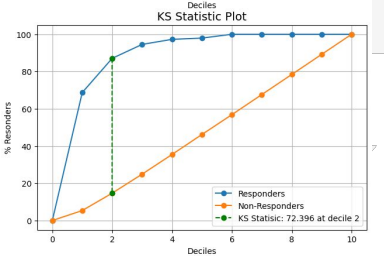
```
ROCAUC score Under-sampling: 0.86
Accuracy score: 0.82
F1 score: 0.43
Confusion matrix:
```







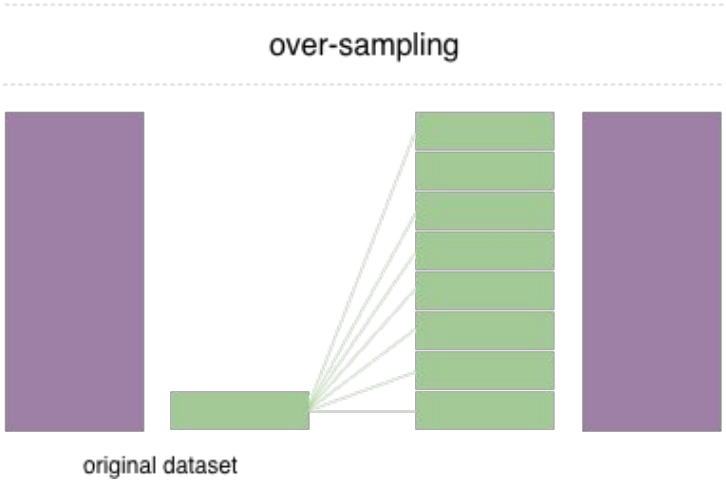




Desarrollo del problema

3. Balanceo de clases

b. RandomOverSampler: identifica la clase con más instancias en el conjunto de datos y las reduce de forma aleatoria hasta alcanzar un set de datos más equilibrado.



- ✓ Método fácil de implementar
- ✓ Aumento de información
- ✗ Posibilidad de overfitting si la muestra no representa la distribución de la clase minoritaria
- ✗ Más lento

Desarrollo del problema

3. Balanceo de clases

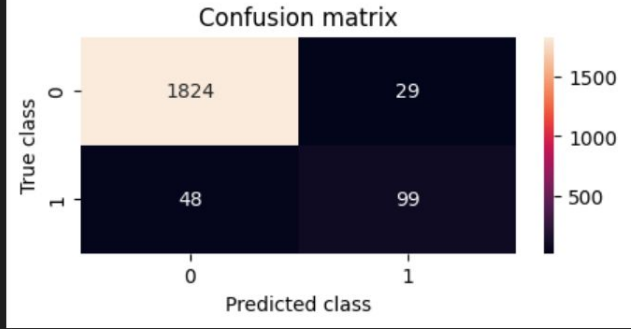
b. RandomOverSampler

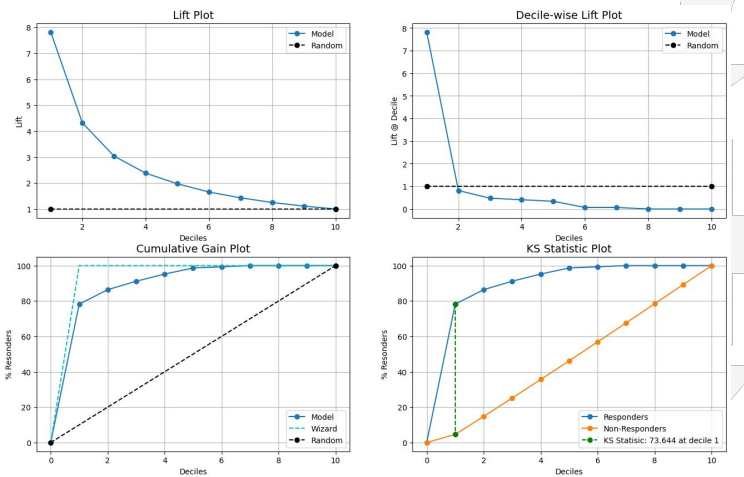
```
ros = RandomOverSampler(random_state=42)

x_ros, y_ros = ros.fit_resample(X_train, y_train)

print('Original dataset shape', Counter(y_train))
print('Resample dataset shape', Counter(y_ros))
```

```
Original dataset shape Counter({0: 7388, 1: 612})
Resample dataset shape Counter({0: 7388, 1: 7388})
```

