

Inteligencia Artificial
Act12: Árbol de Decisión

Arturo Garza Rodríguez

March 2025

1. Introducción

¿Qué es un árbol de decisión?

Un árbol de decisión es un algoritmo de aprendizaje supervisado no paramétrico que se utiliza tanto en tareas de clasificación como de regresión. Su estructura jerárquica, similar a un árbol, consta de un nodo raíz, ramas, nodos internos y nodos de hoja. Este modelo permite representar de manera visual y comprensible las posibles decisiones y sus consecuencias, comparando diferentes acciones según sus costos, probabilidades y beneficios. De esta manera, puede ser usado para trazar algoritmos matemáticos que anticipen la mejor opción o para dirigir intercambios de ideas informales en un proceso de toma de decisiones.

2. Metodología

2.1. Requerimientos

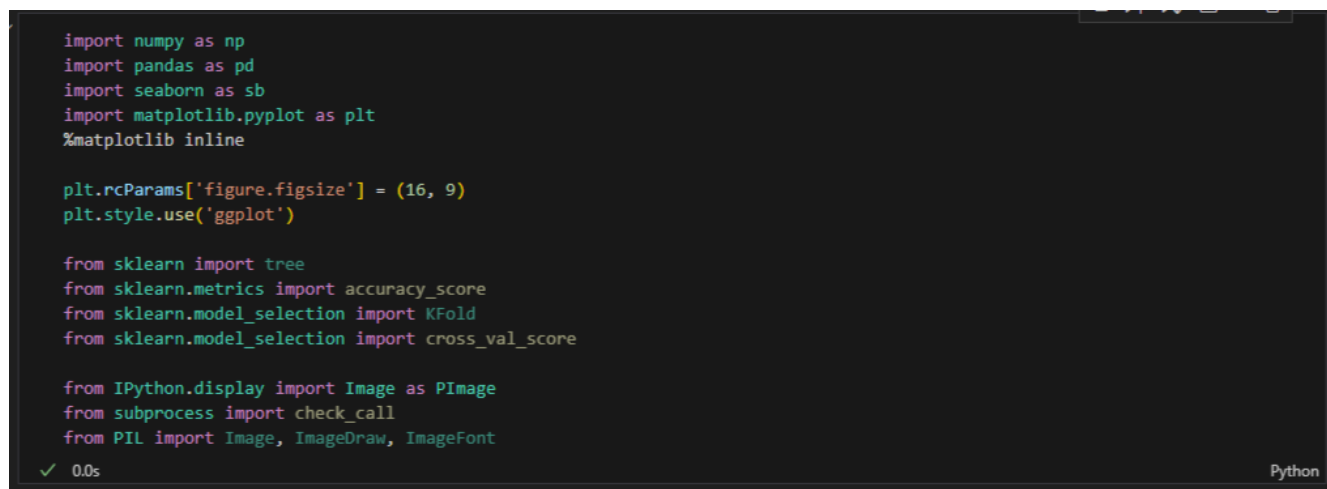
2.1.1. Jupyter notebook y dataset

En este trabajo estaremos utilizando un entorno de una *Jupyter Notebook* para segmentar el código e ir tomando nota de los resultados con cada procedimiento.

En este caso usaremos el dataset 'artists_billboard_fix3.csv' que se nos proporciona en el apartado pertinente del libro.

2.2. Desarrollo de código

2.2.1. Imports



```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline

plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

from IPython.display import Image as PImage
from subprocess import check_call
from PIL import Image, ImageDraw, ImageFont
```

✓ 0.0s Python

Figura 1: Se realizan los imports de las librerías necesarias para configurar nuestro árbol de decisión.

2.2.2. Definición de dataset

```
artists_billboard = pd.read_csv('artists_billboard_fix3.csv')
artists_billboard.shape
artists_billboard.head()
```

[4] ✓ 0.0s Python

	id	title	artist	mood	tempo	genre	artist_type	chart_date	durationSeg	top	anioNacimiento
0	0	Small Town Throwdown	BRANTLEY GILBERT featuring JUSTIN MOORE & THOM...	Brooding	Medium Tempo	Traditional	Male	20140628	191.0	0	1975.0
1	1	Bang Bang	JESSIE J, ARIANA GRANDE & NICKI MINAJ	Energizing	Medium Tempo	Pop	Female	20140816	368.0	0	1989.0
2	2	Timber	PITBULL featuring KE\$HA	Excited	Medium Tempo	Urban	Mixed	20140118	223.0	1	1993.0
3	3	Sweater Weather	THE NEIGHBOURHOOD	Brooding	Medium Tempo	Alternative & Punk	Male	20140104	206.0	0	1989.0
4	4	Automatic	MIRANDA LAMBERT	Yearning	Medium Tempo	Traditional	Female	20140301	232.0	0	0.0

```
artists_billboard.groupby('top').size()
```

[5] ✓ 0.0s Python

```
top
0    494
1    141
dtype: int64
```

Spaces: 4 Cell 1 of 30

Figura 2: Se lee el dataset usando la librería pandas y se imprimen los primeros 5 registros, para ver el formato del dataset, a través de `data.head()`.

