Actividad 12

Jose Manuel Enriquez

29 de marzo del 2025

1 Introduction

1.1 ¿Qué es un árbol de decisión?

Los árboles de decisión son uno de los algoritmos más fundamentales en el aprendizaje automático (machine learning), reconocidos por su simplicidad, interpretabilidad y versatilidad. Inspirados en procesos de toma de decisiones humanas, estos modelos dividen un problema complejo en una serie de preguntas jerárquicas, cuyas respuestas conducen a una solución predictiva. Su estructura intuitiva —similar a un diagrama de flujo— los hace accesibles incluso para quienes no son expertos en ciencia de datos, mientras que su eficacia los mantiene relevantes en aplicaciones que van desde diagnósticos médicos hasta análisis financieros.

La importancia de los árboles de decisión radica en su capacidad para:

Automatizar decisiones basadas en datos históricos.

Manejar tanto variables categóricas como numéricas sin requerir preprocesamiento exhaustivo.

Ser la base de modelos más avanzados como Random Forest y Gradient Boosting.

Ofrecer transparencia, algo crítico en sectores regulados (banca, salud) donde explicar las predicciones es tan importante como su precisión.

Aunque tienen limitaciones (como propensión al sobreajuste), su equilibrio entre rendimiento y claridad los convierte en una herramienta esencial para cualquier profesional de datos. En esta exploración, profundizaremos en su funcionamiento, aplicaciones y buenas prácticas para aprovechar su potencial.

Este ejercicio fue realizado en una Jupyter Notebook y Python 3.12.4 Lo primero que se hizo fue importar todas las librerías necesarias.

```
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    ModuleNotFoundError: No module named 'sklean'
from sklearn.model_selection import KFold, cross_val_score
from IPython.display import Image as PImage
from subprocess import check_call
from PIL import Image, ImageDraw, ImageFont
✓ 0.0s
```

