Actividad 12: Programando Árbol de Decisión en Python

Gerardo Enrique Torres Flores 2064063

30 de marzo de 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 de aprendizaje supervisado más utilizados en machine learning y pueden realizar tareas de clasificación o regresión.

Los árboles de decisión tienen un primer nodo llamado **raíz** (root) y luego se descomponen el resto de atributos de entrada en dos ramas planteando una condición que puede ser cierta o falsa. Se bifurca cada nodo en 2 y vuelven a su**bdividirse hasta llegar a las hojas que son los nodos finales** y que equivalen a respuestas a la solución: **Si/No, Comprar/Vender, o lo que sea que estemos clasificando**.

En esta actividad se empleará un árbol de decisión para analizar un conjunto de datos musical (`artists_billboard_fix3.csv`), evaluar el desempeño del modelo y realizar predicciones sobre el rendimiento de artistas en el billboard.

2. Metodología

Para realizar esta actividad se siguieron los siguientes pasos:

1. Importación de librerías y configuración

Se importaron las librerías necesarias para el análisis, visualización y creación del modelo de Árbol de Decisión:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
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, cross_val_score
from IPython.display import Image as PImage
from subprocess import check_call
from PIL import Image, ImageDraw, ImageFont
```

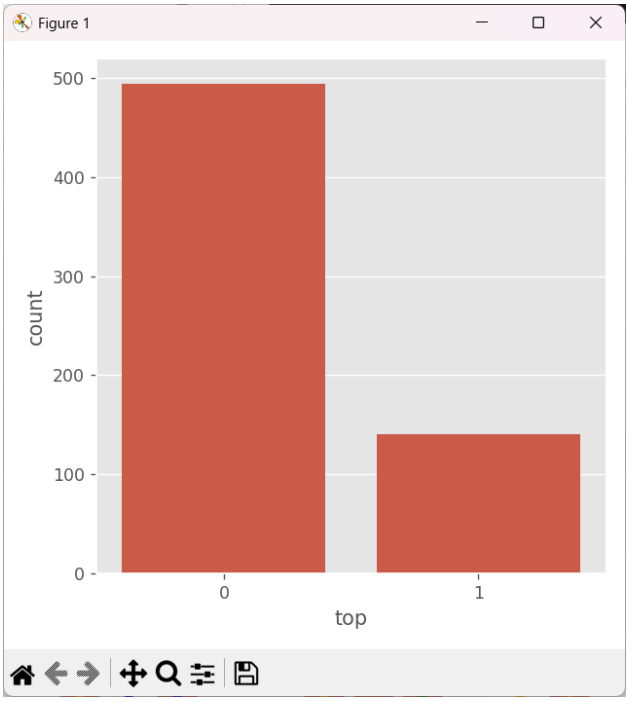
2. Lectura y exploración de datos

Se carga el conjunto de datos `artists_billboard_fix3.csv` y se exploran las distribuciones de las variables y las clases:

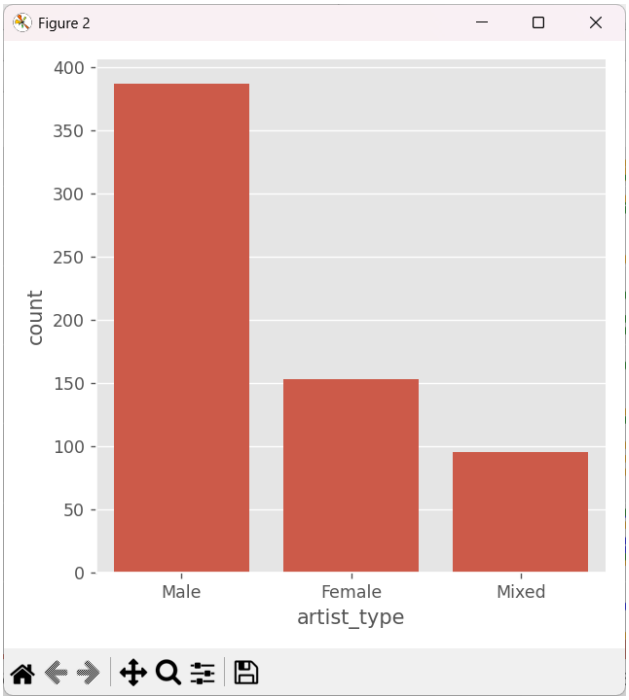
```
artists_billboard = pd.read_csv(r"artists_billboard_fix3.csv")
print(artists_billboard.groupby('top').size())

