Actividad 12: Programando Arbol de Decisión en Python

García Herrera Carlos Eduardo Marzo 2025

1 Introducción

Un árbol de decisión es un modelo gráfico utilizado para tomar decisiones basadas en una serie de condiciones. Su estructura se asemeja a un árbol, donde cada nodo interno representa una pregunta o criterio de decisión, las ramas indican posibles respuestas o caminos, y los nodos hoja muestran los resultados finales.
Se usa en diversos campos como la inteligencia artificial, la toma de decisiones empresariales y la ciencia de datos. Permite analizar diferentes escenarios de manera estructurada, facilitando la evaluación de riesgos, costos y beneficios asociados a cada opción.
En términos prácticos, un árbol de decisión ayuda a desglosar problemas complejos en decisiones más pequeñas y manejables, asegurando un proceso lógico y visualmente intuitivo para la toma de decisiones.
En el siguiente programa, hace uso de un árbol de decisión para predecir si una canción se encontró en el Top 1

2 Metodología

2.1 Parte 1: Creacion del Ambiente Virtual

```
#Automatic creation of an virtual environment to run the script and intall
the libraries
import subprocess
import os
import venv
import sys
script_dir = os.path.dirname(os.path.realpath(__file__))
env_name = os.path.join(script_dir, "VirtualEnv")
if os.path.exists(os.path.join(script_dir, "VirtualEnv")):
    #Checks if the VirtualEnv is activated (This is the path to the Python
installation currently in use. If the virtual environment is active,
sys.prefix will point to the virtual environment directory, while
sys.base_prefix points to the global Python installation.)
    if sys.prefix == sys.base_prefix:
        print("Activating the Virtual Environment...")
        python_exe = os.path.join(env_name, "Scripts", "python")
        subprocess.run([python_exe, __file__])
```

```
else:
    print("Installing the Required Libraries on a New Virtual Environment")
    venv.create(env_name, with_pip=True)

    # Step 2: Install the libraries
    libraries = ["scikit-learn",
"matplotlib","seaborn","pandas","numpy","IPython","graphviz"]
    for lib in libraries:
        subprocess.run([os.path.join(env_name, "Scripts", "pip"), "install",
lib], check=True)

    #Re-Run the script with the Virtual Env Activated
    python_exe = os.path.join(env_name, "Scripts", "python")
    subprocess.run([python_exe, __file__])
```

2.2 Parte 2: Importación de las librerías Necesarias

```
# Imports necesarios
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
from sklearn.model_selection import cross_val_score
from subprocess import check_call
import graphviz
```

2.3 Parte 3: Carga de los datos de Entrada y análisis de los parámetros

```
#Cargamos los datos de Entrada
artists_billboard = pd.read_csv(r"artists_billboard_fix3.csv")
print("Shape of the Data:")
print(artists_billboard.shape)
print("First 5 Registers:")
pd.set_option('display.max_columns', None) #Muestra todas las columnas del
dataSet
print(artists_billboard.head())

#Atributos de Entrada
#Cuantos alcanzaron el top 1
```

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

#Tipo de Artista
sb.catplot(data=artists_billboard,kind="count",x="artist_type",palette="viri
dis")
plt.show()

#Tipo de Artista con Top
sb.catplot(data=artists_billboard,kind="count",x="top",hue="artist_type",pal
ette="viridis")
plt.show()

#Mood del artista
sb.catplot(data=artists_billboard,kind="count",x="mood",palette="viridis")
plt.show()

#Tempo del artista con Top
sb.catplot(data=artists_billboard,kind="count",x="mood",hue="top",palette="v
iridis")
plt.show()

#Genero del Artista
sb.catplot(data=artists_billboard,kind="count",x="genre",palette="viridis")
plt.show()

#Genero del Artista con Top
sb.catplot(data=artists_billboard,kind="count",x="genre",hue="top",palette="
viridis")
plt.show()

#Anios de nacimiento
sb.catplot(data=artists_billboard,kind="count",x="anioNacimiento",palette="v
iridis")
plt.show()
```

2.4 Parte 4: Análisis y corrección de las edades de los datos