Figure 1: Librerias

Luego importamos el archivos con los datos y vemos su contenido.

```
artists_billboard = pd.read_csv("artists_billboard_fix3.csv")
✓ 0.0s                                                                Python
```

```
artists_billboard.shape
✓ 0.0s                                                                Python
(635, 11)
```

```
artists_billboard.head()
✓ 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

Figure 2: Dataframe

Luego visualizamos los datos

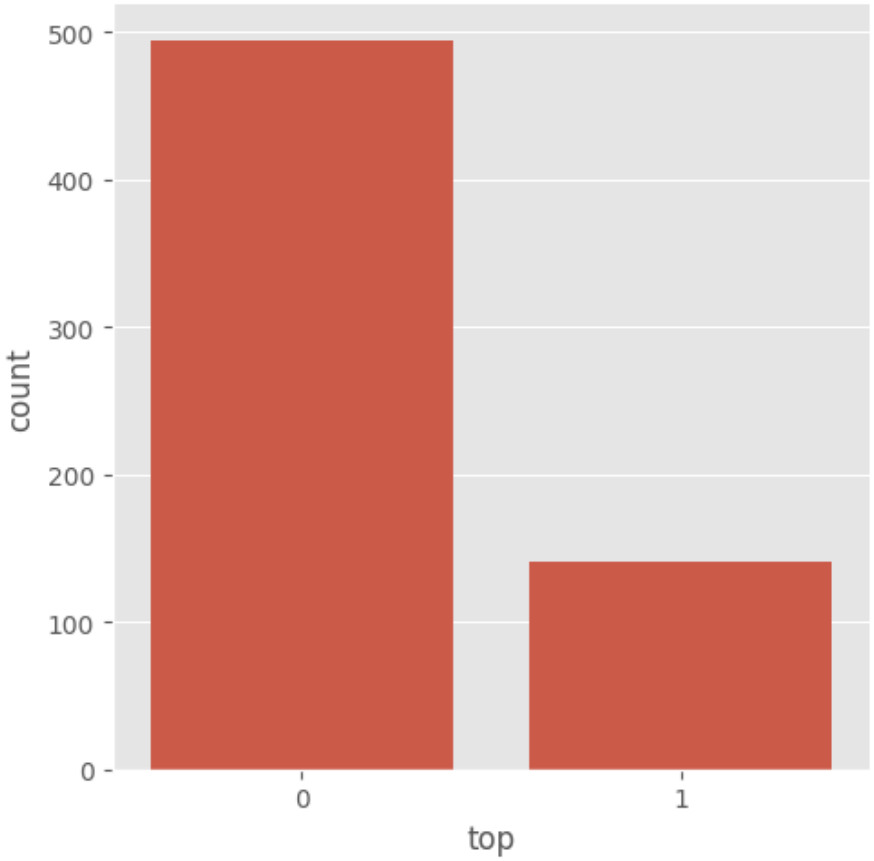

Figure 3: Top 1

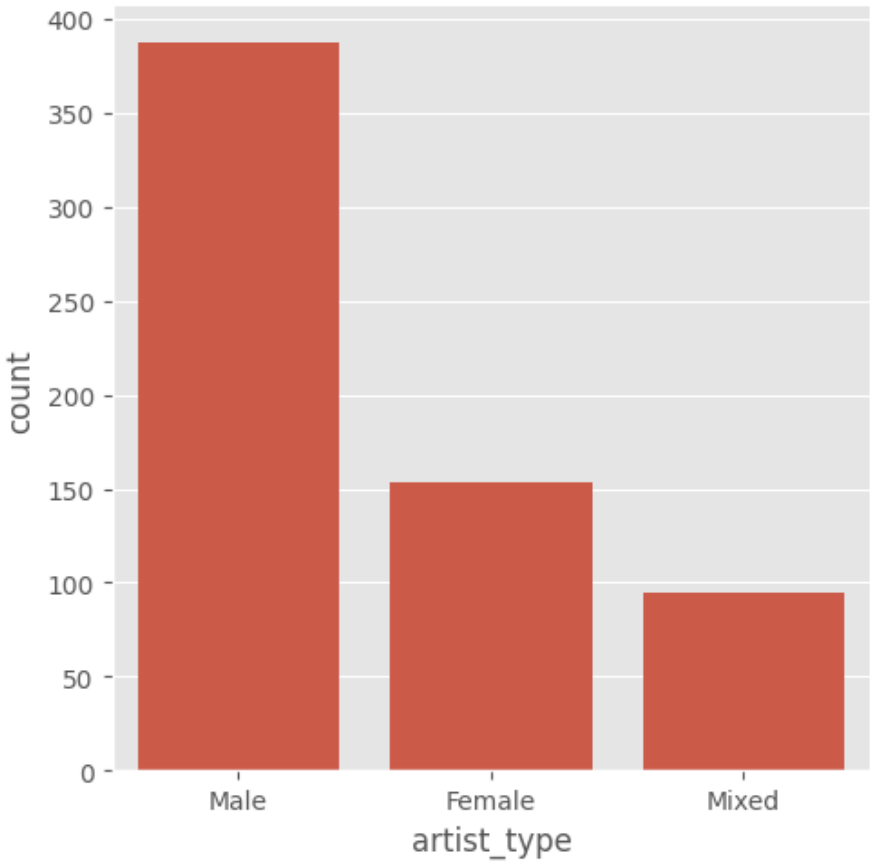

Figure 4: Tipo de artista

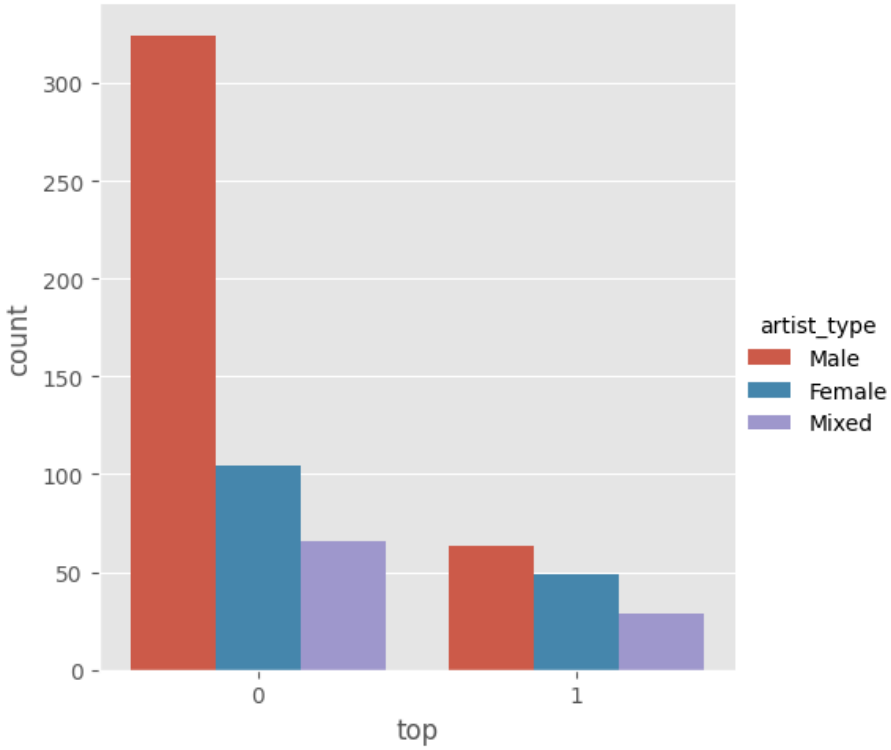

Figure 5: Tipo de artista en Top

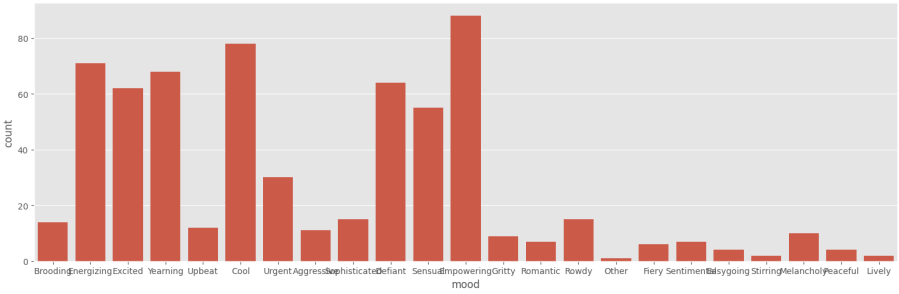

Figure 6: Mood de las canciones

Se busca una relación entre el año y la duración de la canción

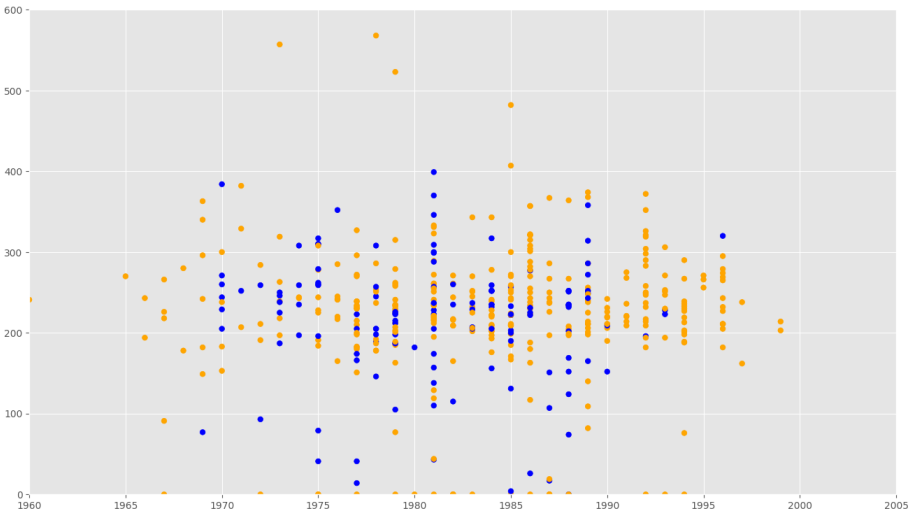

Figure 7: Relación Año-Duración

Se arreglan las edades de los artistas