sb.catplot(x='top', data=artists_billboard, kind="count")
sb.catplot(x='artist_type', data=artists_billboard, kind="count")
sb.catplot(x='top', data=artists_billboard, hue='artist_type', kind="count")
sb.catplot(x='mood', data=artists_billboard, kind="count", aspect=3)
sb.catplot(x='tempo', data=artists_billboard, hue='top', kind="count")
sb.catplot(x='genre', data=artists_billboard, kind="count", aspect=3)
sb.catplot(x='mood', data=artists_billboard, hue='top', kind="count", aspect=3)
sb.catplot(x='anioNacimiento', data=artists_billboard, kind="count", aspect=3)
```

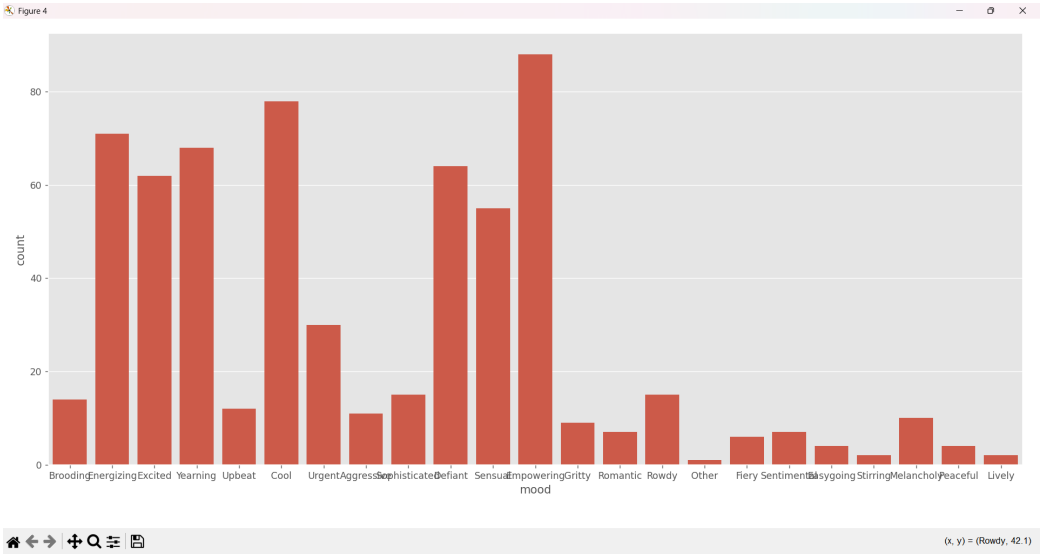