```
#Datos sin años de nacimiento:
nacimientosPorAnio = artists_billboard['anioNacimiento']
print("Cantidad de artistas sin año de nacimiento:")
print(len(nacimientosPorAnio[nacimientosPorAnio<=0]))

#Relacion entre Año y Duracion de la 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)
plt.axis([1960,2005,0,600])
plt.show()

#Relacion entre Años y Top
f1 = artists_billboard['chart_date'].values
f2 = artists_billboard['durationSeg'].values

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

#Arreglar 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)

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)

#Calculo de Promedios de Edad y asignacion a los registros 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)))

#Visualizamos las edades Agregadas:
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]) # verde
    else:
        asignar.append(colores[row['top']])

plt.scatter(f1, f2, c=asignar, s=30)
plt.axis([15,50,0,650])
plt.show()
```

2.5 Parte 5: Mapeo de los Datos

```
#Mapeo de Atributos
separador = "### ### ###"
grouped11 = artists_billboard.groupby('mood').size()#.sum().reset_index()
neworder11 = grouped11.sort_values(ascending=False)
```

```
print("\n\nCategorias de Mood:")
print(neworder11)
print(separador)

print("\n\nCategorias de Tempo:")
print("Tempos de Canción: " + str(artists_billboard['tempo'].unique()))
print(separador)

print("\n\nCategorias de Tipo de Artista:")
print("Tipos de Artista: " + str(artists_billboard['artist_type'].unique()))
print(separador)

print("\n\nCategorias de Genero:")
grouped11 = artists_billboard.groupby('genre').size()#.sum().reset_index()
neworder11 = grouped11.sort_values(ascending=False)
print(neworder11)

# 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)
# 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

drop_elements =
['id','title','artist','mood','tempo','genre','artist_type','chart_date','an
ioNacimiento','durationSeg','edad_en_billboard']
artists_encoded = artists_billboard.drop(drop_elements, axis = 1)
```

2.6 Parte 6: Análisis y Muestreo de los datos mapeados

```
print("\n\nDatos de Entrada Categoricos:")
print(artists_encoded.head())

print("\n\nCaracteristicas de los Datos de Entrada:")
print(artists_encoded.describe())

#Cracion de un HeatMap
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()

#Caracteristicas de cada entrada
print("\n\n")
print(artists_encoded[['moodEncoded', 'top']].groupby(['moodEncoded'],
as_index=False).agg(['mean', 'count', 'sum']))
print("\n\n")
print(artists_encoded[['artist_typeEncoded',
'top']].groupby(['artist_typeEncoded'], as_index=False).agg(['mean',
'count', 'sum']))
print("\n\n")
print(artists_encoded[['genreEncoded', 'top']].groupby(['genreEncoded'],
as_index=False).agg(['mean', 'count', 'sum']))
print("\n\n")
print(artists_encoded[['tempoEncoded', 'top']].groupby(['tempoEncoded'],
as_index=False).agg(['mean', 'count', 'sum']))
print("\n\n")
print(artists_encoded[['durationEncoded',
'top']].groupby(['durationEncoded'], as_index=False).agg(['mean', 'count',
'sum']))
```

```
print("\n\n")
print(artists_encoded[['edadEncoded', 'top']].groupby(['edadEncoded'],
as_index=False).agg(['mean', 'count', 'sum']))
```

2.7 Parte 7: Búsqueda árbol de decisión

```
#Busqueda del Arbol de Decision:
cv = KFold(n_splits=10) # Numero deseado de "folds" que haremos
accuracies = list()
max_attributes = len(list(artists_encoded))
depth_range = range(1, max_attributes + 1)

# Testearemos la profundidad de 1 a 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"]) # calculamos la
precision con el segmento de validacion
        fold_accuracy.append(valid_acc)

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

# Mostramos los resultados obtenidos
print("\n\nResultados del Arbol:")
df = pd.DataFrame({"Max Depth": depth_range, "Average Accuracy":
accuracies})
df = df[["Max Depth", "Average Accuracy"]]
print(df.to_string(index=False))
```

2.8 Parte 8: Creación árbol de decisión

```
#Creacion del Arbol de Decision
```

```
# Crear arrays de entrenamiento y las etiquetas que indican si llegó a top o
no
y_train = artists_encoded['top']
x_train = artists_encoded.drop(['top'], axis=1).values