Recall: **0.9743**

```
ROCAUC score Over-sampling: 0.83
Accuracy score: 0.96
F1 score: 0.72
Confusion matrix:
```

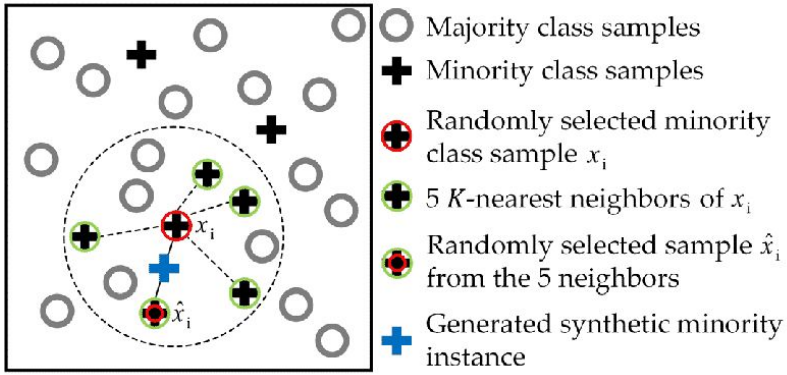




Desarrollo del problema

3. Balanceo de clases

c. Synthetic Minority Oversampling Technique (SMOTE): A diferencia del Random Over Sampler, que replica instancias de la clase minoritaria, SMOTE genera instancias sintéticas de la clase minoritaria mediante interpolación entre instancias vecinas en el espacio de características.



✔ Ayuda a mitigar el overfitting: variedad en los datos sintéticos.

✘ Las instancias sintéticas se generan en el espacio de características existente y pueden no representar completamente la variabilidad de la clase minoritaria.

Desarrollo del problema

3. Balanceo de clases

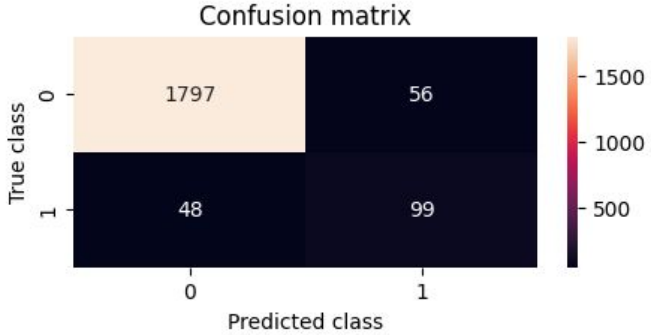
c. SMOTE

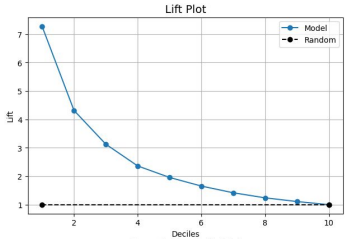
```
smote = SMOTE()
x_smote, y_smote = smote.fit_resample(X_train, y_train)

print('Original dataset shape', Counter(y_train))
print('Resample dataset shape', Counter(y_smote))
```

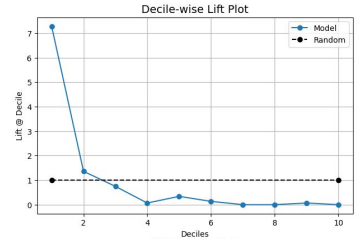
```
Original dataset shape Counter({0: 7388, 1: 612})
Resample dataset shape Counter({0: 7388, 1: 7388})
```

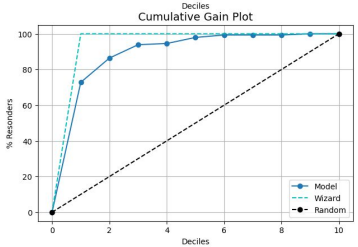
Recall: **0.9739**

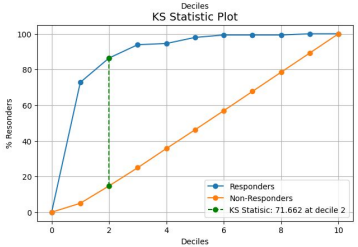
```
ROCAUC score SMOTE: 0.82
Accuracy score: 0.95
F1 score: 0.66
Confusion matrix:
```