Visualización de cuántos estuvieron en Top 1 y no:



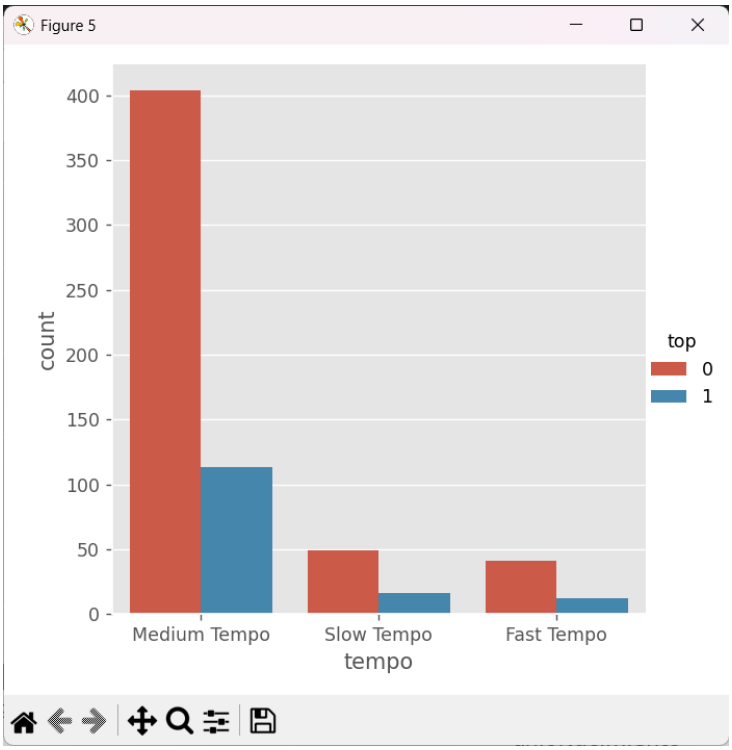
Visualización de Género del Artista(s):



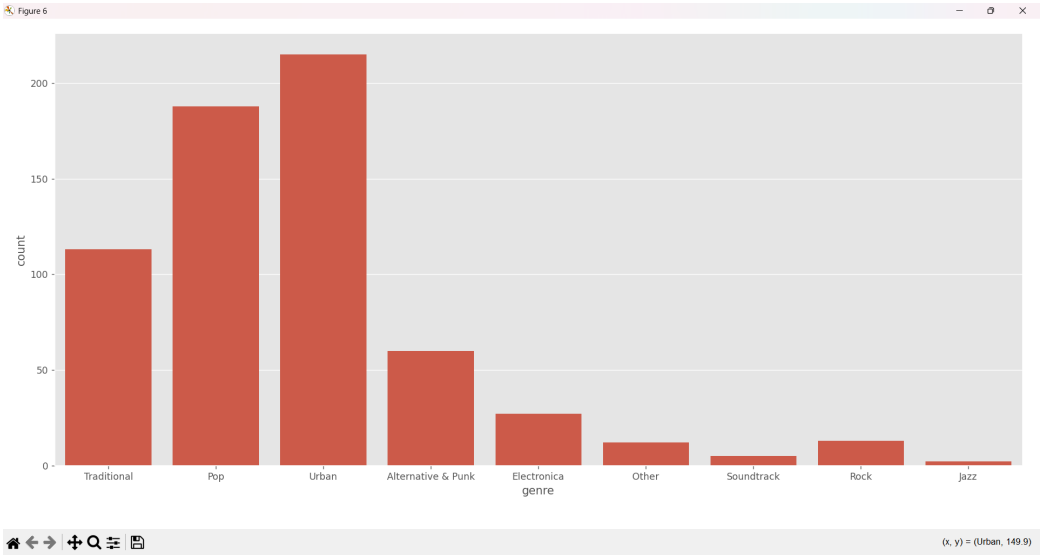
Visualización de Tipos de Mood:



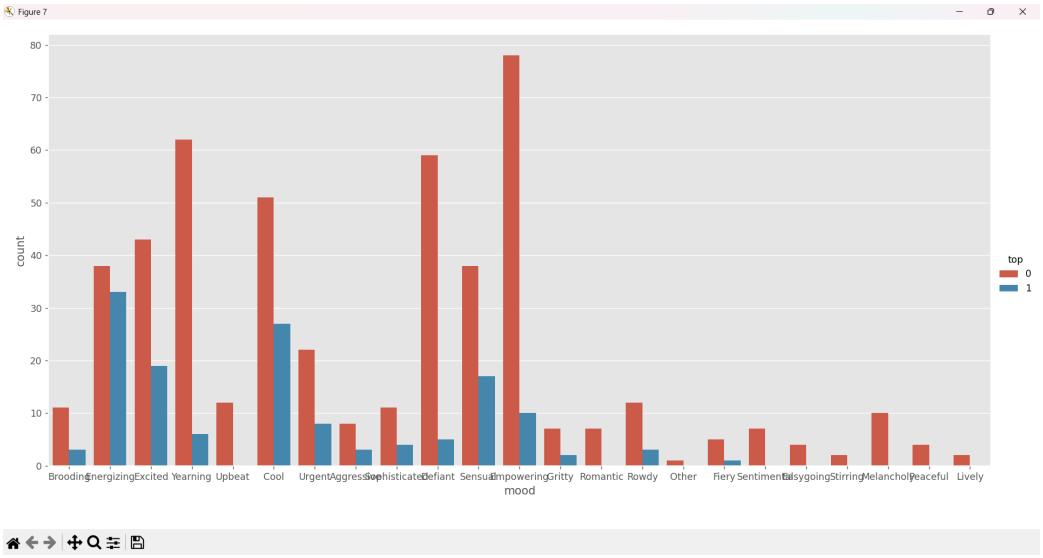
Visualización de Tempo de la Canción:



Visualización de Géneros Musicales:



Visualización de Géneros Musicales en Top 1 y no:



3. Visualización y preprocesamiento

Se realizaron varias visualizaciones para analizar relaciones entre variables (como duración de canción y año de nacimiento) y se realizó el preprocesamiento necesario (codificación de variables, manejo de datos faltantes, etc.):