# Crear Arbol de decision con profundidad = 4
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)
```

2.9 Parte 9: Exportación del árbol de decisión

```
# exportar el modelo a archivo .dot
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 )

# Convertir el archivo .dot a png para poder visualizarlo
os.environ["PATH"] +=os.pathsep+"C:\Program Files\Graphviz\\bin"
graph = graphviz.Source.from_file("tree1.dot")
graph.render("tree1", format="png", cleanup=True)  # Guarda como salida.png

#Precision del Arbol:
acc_decision_tree = round(decision_tree.score(x_train, y_train) * 100, 2)
print ("Precision del Arbol")
print(acc_decision_tree)
```

2.10 Parte 10: Predicciones

```
#Prediccion del Arbol:
#predecir artista CAMILA CABELLO featuring YOUNG THUG
# con su canción Havana llego a numero 1 Billboard US en 2017
x_test = pd.DataFrame(columns=('top','moodEncoded', 'tempoEncoded',
'genreEncoded','artist_typeEncoded','edadEncoded','durationEncoded'))
x_test.loc[0] = (1,5,2,4,1,0,3)
```

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

#predecir artista Imagine Dragons
# con su canción Believer llego al puesto 42 Billboard US en 2017

x_test = pd.DataFrame(columns=('top','moodEncoded', 'tempoEncoded',
'genreEncoded','artist_typeEncoded','edadEncoded','durationEncoded'))
x_test.loc[0] = (0,4,2,1,3,2,3)
y_pred = decision_tree.predict(x_test.drop(['top'], axis = 1))
print("Prediccion: " + str(y_pred))
y_proba = decision_tree.predict_proba(x_test.drop(['top'], axis = 1))
print("Probabilidad de Acierto: " + str(round(y_proba[0][y_pred][0]* 100,
2))+"%")

input("\n\nPresiona Cualquier tecla para Cerrar...")
```

3 Resultados

Al ejecutar el script de Python la información obtenida es la siguiente, cabe recalcar que la información es obtenida de forma secuencial a como se muestran a continuación:

3.1 Vista de los Datos

Se hace un análisis de los datos al ver distintas características de ellos, por medio de distintas graficas de barras para poder reconocer ciertas relaciones entre ellos.

```
Activating the Virtual Environment...
Shape of the Data:
(635, 11)
First 5 Registers:
   id                title  \
0   0  Small Town Throwdown
1   1             Bang Bang
2   2                Timber
3   3       Sweater Weather
4   4             Automatic

                                        artist        mood  \
0  BRANTLEY GILBERT featuring JUSTIN MOORE & THOM...    Brooding
1             JESSIE J, ARIANA GRANDE & NICKI MINAJ  Energizing
2                        PITBULL featuring KE$HA     Excited
3                                THE NEIGHBOURHOOD    Brooding
4                                  MIRANDA LAMBERT    Yearning

         tempo              genre artist_type  chart_date  durationSeg  top  \
0  Medium Tempo        Traditional        Male    20140628        191.0    0
1  Medium Tempo                Pop      Female    20140816        368.0    0
2  Medium Tempo              Urban       Mixed    20140118        223.0    1
3  Medium Tempo  Alternative & Punk        Male    20140104        206.0    0
4  Medium Tempo        Traditional      Female    20140301        232.0    0

   anioNacimiento
0          1975.0
1          1989.0
2          1993.0
3          1989.0
4             0.0
```

Ilustración 1: Vista preliminar de los datos, tanto la forma como las columnas del DataSet