Asimismo, se imprimen las dimensiones de la agrupación en base a 'top'.

2.2.3. Histogramas para análisis de datos

Los histogramas fueron generados a través de `sb.catplot()`.

```
# ***factorplot has been deprecated and removed from Seaborn as of version 0.11.0***
sb.catplot(x='artist_type', data=artists_billboard, kind="count")
```

✓ 0.2s Python

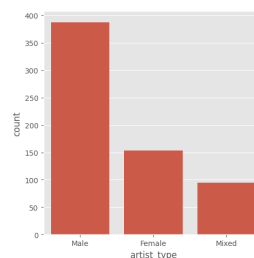


Figura 3: Aquí vemos que tenemos más del doble de artistas masculinos que femeninos y unos 100 registros de canciones mixtas.

```
sb.catplot(x='mood',data=artists_billboard,kind="count", aspect=3)
```

✓ 0.4s Python

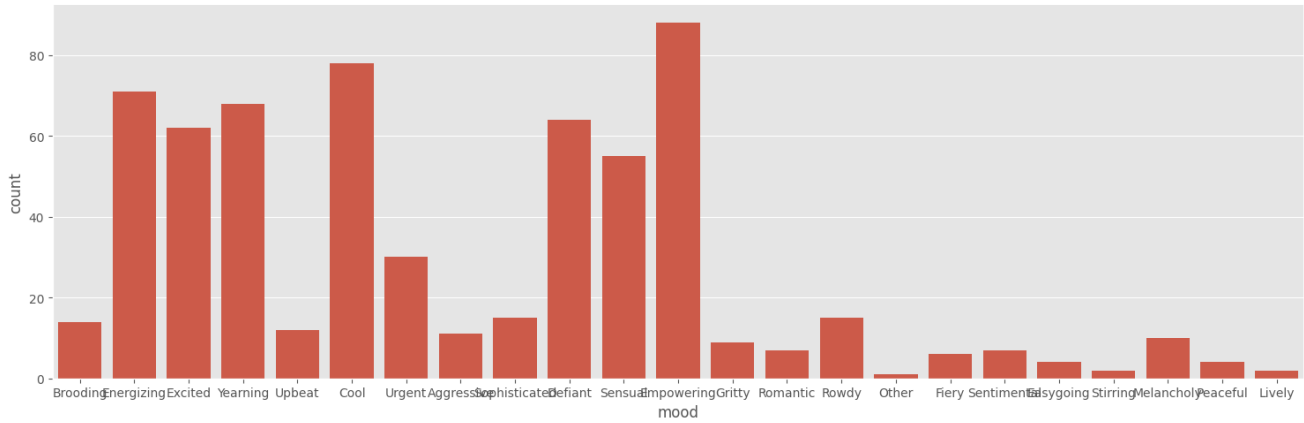


Figura 4: Vemos que de 23 tipos de Mood, destacan 7 con picos altos. Además notamos que algunos estados de ánimo son similares.

```
sb.catplot(x='genre', data=artists_billboard, kind="count", aspect=3)
```

✓ 0.2s Python

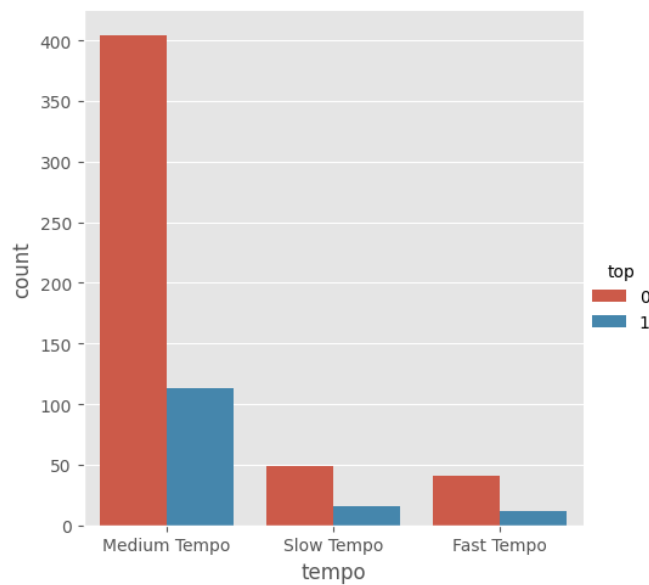


Figura 5: En esta gráfica vemos que hay 3 tipos de Tempo: Medium, Slow y Fast. Evidentemente predominan los tiempos Medium y también es donde encontramos más canciones que hayan alcanzado el Top 1 (en azul)

```
sb.catplot(x='genre', data=artists_billboard, kind="count", aspect=3)
```