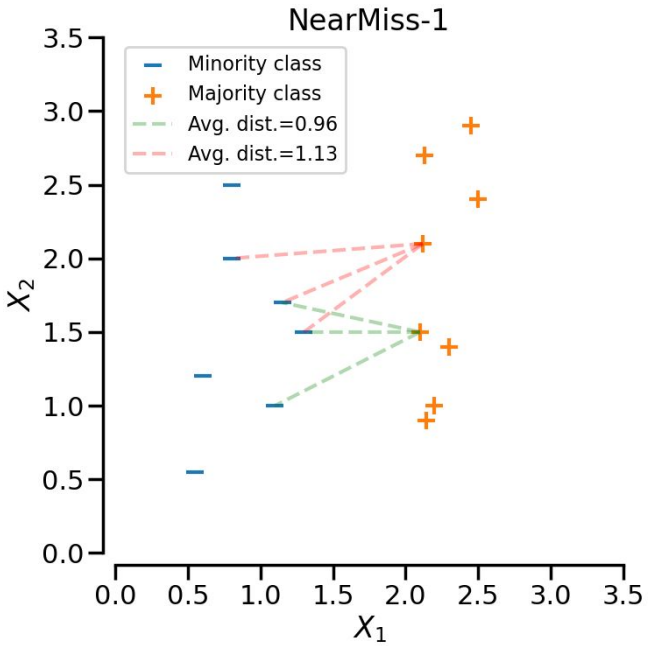


Desarrollo del problema

3. Balanceo de clases

d. Near Miss: técnica de undersampling, donde el objetivo es seleccionar los casos de la clase mayoritaria más semejantes a la minoritaria. Tiene tres variantes, siendo la tercera aquella con mejores resultados:

Se seleccionan k (en nuestro caso, 5) ejemplos cercanos de la clase mayoritaria por cada ejemplo de la clase minoritaria.



Desarrollo del problema

3. Balanceo de clases

d. NearMiss

Uso de Random Forest Classifier

```
# Instanciar y aplicar NearMiss con versión 3
nm = NearMiss(version=3, n_neighbors=5)
x_nm_3, y_nm_3 = nm.fit_resample(X_train, y_train)
print('Resampled dataset shape (NearMiss version 3):', Counter(y_nm_3))

clf_nm_3 = RandomForestClassifier(random_state=42)
clf_nm_3.fit(x_nm_3, y_nm_3)

predict_y_nm_3 = clf_nm_3.predict(X_test)
✓ 0.8s
Resampled dataset shape (NearMiss version 3): Counter({0: 612, 1: 612})
```

```
ROCAUC score NearMiss version 1: 0.83
Accuracy score: 0.8
F1 score: 0.39
```

```
ROCAUC score NearMiss version 2: 0.56
Accuracy score: 0.24
F1 score: 0.15
```

```
ROCAUC score NearMiss version 3: 0.83
Accuracy score: 0.86
F1 score: 0.45
```