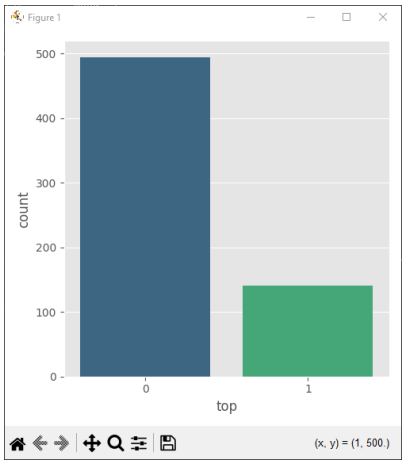

Ilustración 2:Cantidad de Artistas que han estado en Top 1

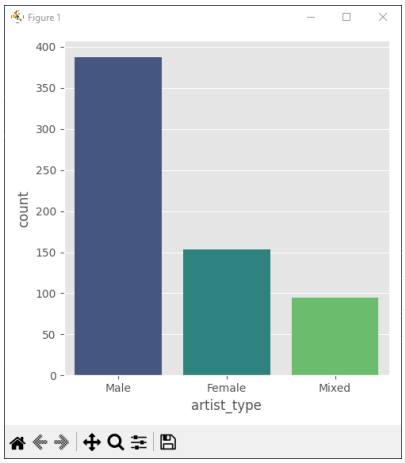

Ilustración 3: Clasificación por Tipo de Artista

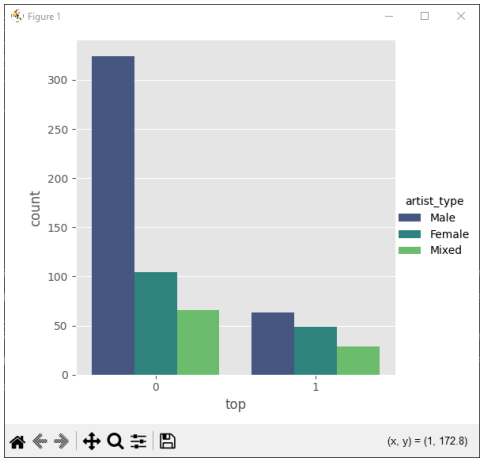

Ilustración 4: Cantidad de Artistas que han estado en el Top 1 clasificado por tipo de Artista