Python

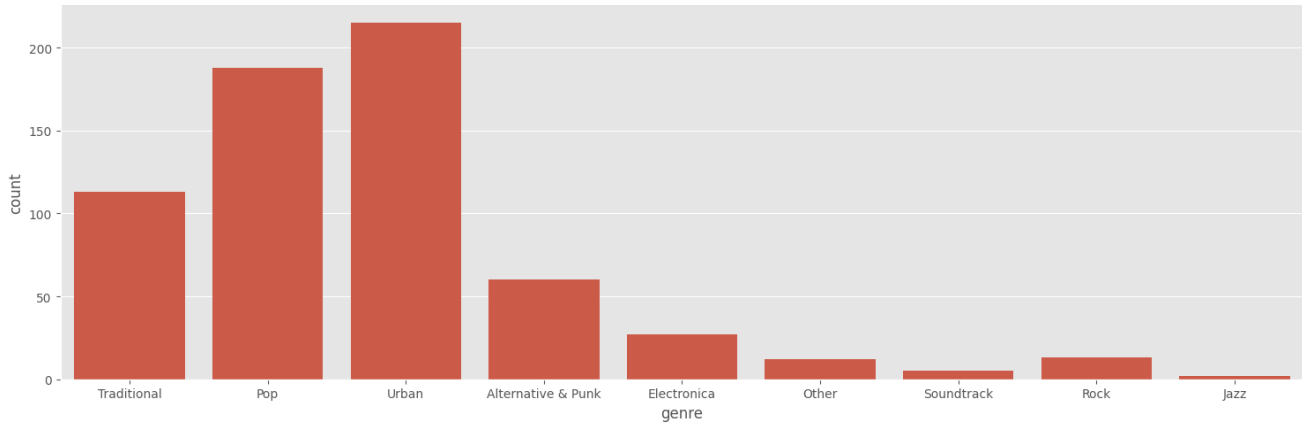


Figura 6: Entre los géneros musicales destacan Urban y Pop, seguidos de Tradicional.

```
sb.catplot(x='anioNacimiento', data=artists_billboard, kind="count", aspect=3)
```

Python

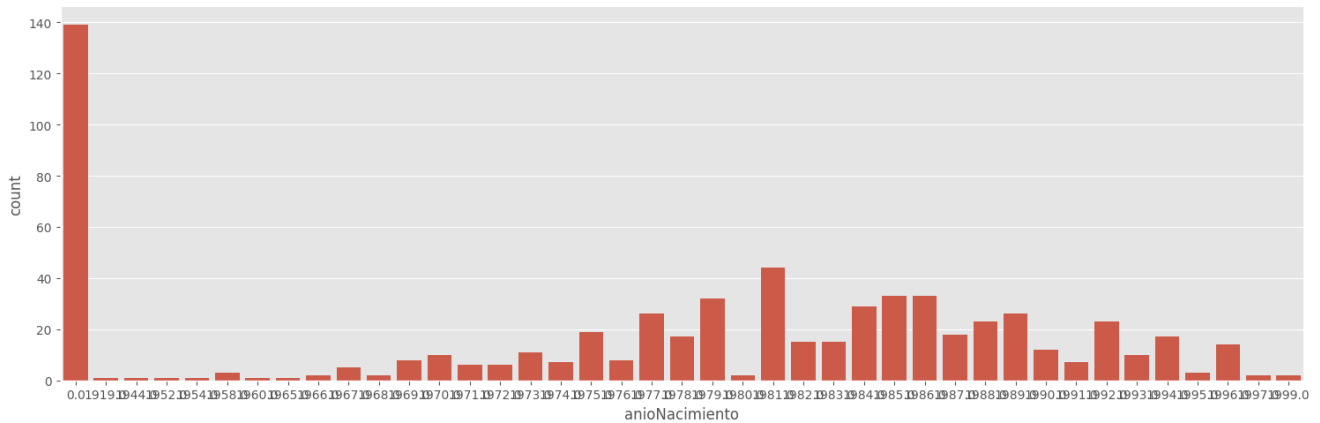


Figura 7: Como se ve en la gráfica tenemos cerca de 140 canciones de las cuales desconocemos el año de nacimiento del artista.

2.2.4. Preparación de datos

```
f1 = artists_billboard['chart_date'].values
f2 = artists_billboard['durationSeg'].values

colores = ['orange', 'blue']
tamanios = [60, 40]

asignar = []
asignar2 = []

for index, row in artists_billboard.iterrows():
    asignar.append(colores[row['top']])
    asignar2.append(tamanios[row['top']])

plt.scatter(f1, f2, c=asignar, s=asignar2)

plt.axis([20030101, 20160101, 0, 600])

plt.show()
```

