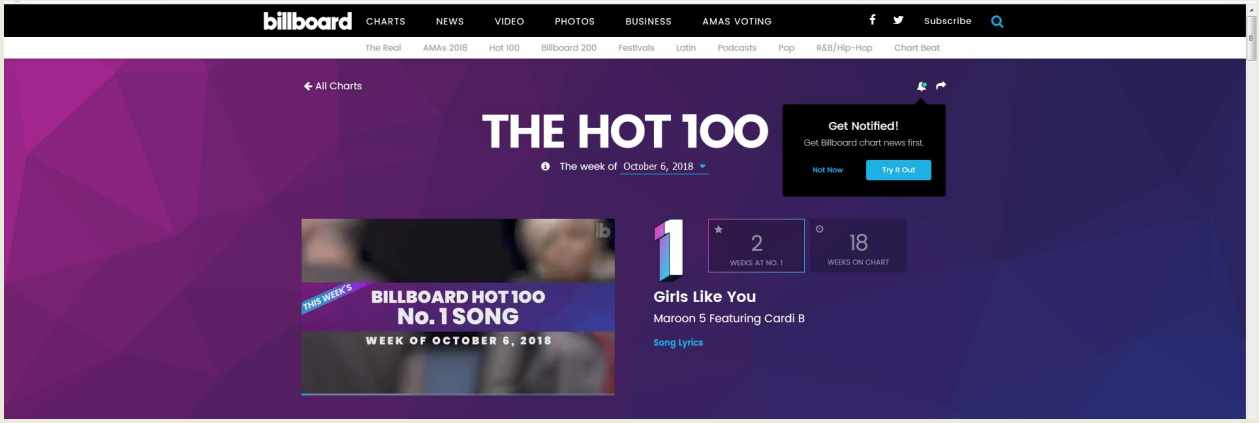
ML_CLASIFICACION_TREE_02	Árbol de Decisión para predecir si un artista de música logra alcanzar el Nº1 en la lista BillBoard 100	ML

Billboard Hot 100 es una gran lista de popularidad de los 100 sencillos más vendidos en los Estados Unidos, que ayuda a promover la industria musical nacional e internacional.



https://www.billboard.com/charts/hot-100

Para realizar esta práctica se utiliza la base de datos "**Base_Artistas_Billboard.csv**" con un histórico de 635 canciones de artistas que han entrado en la lista Billboard 100, y que en algunos casos han alcanzado el número 1 (top=1). Esta base contiene información del título de la canción, artista, estado de ánimo, tiempo, género, tipo de artista, fecha, duración en segundos, …, así como si la canción llegó al Nº1 (columna 'top', que es la etiqueta/target). Además, se ha codificado alguna información con el fin de agrupar en diferentes rangos y utilizarlo en esta práctica.

El objetivo es construir un modelo de aprendizaje maquina (machine learning), basado en el algoritmo de Árbol de Decisión que aprenda a predecir si la canción de un artista logra llegar al número 1 del BillBoard 100.

Se aplica también el árbol de decisión para predecir si dos casos concretos llegan al Nº1:

Canción "Habanna" de Camila Cabello

Canción "Believer" de Imaging Dragons

SOLUCIÓN

Importar las librerías necesarias para realizar la práctica.

```
# Librerías
import numpy as np
import pandas as pd
from sklearn import tree
```

Cargar la base de datos:

```
# Cargar base de datos de artistas de música
artists_billboard = pd.read_csv("Base_Artistas_Billboard.csv")
```

Seleccionar la parte de los datos codificados:

```
# Seleccionamos la parte de los datos codificados (encoded)
datos=artists_billboard.iloc[:,10:]
```

Generar los datos X e y para entrenamiento.

```
# Definir las variables de entrenamiento
y_train = datos['top']
X_train = datos.drop(['top','anioNacimiento','edad_en_billboard'],
axis=1).values
```

Crear el árbol de decisión con los parámetros:

- **criterion=entropy** ó podría ser gini, pero utilizamos entradas categóricas
- **min_samples_split=20** se refiere a la cantidad mínima de muestras que debe tener un nodo para poder subdividir.
- **min_samples_leaf=5** cantidad mínima que puede tener una hoja final. Si tuviera menos, no se formaría esa hoja y "subiría" un nivel, su antecesor.
- **class_weight={1:3.5}** *IMPORTANTíSIMO*: con esto compensamos los desbalances que hubiera. En nuestro caso, tenemos menos etiquetas de tipo top=1 (los artistas que llegaron al número 1 del ranking). Por lo tanto, le asignamos 3.5 de peso a la etiqueta 1 para compensar. El valor sale de dividir la cantidad de top=0 con los top=1.

```
# Crear Arbol de decision con profundidad 'depth=4'
depth=4
decision_tree = tree.DecisionTreeClassifier(criterion='entropy',
                                            min_samples_split=20,
                                            min_samples_leaf=5,
                                            max_depth = depth,
                                            class_weight={1:3.5})
```