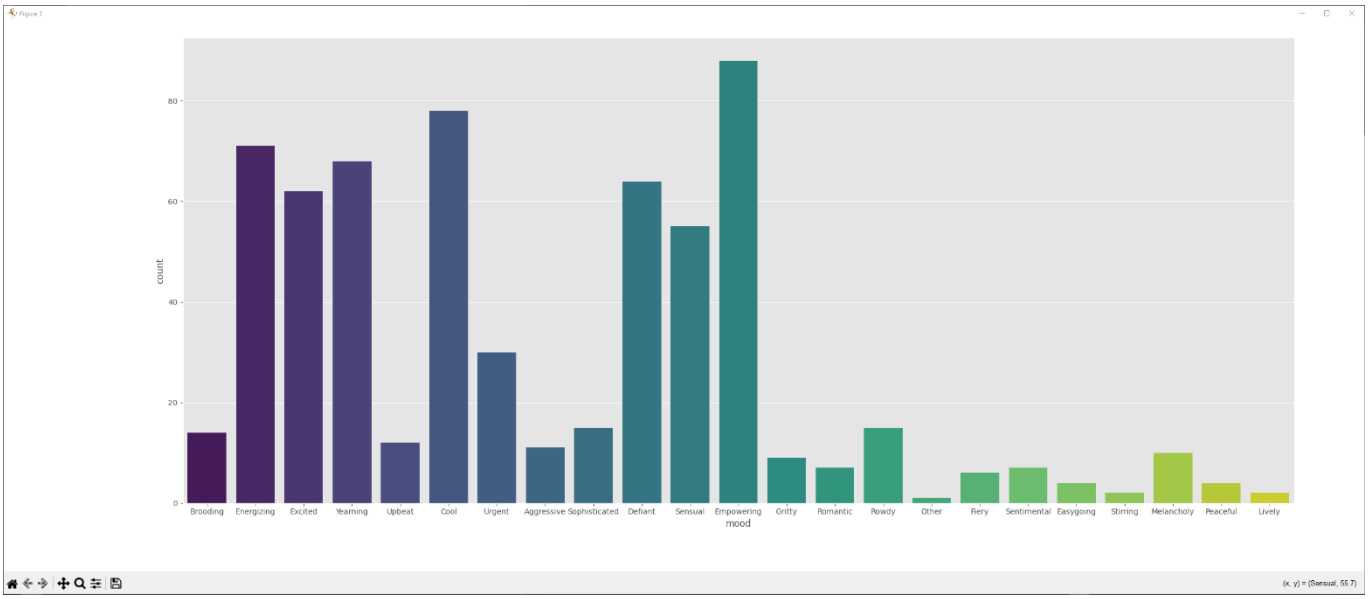

Ilustración 5:Cantidad de cada tipo de Mood

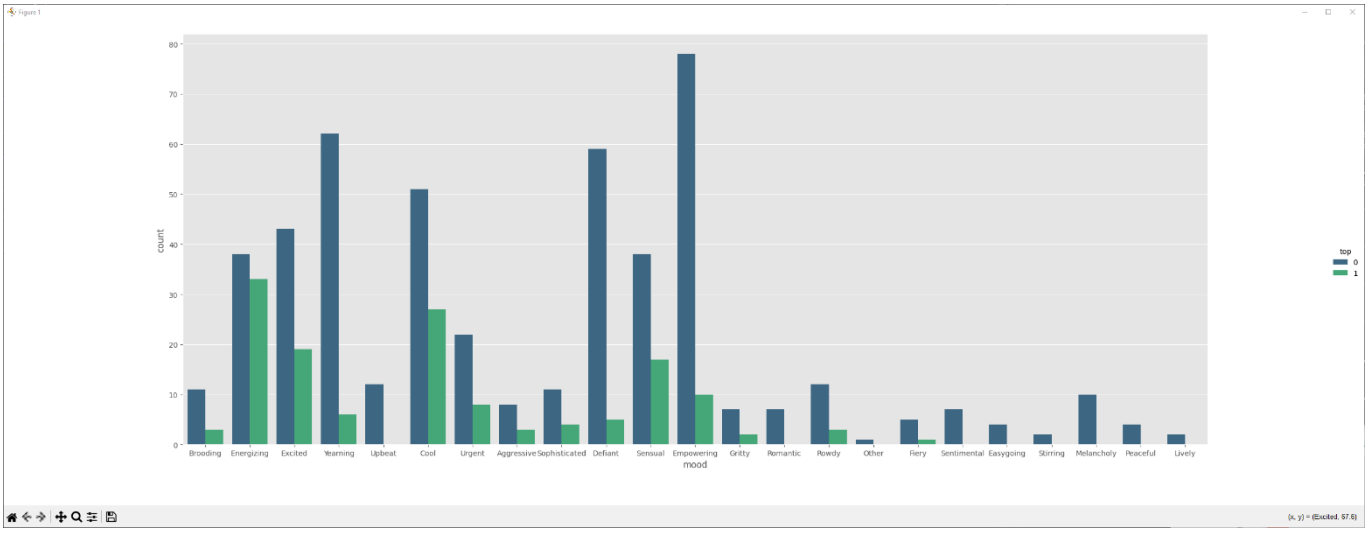

Ilustración 6: Cantidad de Moods clasificado por haber estado en el Top 1