✓ 0.1s Python

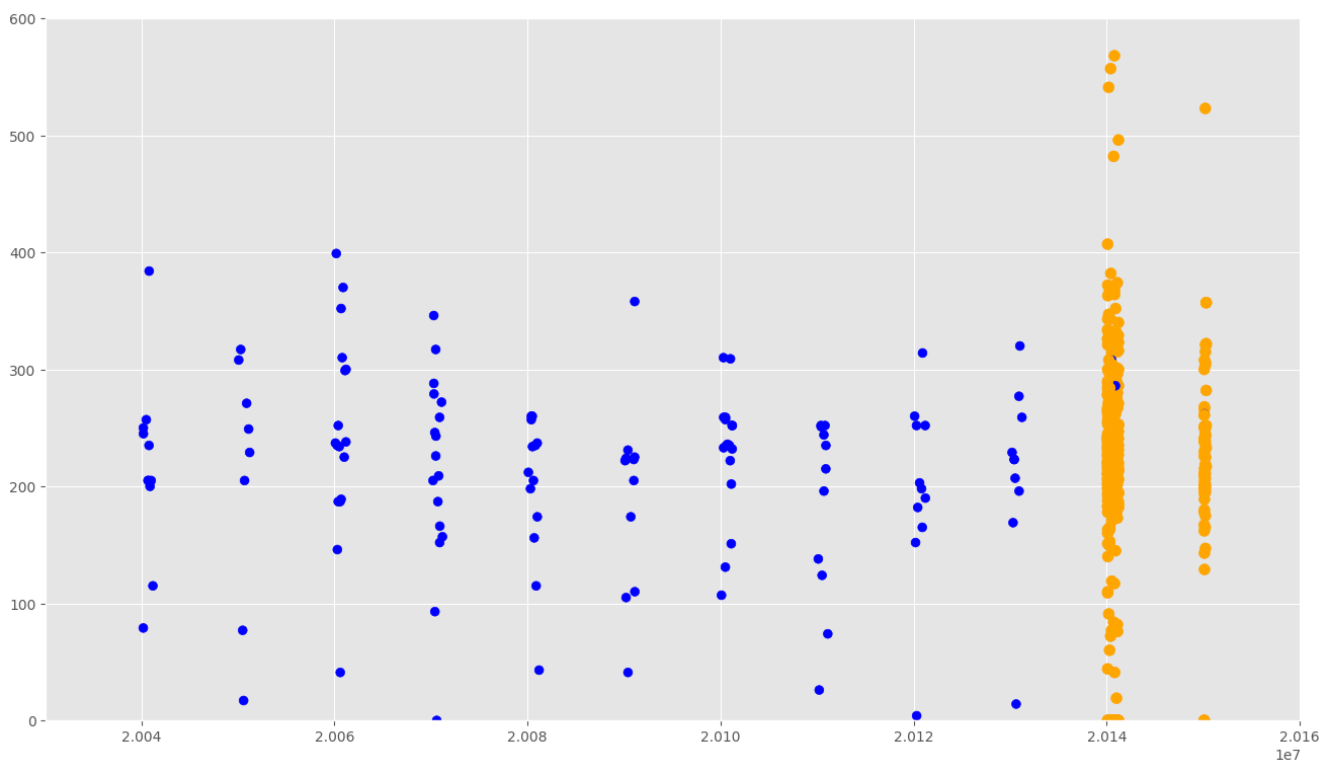


Figura 8: En nuestro conjunto de Datos, se agregaron canciones que llegaron al top (en azul) de años 2004 al 2013 para sumar a los apenas 11 que lo habían logrado en 2014-2015.

```
def edad_fix(anio):
    if anio == 0:
        return None
    return anio

artists_billboard['anioNacimiento'] = artists_billboard.apply(lambda x: edad_fix(x['anioNacimiento']), axis=1)
```

✓ 0.0s

Python

```
def calcula_edad(anio, cuando):
    cad = str(cuando)
    momento = cad[:4]

    if anio == 0.0:
        return None

    return int(momento) - anio

artists_billboard['edad_en_billboard'] = artists_billboard.apply(lambda x:
    calcula_edad(x['anioNacimiento'], x['chart_date']), axis=1)
```

✓ 0.0s

Python

```
age_avg = artists_billboard['edad_en_billboard'].mean()
age_std = artists_billboard['edad_en_billboard'].std()
age_null_count = artists_billboard['edad_en_billboard'].isnull().sum()

age_null_random_list = np.random.randint(age_avg - age_std, age_avg + age_std, size=age_null_count)

conValoresNulos = np.isnan(artists_billboard['edad_en_billboard'])

artists_billboard.loc[conValoresNulos, 'edad_en_billboard'] = age_null_random_list

artists_billboard['edad_en_billboard'] = artists_billboard['edad_en_billboard'].astype(int)

print("Edad Promedio: " + str(age_avg))
print("Desviación Estándar de Edad: " + str(age_std))
print("Intervalo para asignar edad aleatoria: " + str(int(age_avg - age_std)) + " a " + str(int(age_avg + age_std)))
```

