Clasificación del año en el que salió una canción entre 2010-2019

Adrián García Tapia

Abstract—Se ha utilizado el dataset "Top Spotify songs from 2010-2019 – BY YEAR" para intentar determinar el año en el que salió una canción únicamente a partir de sus características. Este es un problema multiclase para un dataset desbalanceado.

Se ha aplicado un análisis de los datos para posteriormente preprocesarlos con la única eliminación de 3 columnas identificativas y una fila con un valor anómalo.

La métrica escogida ha sido el F1-Score y se han obtenido unos resultados mediocres para el RandomForest utilizado al final.

Keywords—Clasificación multiclase, carencia de datos, f1-score, Random Forest, spotify

1 Introducción

El dataset escogido en esta práctica es "Top Spotify songs from 2010-2019 – BY YEAR", que pertenece al conjunto de datasets revisados de Kaggle que se propusieron.

Escogí probar si se podía entrenar un modelo para saber, en función de cada canción, en qué año salió.

Este dataset únicamente incluye canciones populares hasta cierto punto, pero aun así la idea era ver si podría clasificar bien estas canciones y de esta forma si se podría escalar más adelante con un conjunto de datos mayor.

Una vez definido el objetivo, empecé a examinar los datos.

2 Metodologia de tratado de datos

En primer lugar, accedí a la página original de la que se extrajeron los datos: Organize Your Music, donde se da una explicación de qué es cada atributo del dataset. Estos son los atributos y sus descripciones:

- Unnamed(Entero): Atributo que no tiene nombre, pero que en su descripción en Kaggle se indica que es la ID de la fila. Por lo tanto, este atributo se puede eliminar al no guardar ningún tipo de información importante
- title(Categórico): Título de la canción
- artist(Categórico): Autor de la canción
- top genre(Categórico): Género de la canción
- year(Entero): Año en el que se lanzó la canción. Según la página de la que provienen los datos, el lanzamiento de una canción puede no ser el esperado debido a temas de lanzamiento, relanzamiento, etc.
- bpm(Entero): Tempo de la canción
- nrgy(Entero): Nivel de energía de la canción
- dnce(Entero): Nivel de facilidad a la hora de bailar la canción
- dB(Entero): Volumen de la canción
- live(Entero): Probabilidad de que haya sido grabada en vivo
- val(Entero): Nivel de positivismo en la canción
- dur(Entero): Duración de la canción
- acous(Entero): Nivel de acústica de la canción
- spch(Entero): Cantidad de palabras que contiene
- pop(Entero): Popularidad de la canción

Tanto "Unnamed", como "title" y "artist" decidí descartarlos del dataset debido a que uno tenía una función puramente identificativa y los otros dos aunque fuesen rasgos de la canción que la identifican, son categóricos y prácticamente únicos para cada canción. Por lo que decidí englobarlos como atributos identificativos para descartar.

A continuación, una vez tuve más claro el significado de cada atributo, comprobé con el histograma de la Figura 1 que se trata de un problema desbalanceado. Por lo que es importante una buena estratificación en la división entre train y test y en el cross validation de más adelante.

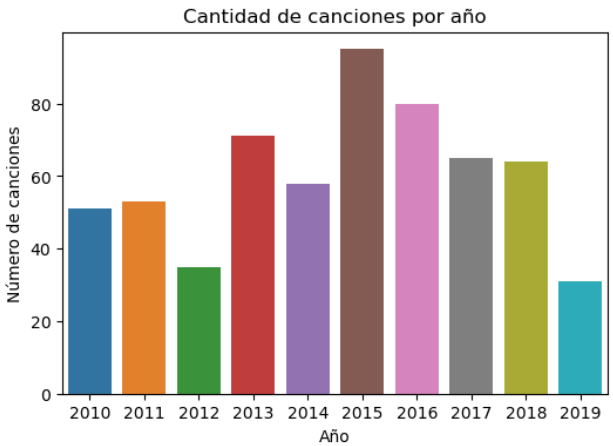

Fig 1. Histograma de la cantidad de canciones en función del año.

Por último, decidí generar un gráfico de nube de puntos entre cada atributo numérico para ver la relación que había entre sí. El único detalle que extraje fue un punto de la nube de puntos de los decibelios y cualquier otro atributo, como es el caso de la nube de puntos izquierda de la Figura 2. Decidí que era un outlier así que lo extraje la fila del conjunto y volví a generar la nube de puntos, obteniendo así un resultado mucho más separado y aparentemente organizado en niveles de decibelios discretizados.