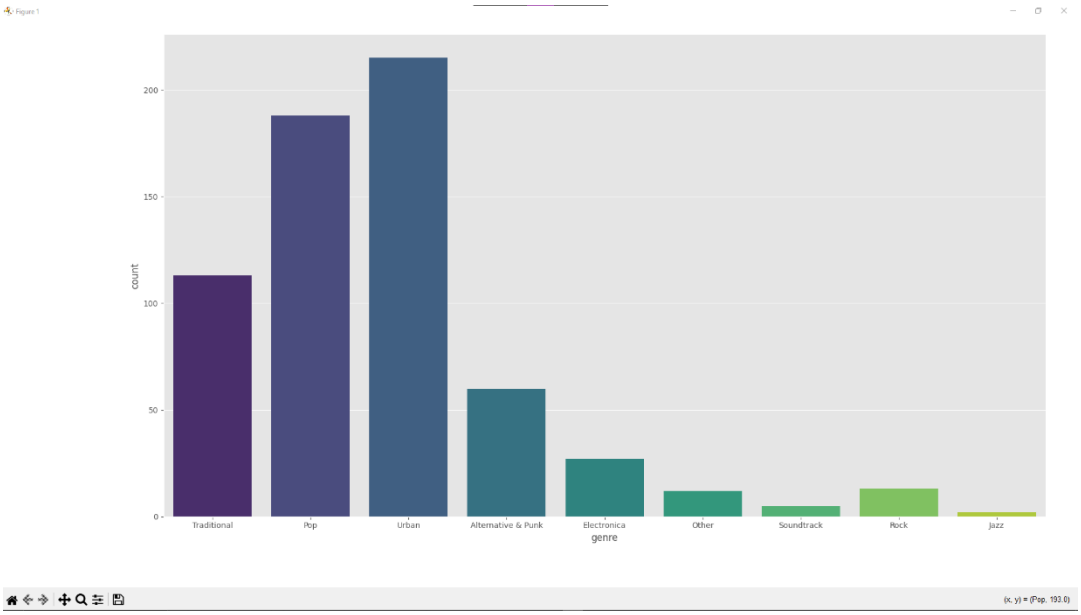

Ilustración 7: Cantidad de cada genero

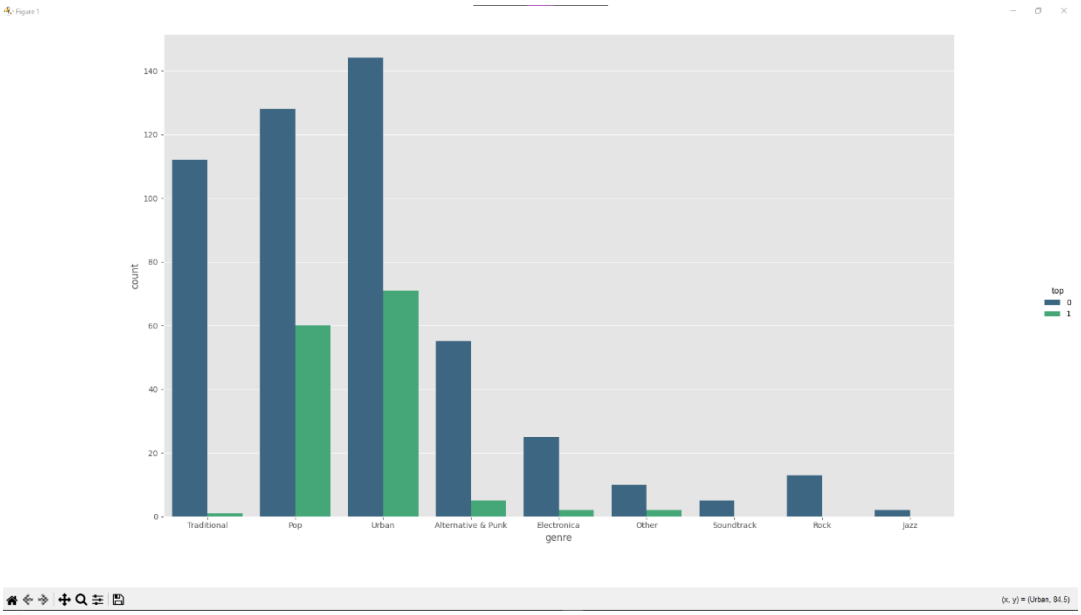

Ilustración 8: Cantidad de géneros clasificado por haber estado en el Top 1

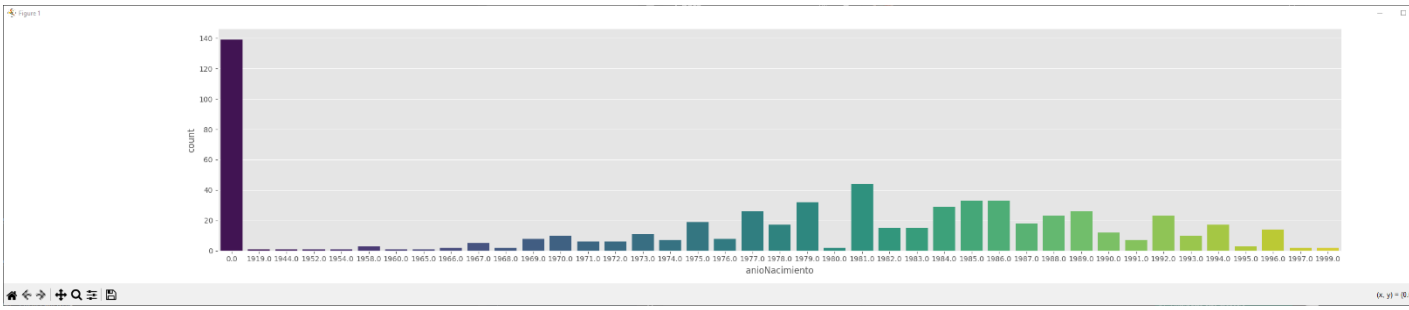