✓ 0.0s

Python

Edad Promedio: 30.10282258064516
Desviación Estándar de Edad: 8.40078832861513
Intervalo para asignar edad aleatoria: 21 a 38

```

f1 = artists_billboard['edad_en_billboard'].values
f2 = artists_billboard.index

colores = ['orange', 'blue', 'green']

asignar = []

for index, row in artists_billboard.iterrows():
    if conValoresNulos[index]:
        asignar.append(colores[2])
    else:
        asignar.append(colores[row['top']])

plt.scatter(f1, f2, c=asignar, s=30)

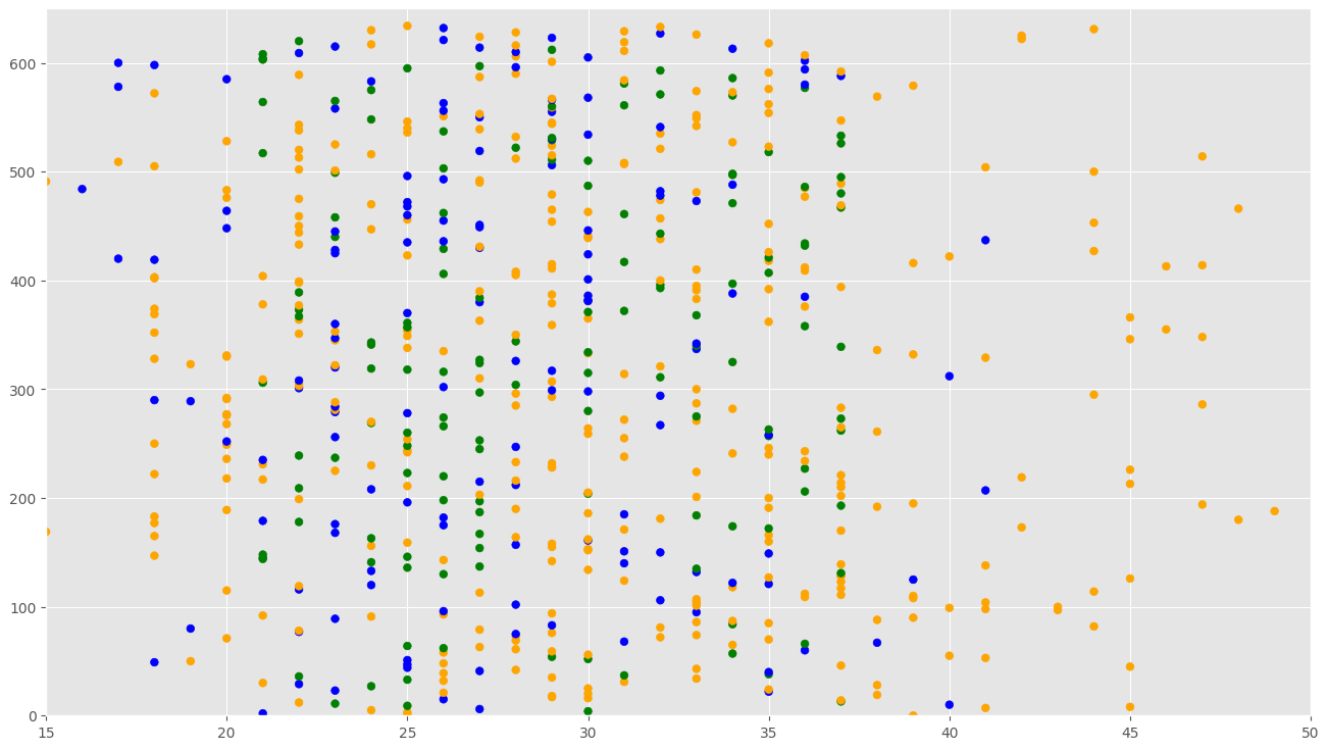
plt.axis([15, 50, 0, 650])

plt.show()

```

✓ 0.2s

Python



2.2.5. Mapeo de datos

MoodEncoded

```
# Mood Encoding
artists_billboard['moodEncoded'] = artists_billboard['mood'].map({
    'Energizing': 6,
    'Empowering': 6,
    'Cool': 5,
    'Yearning': 4, # anhelo, deseo, ansia
    'Excited': 5, # emocionado
    'Defiant': 3,
    'Sensual': 2,
    'Gritty': 3, # coraje
    'Sophisticated': 4,
    'Aggressive': 4, # provocativo
    'Fiery': 4, # caracter fuerte
    'Urgent': 3,
    'Rowdy': 4, # ruidoso alboroto
    'Sentimental': 4,
    'Easygoing': 1, # sencillo
    'Melancholy': 4,
    'Romantic': 2,
    'Peaceful': 1,
    'Brooding': 4, # melancólico
    'Upbeat': 5, # optimista alegre
    'Stirring': 5, # emocionante
    'Lively': 5, # animado
    'Other': 0,
    '': 0
}).astype(int)
```

Tempo and Genre Mapping

```
# Tempo Encoding
artists_billboard['tempoEncoded'] = artists_billboard['tempo'].map({
    'FastTempo': 0,
    'MediumTempo': 2,
    'SlowTempo': 1,
    '': 0
}).fillna(0).astype(int)
```

✓ 0.0s

Python

```
# Genre Encoding
artists_billboard['genreEncoded'] = artists_billboard['genre'].map({
    'Urban': 4,
    'Pop': 3,
    'Traditional': 2,
    'Alternative&Punk': 1,
    'Electronica': 1,
    'Rock': 1,
    'Soundtrack': 0,
    'Jazz': 0,
    'Other': 0,
    '': 0
}).fillna(0).astype(int)
```

✓ 0.0s

Python

Other mapping

```
# Artist Type Encoding
artists_billboard['artist_typeEncoded'] = artists_billboard['artist_type'].map({
    'Female': 2,
    'Male': 3,
    'Mixed': 1,
    '': 0
}).astype(int)
```