```
# Visualizacion de relacion entre anio de nacimiento y duracion de cancion
colores = ['orange','blue']
tamanios = [60,40]
f1 = artists_billboard['anioNacimiento'].values
f2 = artists_billboard['durationSeg'].values

asignar = []
for index, row in artists_billboard.iterrows():
    asignar.append(colores[row['top']])
plt.scatter(f1, f2, c=asignar, s=30)
plt.axis([1960,2005,0,600])

# Preprocesamiento: ajuste de la edad y codificacion de variables
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)
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)

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[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("Desvio Std Edad: " + str(age_std))
print("Intervalo para asignar edad aleatoria: " + str(int(
```

```
    age_avg - age_std)) + " a " + str(int(age_avg + age_std)))
36
37  # Codificacion de variables (mood, tempo, genre, artist_type,
        edad, duracion)
38  artists_billboard['moodEncoded'] = artists_billboard['mood'].
        map({
39      'Energizing': 6, 'Empowering': 6, 'Cool': 5, 'Yearning':
            4,
40      'Excited': 5, 'Defiant': 3, 'Sensual': 2, 'Gritty': 3,
41      'Sophisticated': 4, 'Aggressive': 4, 'Fiery': 4, 'Urgent'
            : 3,
42      'Rowdy': 4, 'Sentimental': 4, 'Easygoing': 1, 'Melancholy
            ': 4,
43      'Romantic': 2, 'Peaceful': 1, 'Brooding': 4, 'Upbeat': 5,
44      'Stirring': 5, 'Lively': 5, 'Other': 0, '': 0
45  }).astype(int)
46  artists_billboard['tempoEncoded'] = artists_billboard['tempo'
        ].map({
47      'Fast Tempo': 0, 'Medium Tempo': 2, 'Slow Tempo': 1, '':
            0
48  }).astype(int)
49  artists_billboard['genreEncoded'] = artists_billboard['genre'
        ].map({
50      'Urban': 4, 'Pop': 3, 'Traditional': 2, 'Alternative &
            Punk': 1,
51      'Electronica': 1, 'Rock': 1, 'Soundtrack': 0, 'Jazz': 0,
52      'Other': 0, '': 0
53  }).astype(int)
54  artists_billboard['artist_typeEncoded'] = artists_billboard['
        artist_type'].map({
55      'Female': 2, 'Male': 3, 'Mixed': 1, '': 0
56  }).astype(int)
57  artists_billboard.loc[artists_billboard['edad_en_billboard']
        <= 21, 'edadEncoded'] = 0
58  artists_billboard.loc[(artists_billboard['edad_en_billboard']
        > 21) & (artists_billboard['edad_en_billboard'] <= 26), '
        edadEncoded'] = 1
59  artists_billboard.loc[(artists_billboard['edad_en_billboard']
        > 26) & (artists_billboard['edad_en_billboard'] <= 30), '
        edadEncoded'] = 2
60  artists_billboard.loc[(artists_billboard['edad_en_billboard']
        > 30) & (artists_billboard['edad_en_billboard'] <= 40), '
        edadEncoded'] = 3
61  artists_billboard.loc[artists_billboard['edad_en_billboard']
        > 40, 'edadEncoded'] = 4
```