```
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
```

```
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
```

Figure 8: Corrección de edades

Se calcula la edad promedio y se asigna a los valores nulos

```
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[np.isnan(artists_billboard['edad_en_billboard']), '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("Desvió Std 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

Figure 9: Correción de valores nulos

Ahora mapeamos los atributos

```
separador = "### ### ###"
grouped11 = artists_billboard.groupby('mood').size()#.sum().reset_index()
neworder11 = grouped11.sort_values(ascending=False)
print(neworder11)
print(separador)
print("Tempos de Canción: " + str(artists_billboard['tempo'].unique()))
print(separador)
print("Tipos de Artista: " + str(artists_billboard['artist_type'].unique()))
print(separador)
grouped11 = artists_billboard.groupby('genre').size()#.sum().reset_index()
neworder11 = grouped11.sort_values(ascending=False)
print(neworder11)
```
✓ 0.0s

Figure 10: Mapeo de datos

Mapeo de mood

```python
# Mood Mapping
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, # melancolico
                                                    'Upbeat': 5, #optimista alegre
                                                    'Stirring': 5, #emocionante
                                                    'Lively': 5, #animado
                                                    'Other': 0,'':0} ).astype(int)
```

Figure 11: Mood Mapping

```python
# Tempo Mapping
artists_billboard['tempoEncoded'] = artists_billboard['tempo'].map( {'Fast Tempo': 0, 'Medium Tempo': 2, 'Slow Tempo': 1, '': 0} ).astype(int)
# Genre Mapping
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}
                                  ).astype(int)
# artist_type Mapping
artists_billboard['artist_typeEncoded'] = artists_billboard['artist_type'].map( {'Female': 2, 'Male': 3, 'Mixed': 1, '': 0} ).astype(int)


# Mapping edad en la que llegaron al 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

# Mapping Song Duration
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
```

(variable) artists_billboard: DataFrame

0.0s

Figure 12: Genero, Tempo y artista

Se crea el arbol de decisión

Figure 13: Código del arbol

2 Resultados

Como resultado se obtuvo una precisión del 64.88

3 Conlusiones

Me pareció muy interesante realizar predicciones con arboles de decisión con Python ya que es de las primeras veces en las cuales implemento esta herramienta.