✓ 0.0s

Python

```
# Age Encoding based on the age when they arrived on the Billboard
artists_billboard.loc[artists_billboard['edad_en_billboard'] <= 21, 'edadEncoded'] = 0
artists_billboard.loc[(artists_billboard['edad_en_billboard'] > 21) & (artists_billboard['edad_en_billboard'] <= 26), 'edadEncoded'] = 1
artists_billboard.loc[(artists_billboard['edad_en_billboard'] > 26) & (artists_billboard['edad_en_billboard'] <= 30), 'edadEncoded'] = 2
artists_billboard.loc[(artists_billboard['edad_en_billboard'] > 30) & (artists_billboard['edad_en_billboard'] <= 40), 'edadEncoded'] = 3
artists_billboard.loc[artists_billboard['edad_en_billboard'] > 40, 'edadEncoded'] = 4
```

✓ 0.0s

Python

```
# Song Duration Encoding
artists_billboard.loc[artists_billboard['durationSeg'] <= 150, 'durationEncoded'] = 0
artists_billboard.loc[(artists_billboard['durationSeg'] > 150) & (artists_billboard['durationSeg'] <= 180), 'durationEncoded'] = 1
artists_billboard.loc[(artists_billboard['durationSeg'] > 180) & (artists_billboard['durationSeg'] <= 210), 'durationEncoded'] = 2
artists_billboard.loc[(artists_billboard['durationSeg'] > 210) & (artists_billboard['durationSeg'] <= 240), 'durationEncoded'] = 3
artists_billboard.loc[(artists_billboard['durationSeg'] > 240) & (artists_billboard['durationSeg'] <= 270), 'durationEncoded'] = 4
artists_billboard.loc[(artists_billboard['durationSeg'] > 270) & (artists_billboard['durationSeg'] <= 300), 'durationEncoded'] = 5
artists_billboard.loc[artists_billboard['durationSeg'] > 300, 'durationEncoded'] = 6
```

3. Resultados

3.0.1. Como quedan los top en relación a los datos mapeados

moodEncoded

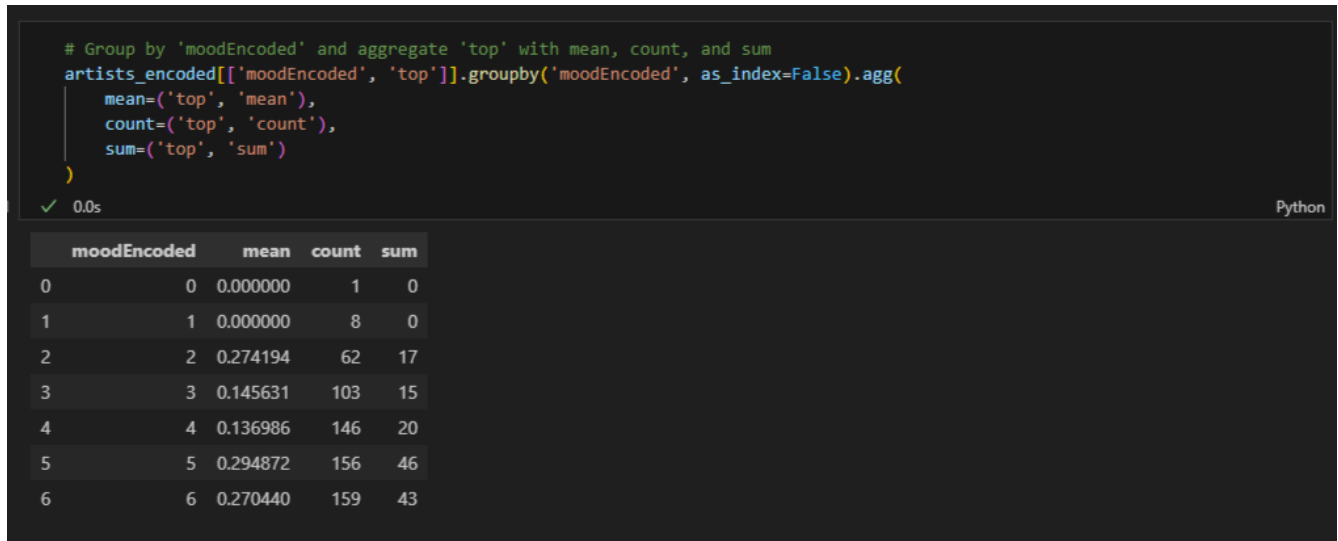


Figura 9: La mayoría de top 1 los vemos en los estados de Ánimo 5 y 6 con 46 y 43 canciones