```
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

drop_elements = ['id','title','artist','mood','tempo','genre','artist_type','chart_date','anioNacimiento','durationSeg','edad_en_billboard']
artists_encoded = artists_billboard.drop(drop_elements, axis=1)
colormap = plt.cm.viridis
plt.figure(figsize=(12,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sb.heatmap(artists_encoded.astype(float).corr(), linewidths=0.1, vmax=1.0, square=True, cmap=colormap, linecolor='white', annot=True)
plt.show()
```

4. Árbol de Decisión

Se realiza la validación mediante KFold, se evalúa la precisión del modelo para diferentes profundidades y se crea el árbol de decisión definitivo para la clasificación:

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

```

```
# Testeamos la profundidad de 1 a la cantidad de atributos + 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({"Max Depth": depth_range, "Average Accuracy": accuracies})
print(df.to_string(index=False))

# Entrenamos el arbol de decision con profundidad = 4
y_train = artists_encoded['top']
x_train = artists_encoded.drop(['top'], axis=1)
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)

# Exportamos el modelo a archivo .dot y generamos la imagen
```

```
    del arbol
with open(r"tree1.dot", 'w') as f:
    f = tree.export_graphviz(decision_tree,
                             out_file=f,
                             max_depth=7,
                             impurity=True,
                             feature_names=list(
                                 artists_encoded.drop(['top'],
                                  axis=1)),
                             class_names=['No', 'N1 Billboard
                                 '],
                             rounded=True,
                             filled=True)
check_call(['dot','-Tpng', r'tree1.dot', '-o', r'tree1.png'])
PImage("tree1.png")

# Precision del arbol de decision
acc_decision_tree = round(decision_tree.score(x_train,
    y_train) * 100, 2)
print(acc_decision_tree)

# Prediccion Camila Cabello
features = x_train.columns.tolist()
x_test = pd.DataFrame(columns=features)
x_test.loc[0] = [5, 2, 4, 1, 0, 3]
y_pred = decision_tree.predict(x_test)
print("Prediccion: " + str(y_pred))
y_proba = decision_tree.predict_proba(x_test)
print("Probabilidad de Acierto: " + str(round(y_proba[0][
    y_pred][0] * 100, 2)) + "%")

# Prediccion Imagine Dragons
x_test = pd.DataFrame(columns=features)
x_test.loc[0] = [4, 2, 1, 3, 2, 3]
y_pred = decision_tree.predict(x_test)
print("Prediccion: " + str(y_pred))
y_proba = decision_tree.predict_proba(x_test)
print("Probabilidad de Acierto: " + str(round(y_proba[0][
    y_pred][0] * 100, 2)) + "%")
```