True Class	1605	248
	36	116
	Predicted Class	

Recall: **0.978**

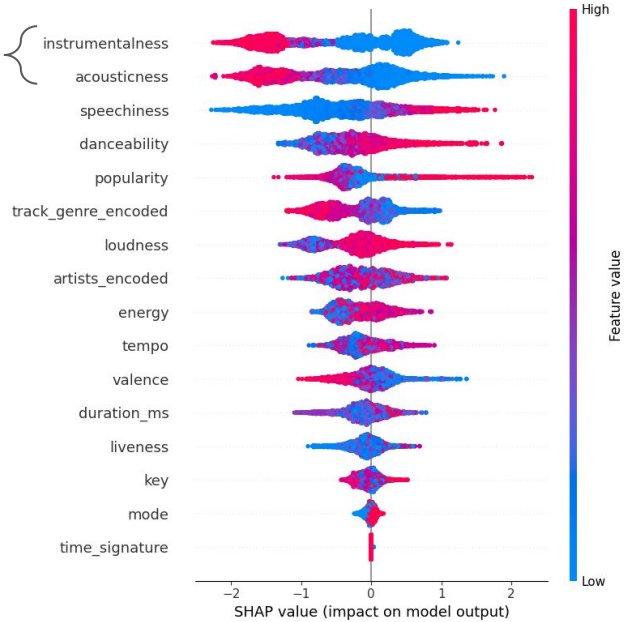
Desarrollo del problema

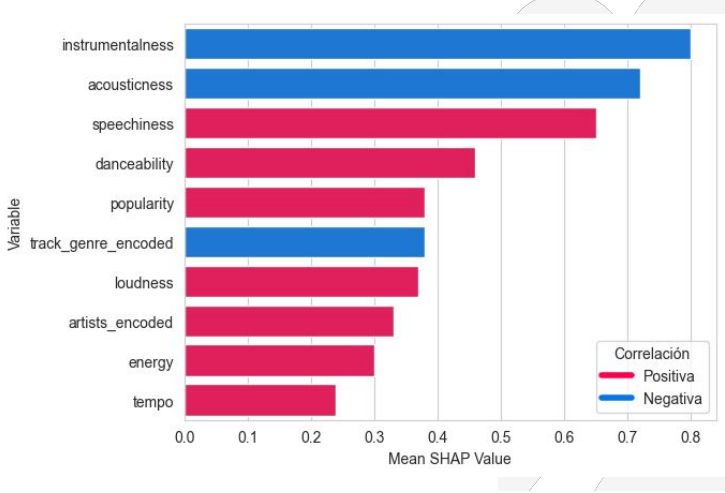
4. Feature Importance - Shap (SHapley Additive exPlanations)

Aporte de las variables al modelo

- Los valores Shapley son el promedio de las contribuciones marginales para todas las permutaciones de las variables predictoras.
- Se calcula la importancia del feature comparando lo que el modelo predice con y sin la feature.

Cuanto menor es el valor de la variable, hay más probabilidad de alcanzar la clase positiva



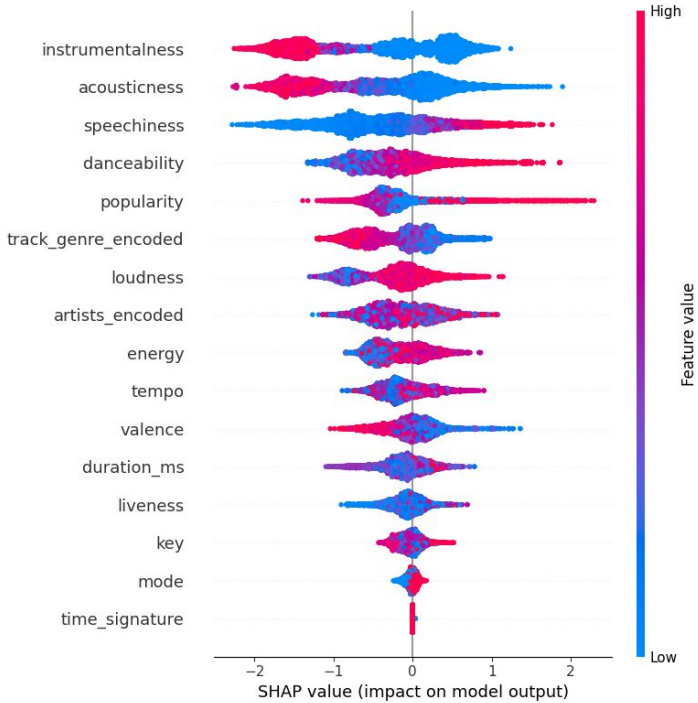


Desarrollo del problema

4. Feature Importance - Shap (SHapley Additive exPlanations)

Comparación con el formato tradicional de importancia de las variables

var	imp
track_genre_encoded	0.138488
instrumentalness	0.111294
popularity	0.092308
speechiness	0.085292
danceability	0.076058
acousticness	0.074736
valence	0.060612
loudness	0.059703
artists_encoded	0.054456
energy	0.048114
tempo	0.045476
duration_ms	0.044496
liveness	0.040340
key	0.038358



CONCLUSIÓN

Conclusiones

- Se encontraron mejores resultados en las técnicas de OverSampling que en las de UnderSampling
  - Siendo SMOTE la de mejor resultado
- Un ajuste de hiper parámetros podría mejorar los resultados obtenidos
  - Por ejemplo, Cross-Validation
- Variables numéricas como instrumentalness, acousticness y speechiness son las que tienen mayor importancia
- Se podrían crear nuevas variables a partir de las preexistentes como también distintos clústers
- Otros modelos para probar: DecisionTreeClasifier, RandomForrestClassifier, GradientBoostingClassifier, CatBoost, SVM, entre otros

ITBA
Instituto Tecnológicc
de Buenos Aires

¡Gracias!

Link del repositorio:
https://github.com/bsoifer/PredictivoAvanzado