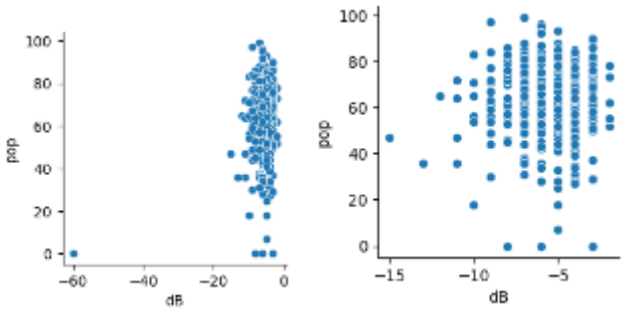

Fig 2. Nube de puntos entre los atributos "pop" y "dB" antes de retirar el punto de -60 y después de retirarlo.

Con este análisis hecho, hice el preprocessing eliminando las columnas que hacían de identificador: "Unnamed", "title" y "artist", y la fila mencionada anteriormente.

Por otro lado, decidí codificar el atributo categórico "top genre" con mean encoding para extraer una relación entre los géneros más populares en función del año y así ayudar al posterior entrenamiento.

Generé una matriz de correlación entre cada atributo para ver, tras todos estos cambios, cuál parecía ser el que más información daría sobre el target. En este punto hubo un problema con la librería de Seaborn a la hora de generar un mapa de calor con esta matriz, por lo que únicamente se muestran en la Figura 3 las relaciones de cada atributo con el target "year".

```
Correlación con year:
year          1.000000
top genre     0.333405
pop           0.249841
acous         0.102411
dnce          0.085272
spch          0.005789
bpm          -0.101932
val          -0.120510
live         -0.135352
dB           -0.179094
dur          -0.215458
nrgy         -0.225296
Name: year, dtype: float64
```

Fig 3. Correlación de cada atributo con year.

El atributo que más información da es "top genre" lo cual es de esperar ya que se le aplicó mean encoding, y el siguiente más relevante parece ser "pop". Por el lado contrario, el atributo que menos información da al respecto parece ser "spch".

Esto fue lo último antes de pasar a la elección de métrica y de modelo.

3 EXPERIMENTOS, RESULTADOS I ANÁLISIS

Como métrica se descartó usar accuracy por tratarse de un problema multiclase y encima desbalanceado, y como no había motivos para priorizar la precision por encima del recall y viceversa, decidí utilizar F1-Score.

El primer paso fue realizar un entrenamiento simple con un modelo de regresor logístico para ver los resultados que obtendría generando curvas Precision-Recall. Pero como se ve en la Figura 4, el resultado deja mucho que desear exceptuando la curva del año 2019 por algún motivo que desconozco.

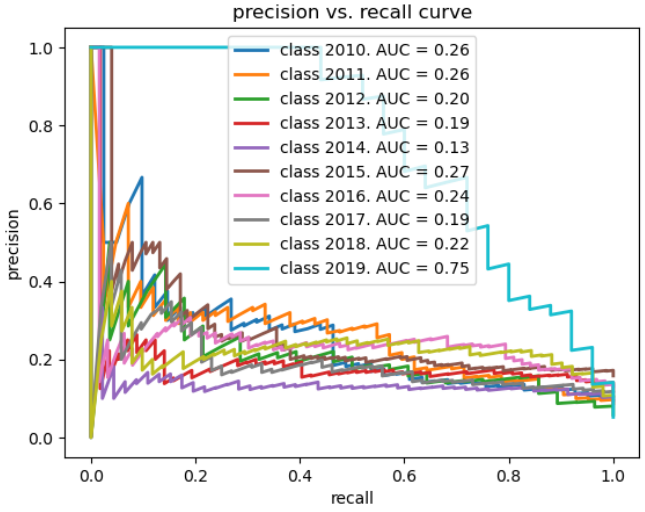

Fig 4. Precision-Recall curve de cada año con Logistic Regression.

Los resultados generados por el classification_report en la Figura 5 en este punto son malos, pero aun así decidí continuar con otros modelos algo más complejos y, para tener algún tipo de referente utilicé un clasificador Dummy que dio los resultados de la Figura 6. Para todas las validaciones de aquí en adelante se utiliza cross-validation de 10 folds estratificado y con shuffle.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	41
1	0.60	0.07	0.13	42
2	0.00	0.00	0.00	28
3	1.00	0.02	0.03	57
4	0.00	0.00	0.00	46
5	0.00	0.00	0.00	76
6	1.00	0.02	0.03	63
7	0.00	0.00	0.00	52
8	0.00	0.00	0.00	51
9	1.00	0.44	0.61	25
micro avg	0.84	0.03	0.06	481
macro avg	0.36	0.05	0.08	481
weighted avg	0.35	0.03	0.05	481
samples avg	0.03	0.03	0.03	481

Fig 5. Resultados del regresor logístico con classification_report

```
DummyClassifier mean f1-score: 0.027274550011392118
```

Fig 6. F1-Score del DummyClassifier

A continuación se mencionan los modelos utilizados en el entrenamiento y un breve motivo por el que se escogieron:

- Support Vector Machine (SVM). Aunque su forma

más simple no se pueda utilizar para problemas multiclase, con estrategias ovr o ovo se puede utilizar para este caso.
- KNN ya es capaz de manejar problemas de clasificación multiclase.
- Random Forest sin utilizar un Decision Tree, ya que es un modelo más complejo que puede evitar overfitting y producir mejores resultados aunque a un mayor costo computacional.
- Gradient Boosting como otro modelo ensemble aparte del Random Forest para tener un mínimo de variedad en las técnicas utilizadas con modelos ensemble.