3. Resultados

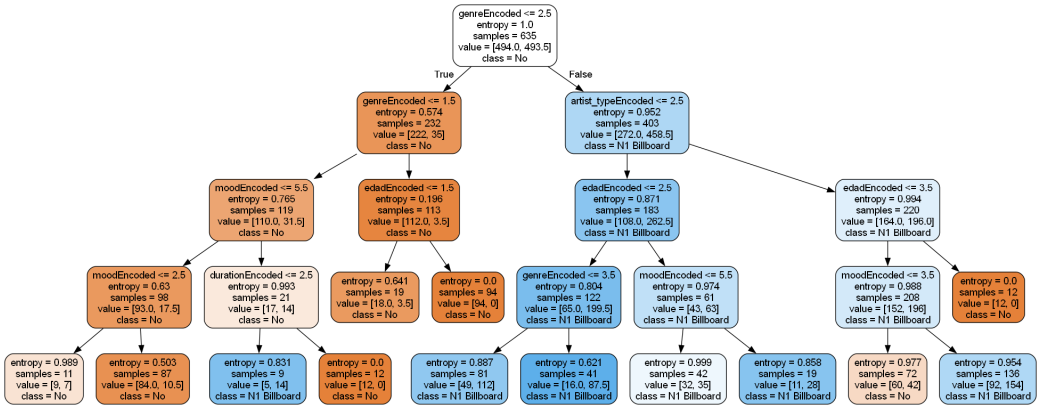
Algunos de los resultados que se obtuvieron fueron:

- **Evaluación de la profundidad del árbol**: Se generó una tabla con la precisión promedio para cada profundidad evaluada.
- **Visualización de correlación**: Se muestra el mapa de calor de correlación entre características.
- **Modelo final de Árbol de Decisión**: Se entrenó un árbol con profundidad 4, se exportó su estructura a un archivo .png y se calculó su precisión.
- **Predicciones**: Se realizaron predicciones para nuevos casos (ej. Camila Cabello e Imagine Dragons) junto con las probabilidades de acierto.

```
Prediccion: [1]
Probabilidad de Acierto: 84.54%
Prediccion: [0]
Probabilidad de Acierto: 88.89%
```

Como resultado de las predicciones, para Camila Cabello, **Havana llegará al top 1 con una probabilidad del 84 %.** Por otra parte, para Imagine Dragons, la segunda predicción nos menciona que **Believer NO llegará al Top 1 con una certeza del 88 %.**

Visualización del Árbol de Decisión de forma gráfica:



4. Conclusión

Se aplicó el **Árbol de Decisión** para clasificar y predecir el desempeño en el billboard a partir de diversas características de los artistas. Se exploraron y preprocesaron los datos, se evaluaron distintos parámetros del modelo mediante validación cruzada y se seleccionó una profundidad óptima para el árbol. Los resultados muestran que es una herramienta interpretativa y efectiva para la clasificación en escenarios complejos, permitiendo además identificar la importancia de cada característica en predicciones finales.

```
Edad Promedio: 30.10282258064516
Desvio Std Edad: 8.40078832861513
Intervalo para asignar edad aleatoria: 21 a 38
mood
Empowering       88
Cool             78
Energizing       71
Yearning         68
Defiant          64
Excited          62
Sensual          55
Urgent           30
Rowdy            15
Sophisticated    15
Brooding         14
Upbeat           12
Aggressive       11
Melancholy       10
Gritty            9
Sentimental       7
Romantic          7
Fiery             6
Easygoing         4
Peaceful          4
Lively            2
Stirring          2
Other             1
dtype: int64
### ### ###
Tempos de Cancion: ['Medium Tempo' 'Slow Tempo' 'Fast Tempo']
### ### ###
Tipos de Artista: ['Male' 'Female' 'Mixed']
### ### ###
genre
```

```
Urban                  215
Pop                    188
Traditional            113
Alternative & Punk      60
Electronica             27
Rock                    13
Other                   12
Soundtrack               5
Jazz                     2
dtype: int64
 Max Depth  Average Accuracy
         1          0.556101
         2          0.556126
         3          0.564038
         4          0.644122
         5          0.601711
         6          0.609400
         7          0.645610
64.88
Prediccion: [1]
Probabilidad de Acierto: 84.54%
Prediccion: [0]
Probabilidad de Acierto: 88.89%
```