artist_typeEncoded

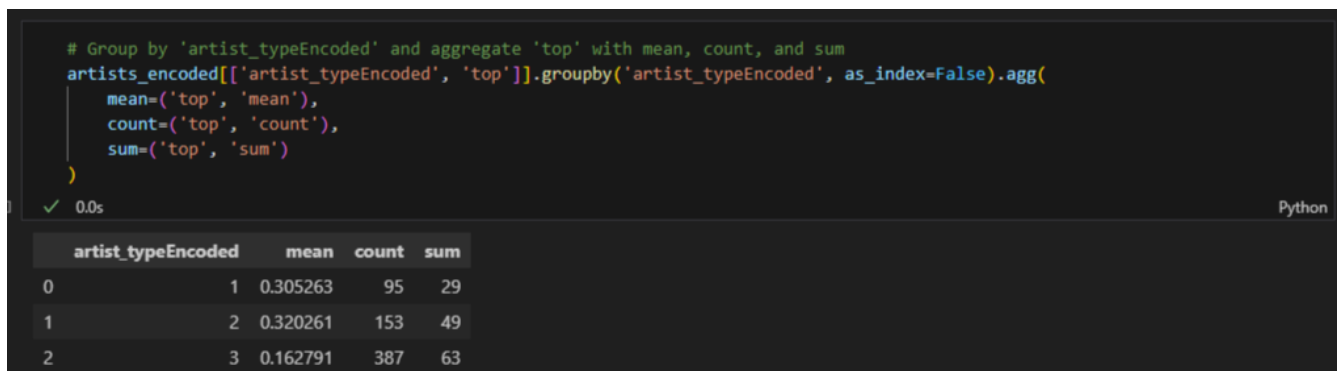


Figura 10: Aquí están bastante repartidos, pero hay mayoría en tipo 3: artistas masculinos

genreEncoded

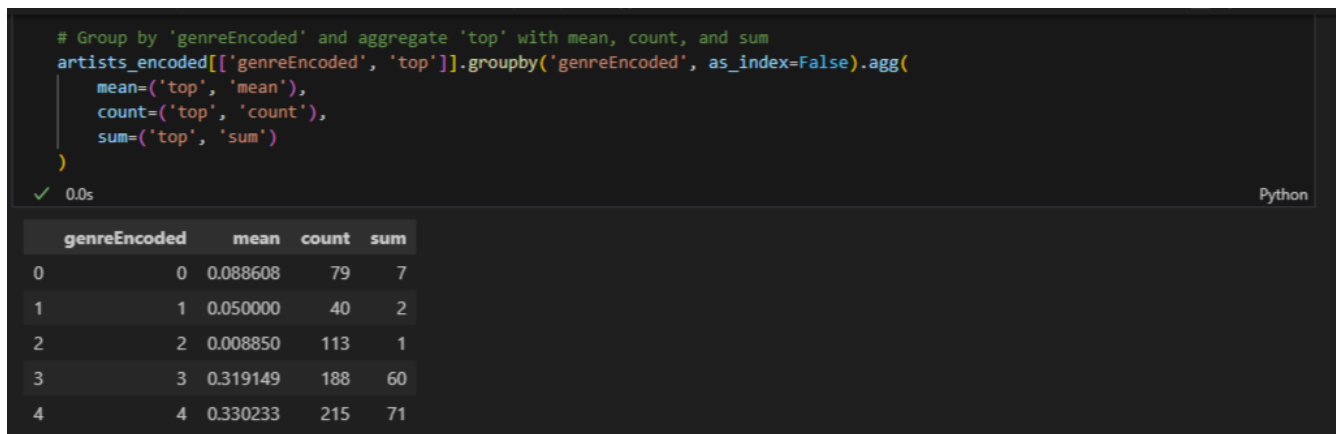


Figura 11: Los géneros con mayoría son evidentemente los géneros 3 y 4 que corresponden con Urbano y Pop