Primero se hizo un entrenamiento y validación para cada uno de los modelos para ver su desempeño con los valores por defecto (Figura 7) y luego se aplicó una búsqueda de hiperparámetros con GridSearchCV (Tabla 1).

```
SVC mean f1-score: 0.154779612348218
KNeighborsClassifier mean f1-score: 0.12530078744784628
RandomForestClassifier mean f1-score: 0.20216567282047157
GradientBoostingClassifier mean f1-score: 0.16780158956148322
```

Fig 7. F1-Score de los distintos modelos con los valores por defecto.

	Classifier	F1-Score (Train)
0	SVC	0.197210
1	KNN	0.149058
2	Random Forest	0.149058
3	Gradient Boosting	0.187277

Tabla 1. F1-Score obtenidos del mejor modelo encontrado con GridSearchCV. La columna con los hiperparámetros no se muestra por motivos de espacio.

Como comentario adicional, en la búsqueda de los hiperparámetros se introdujeron los valores por defecto de todos los modelos, pero los resultados del Random Forest fueron inferiores a los del modelo por defecto inicial.

El mejor resultado se obtuvo del Random Forest por defecto con un F1-Score de 0,2. Sigue siendo un resultado bastante bajo pero tratándose de un problema multiclase y teniendo en cuenta que el DummyClassifier dio un resultado de 0,02, no es un resultado del todo malo.

Una vez obtenido tanto el modelo como definida la métrica, entrené el modelo para predecir el conjunto de test de una división entre train y test aleatoria y el resultado se muestra en la Figura 8.

```
F1-Score (Train): 0.17302
F1-Score (Test): 0.20572
```

Fig 8. F1-Score del Random Forest sobre el conjunto de Train arriba y del conjunto de Test abajo.

Como último detalle, se hizo una nueva Precisión-Recall curve con el Random Forest (Figura 9), pero esta vez aplicando las predicciones al conjunto de test. Se puede apreciar una mejora visual y una leve mejora en algunas AUC calculadas como es el caso de la de 2010 y 2012.

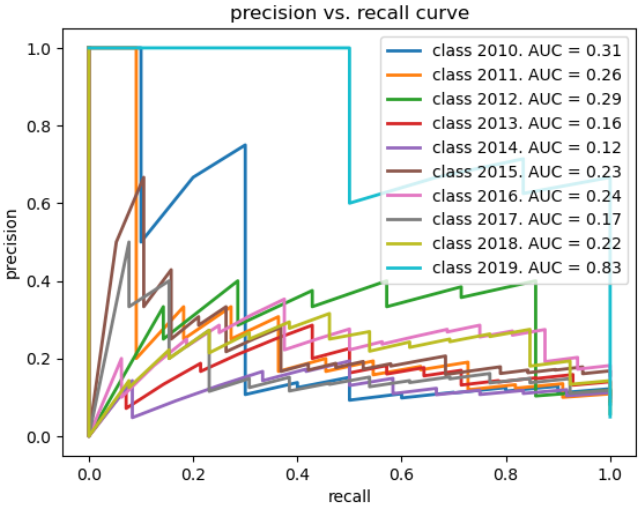

Fig 9. Precision-Recall curve de cada año con Random Forest.

4 CONCLUSIONES

En conclusión, los resultados obtenidos no son suficientemente buenos como para ser fiables al poner el modelo en práctica, y esto es debido a la dificultad del dataset y la propuesta de entrenamiento.

El dataset y la idea propuesta implican una clasificación multiclase, sin embargo los datos del dataset son muy escasos para un problema de este estilo. Únicamente 603 filas para 10 clases distintas es una muestra muy pobre, correspondiendo unas 60 filas a cada clase del target si fuese balanceado. A esto se le añade que no es balanceado, y la clase con menos muestras contiene únicamente 31 datos.

Si contamos la división entre train y test estratificada del 0,2 que se ha aplicado, esa clase se queda con unas 24 muestras. Si además se le añade el cross validation con 10 folds también estratificado, la clase se queda con 19 ejemplos en cada entrenamiento.

Esta cantidad no es suficientemente grande como para ser representativa, y eso se puede aplicar a las demás clases, aunque dispongan de más muestras.

Por lo que, aunque el dataset esté limpio y no haya que aplicar mucho preprocesamiento para poder empezar a entrenar un modelo, no es suficiente para un problema de este estilo.

Una posible solución que queda como idea para otro posible intento sería dividir el target en intervalos para así aumentar la probabilidad de acierto del modelo, o directamente plantear otro tipo de problema sobre este dataset.