Árbol de Decisión en Python

Julio Cesar Torres Marquez

Marzo 2025

1 Introducción

Los árboles de decisión son representaciones gráficas de posibles soluciones a una decisión basadas en ciertas condiciones. Es uno de los algoritmos más utilizados en machine learning y puede realizar tareas de clasificación o regresión. Los árboles de decisión tienen un primer nodo llamado raíz y luego se descomponen en los atributos de entrada en dos ramas, planteando la condición que puede ser cierta o falsa. Se bifurca cada nodo en 2 y vuelve a subdividirse hasta llegar a las hojas, que son los nodos finales, y que equivalen a respuestas a la solución: Sí/No o lo que se esté clasificando.

2 Metodología

Para ejecutar el archivo en Python, se instalaron las siguientes librerías:

- numpy
- pandas
- seaborn
- matplotlib
- scikit-learn
- ipython
- pillow
- graphviz

Además, fue necesario instalar Graphviz desde su sitio oficial para poder visualizar el árbol. El enlace de descarga es el siguiente: Graphviz Download.

El código utilizado es el siguiente:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.metrics import accuracy_score
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
from graphviz import Source
```

2.1 Lectura del archivo CSV

El archivo CSV utilizado para este análisis es `artists_billboard_fix3.csv`.

```
artists_billboard = pd.read_csv("artists_billboard_fix3.csv")
```

2.2 Impresión de los primeros 5 registros

Para obtener una visión preliminar de los datos, se imprime la forma y los primeros 5 registros del DataFrame:

```
print(artists_billboard.shape)
print(artists_billboard.head())
```

2.3 Agrupación de los registros

Se agruparon los registros para ver cuántos alcanzaron el número uno y cuántos no:

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

2.4 Visualización gráfica

Se realizaron varias visualizaciones gráficas usando `seaborn` para explorar los datos, tales como:

- Tipo de artista
- Tempo de la canción
- Género de la canción
- Año de nacimiento

El código para visualización de tipo de artista es el siguiente:

```
sb.catplot(x="artist type", data=artists_billboard, kind="count", palette="viridis")
plt.show()
```

2.5 Sustitución de ceros por valores nulos en la columna "anioNacimiento"

Se sustituyeron los ceros en la columna de "anioNacimiento" por valores nulos utilizando la siguiente función:

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

artists_billboard['anio Nacimiento'] = artists_billboard.apply(lambda x: edad_fix(x['anio Nacimiento']),
```

2.6 Cálculo de la edad

Se creó una nueva columna llamada `edad en billboard` restando el año de aparición del artista en la Billboard (extraído de `chart date`) del año de nacimiento del artista:

```
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['anio Nacimien
```

2.7 Asignación de edades aleatorias a los registros faltantes

Para los registros con edades faltantes, se asignaron edades aleatorias basadas en la media y desviación estándar de las edades existentes:

```
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)

artists_billboard.loc[artists_billboard['edad en billboard'].isnull(), 'edad en billboard'] = age_null_r
artists_billboard['edad en billboard'] = artists_billboard['edad en billboard'].astype(int)
```

2.8 Visualización de los datos

Se realizó una visualización de los datos con colores representando si los registros tienen valores nulos o no:

```
f1 = artists_billboard['edad en billboard'].values
f2 = artists_billboard.index

colores = ['orange', 'blue', 'green']
asignar = []
for index, row in artists_billboard.iterrows():
    if (con_valores_nulos[index]):
        asignar.append(colores[2])
    else:
        asignar.append(colores[row['top']])

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

2.9 Mapeo de datos

Finalmente, se realizó un mapeo de los valores de las variables categóricas a valores numéricos para el entrenamiento del modelo de árbol de decisión:

```
artists_billboard['moodEncoded'] = artists_billboard['mood'].map({'Energizing': 6, 'Empowering': 6, 'Coo
artists_billboard['tempoEncoded'] = artists_billboard['tempo'].map({'Fast Tempo': 0, 'Medium Tempo': 2,
artists_billboard['genreEncoded'] = artists_billboard['genre'].map({'Urban': 4, 'Pop': 3, 'Traditional':
artists_billboard['artist type Encoded'] = artists_billboard['artist type'].map({'Female': 2, 'Male': 3,
```

3 Conclusiones

Los árboles de decisión son una herramienta poderosa en machine learning para la clasificación y regresión de datos. La visualización y el análisis de los datos es un paso crucial para preparar los datos para el modelo de árbol de decisión, y en este caso, se manejaron los datos faltantes mediante la asignación de edades aleatorias, lo cual permitió que el modelo pudiera ser entrenado de manera eficiente.