Ajustar el modelo

```
# Entrenar el modelo
```

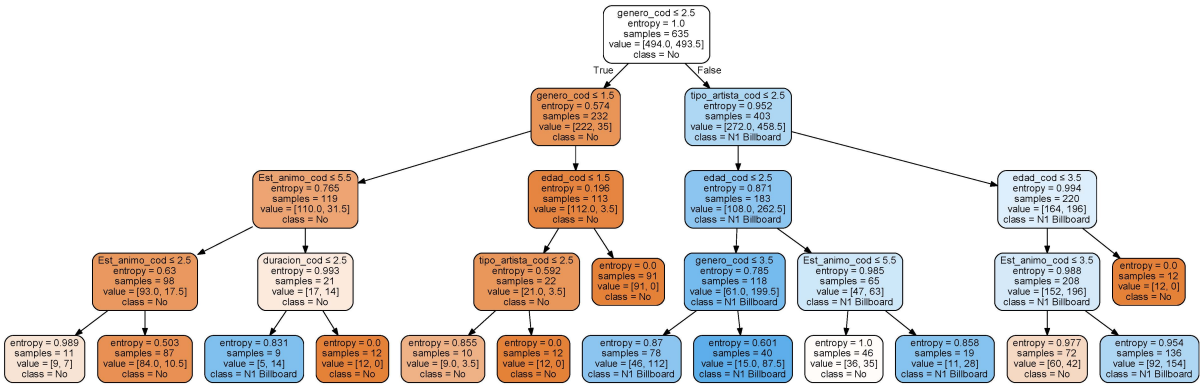
```
decision_tree.fit(X_train, y_train)
```

Generar el gráfico del árbol.
Previamente se instala la librería graphviz (Terminal: conda install python-graphviz)

```
# Generar el gráfico
dot_data = tree.export_graphviz(decision_tree, out_file=None,

feature_names=list(datos.drop(['top','anioNacimiento','edad_en_billboard'],
axis=1)),
                                class_names=['No', 'N1 Billboard'],
                                filled=True, rounded=True,
                                special_characters=True)

# Exportar el gráfico a formato PDF
graph = graphviz.Source(dot_data)
graph.render("Arbol_N1_Billboard")
```



Evaluar el modelo:

```
# EVALUACIÓN. Precisión alcanzada por el árbol
acc_decision_tree = np.round(decision_tree.score(X_train, y_train) * 100, 2)
print("Precisión del Árbol de Decisión: ",acc_decision_tree)
print("")

Precisión del Árbol de Decisión:  68.98
```

Predecir para los dos ejemplos

```
# Probar el árbol con 2 artistas que entraron al billboard 100 en 2017:
# Camila Cabello que llegó al numero 1 con la Canción Havana
# e Imagine Dragons con su canción Believer que alcanzó un puesto 42 pero no
```

```
llegó a la cima

# Camila Cabello con su canción Havana llego a numero 1 Billboard US en 2017
x_test = pd.DataFrame(columns=('top','Est_animo_cod', 'tiempo_cod',
'genero_cod','tipo_artista_cod','edad_cod','duracion_cod'))
x_test.loc[0] = (1,5,2,4,1,0,3)
y_pred = decision_tree.predict(x_test.drop(['top'], axis = 1))

print("Predicción (Camila Cabello): " + str(y_pred))
y_proba = decision_tree.predict_proba(x_test.drop(['top'], axis = 1))
print("Probabilidad de Acierto: " + str(np.round(y_proba[0][y_pred]* 100,
2))+"%")

Predicción (Camila Cabello): [1]
Probabilidad de Acierto: [85.37]%


# predecir artista Imagine Dragons
# con su canción Believer llego al puesto 42 Billboard US en 2017
x_test = pd.DataFrame(columns=('top','Est_animo_cod', 'tiempo_cod',
'genero_cod','tipo_artista_cod','edad_cod','duracion_cod'))
x_test.loc[0] = (0,4,2,1,3,2,3)
y_pred = decision_tree.predict(x_test.drop(['top'], axis = 1))
print("Predicción (Imaging Dragons): " + str(y_pred))
y_proba = decision_tree.predict_proba(x_test.drop(['top'], axis = 1))
print("Probabilidad de Acierto: " + str(np.round(y_proba[0][y_pred]* 100,
2))+"%")

Predicción (Imaging Dragons): [0]
Probabilidad de Acierto: [88.89]%
```