edadEncoded

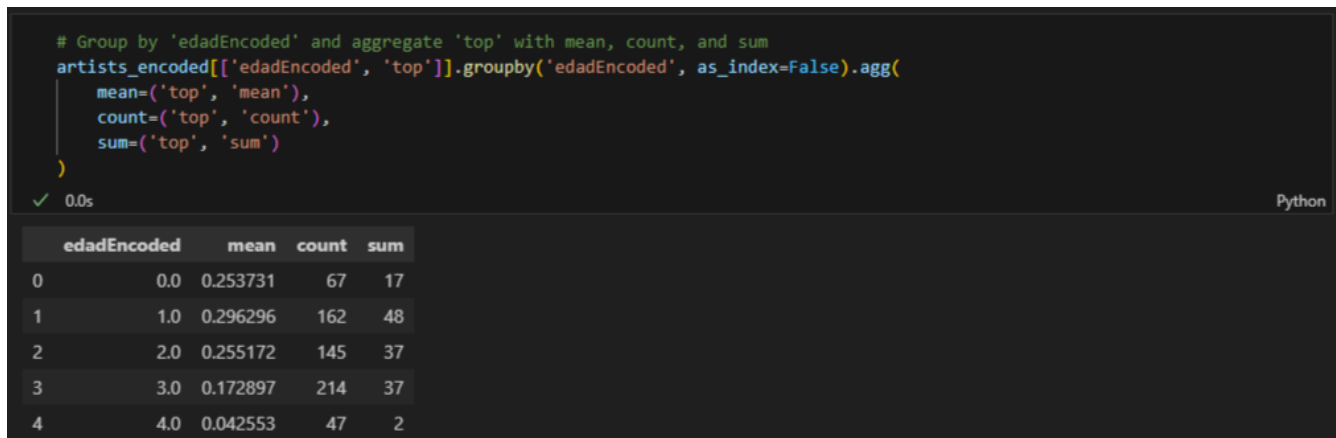


Figura 12: Edad con mayoría es la tipo 1 que comprende de 21 a 25 años

4. Creación de árbol

```
cv = KFold(n_splits=10)
accuracies = list()
max_attributes = len(list(artists_encoded))
depth_range = range(1, max_attributes + 1)

for depth in depth_range:
    fold_accuracy = []
    tree_model = tree.DecisionTreeClassifier(
        criterion='entropy',
        min_samples_split=20,
        min_samples_leaf=5,
        max_depth=depth,
        class_weight={1: 3.5}
    )

    for train_fold, valid_fold in cv.split(artists_encoded):
        f_train = artists_encoded.loc[train_fold]
        f_valid = artists_encoded.loc[valid_fold]

        model = tree_model.fit(X=f_train.drop(['top'], axis=1), y=f_train['top'])

        valid_acc = model.score(X=f_valid.drop(['top'], axis=1), y=f_valid['top'])
        fold_accuracy.append(valid_acc)

    avg = sum(fold_accuracy) / len(fold_accuracy)
    accuracies.append(avg)

df = pd.DataFrame({"MaxDepth": depth_range, "AverageAccuracy": accuracies})
df = df[["MaxDepth", "AverageAccuracy"]]
print(df.to_string(index=False))
```

Max Depth	Average Accuracy
1	0.556101
2	0.556126
3	0.564038
4	0.648859
5	0.617386
6	0.614236
7	0.625124

Podemos ver que en 4 niveles de splits tenemos el score más alto, con casi 65%. Ahora ya sólo nos queda crear y visualizar nuestro árbol de 4 niveles de profundidad.

```

y_train = artists_encoded['top']
x_train = artists_encoded.drop(['top'], axis=1).values

decision_tree = tree.DecisionTreeClassifier(
    criterion='entropy',
    min_samples_split=20,
    min_samples_leaf=5,
    max_depth=4,
    class_weight={1: 3.5}
)
decision_tree.fit(x_train, y_train)
dot_path = "tree1.dot"
with open(dot_path, 'w') as f:
    tree.export_graphviz(
        decision_tree,
        out_file=f,
        max_depth=7,
        impurity=True,
        feature_names=list(artists_encoded.drop(['top'], axis=1).columns),
        class_names=['No', 'N1Billboard'],
        rounded=True,
        filled=True
    )

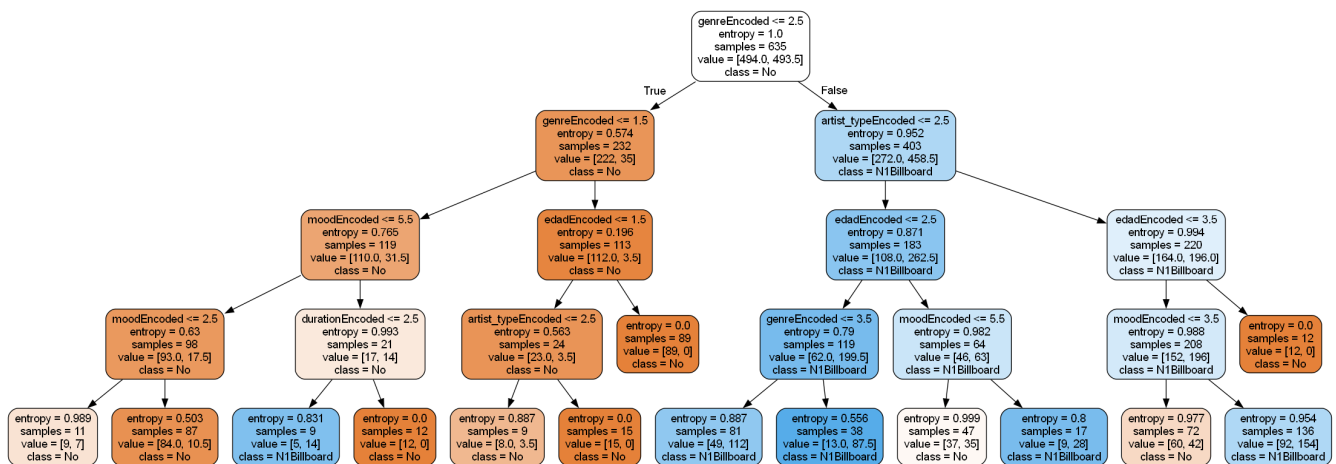
with open(dot_path) as f:
    dot_data = f.read()

graph = graphviz.Source(dot_data)
graph.render("tree1", format="png", cleanup=True)
PImage("tree1.png")

```

Figura 13: Se uso la libreria graphviz para visualizar el árbol.

Árbol



5. Conclusión

En este trabajo se utilizó un árbol de decisión para analizar un dataset de artistas musicales. A través de la visualización de datos y la preparación del conjunto, se identificaron patrones como la predominancia de artistas masculinos, géneros musicales urbanos y pop, así como la influencia del estado de ánimo y la edad en los éxitos musicales. Finalmente, el árbol de decisión permitió clasificar estos datos de manera clara, demostrando la utilidad de esta herramienta para comprender y predecir tendencias en datos complejos.

6. Referencias

- Material de clase
- <https://www.ibm.com/mx-es/think/topics/decision-trees>
- <https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-arbol-de-decision>