Ilustración 9: Cantidad de artistas por año de nacimiento

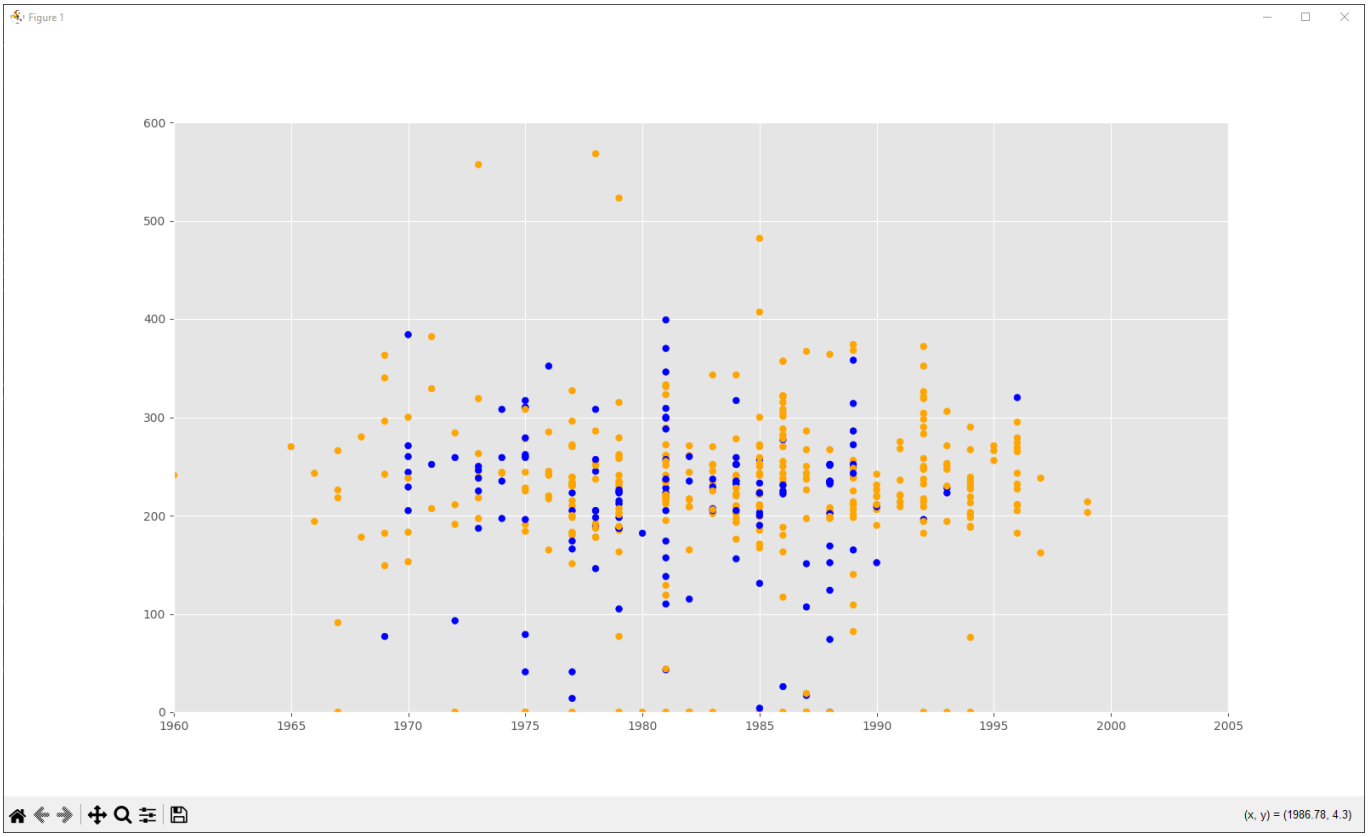

Ilustración 10: Relación entre Año de Nacimiento y Duracion de la Cancion (Top-Azul y no Top-Naranja)

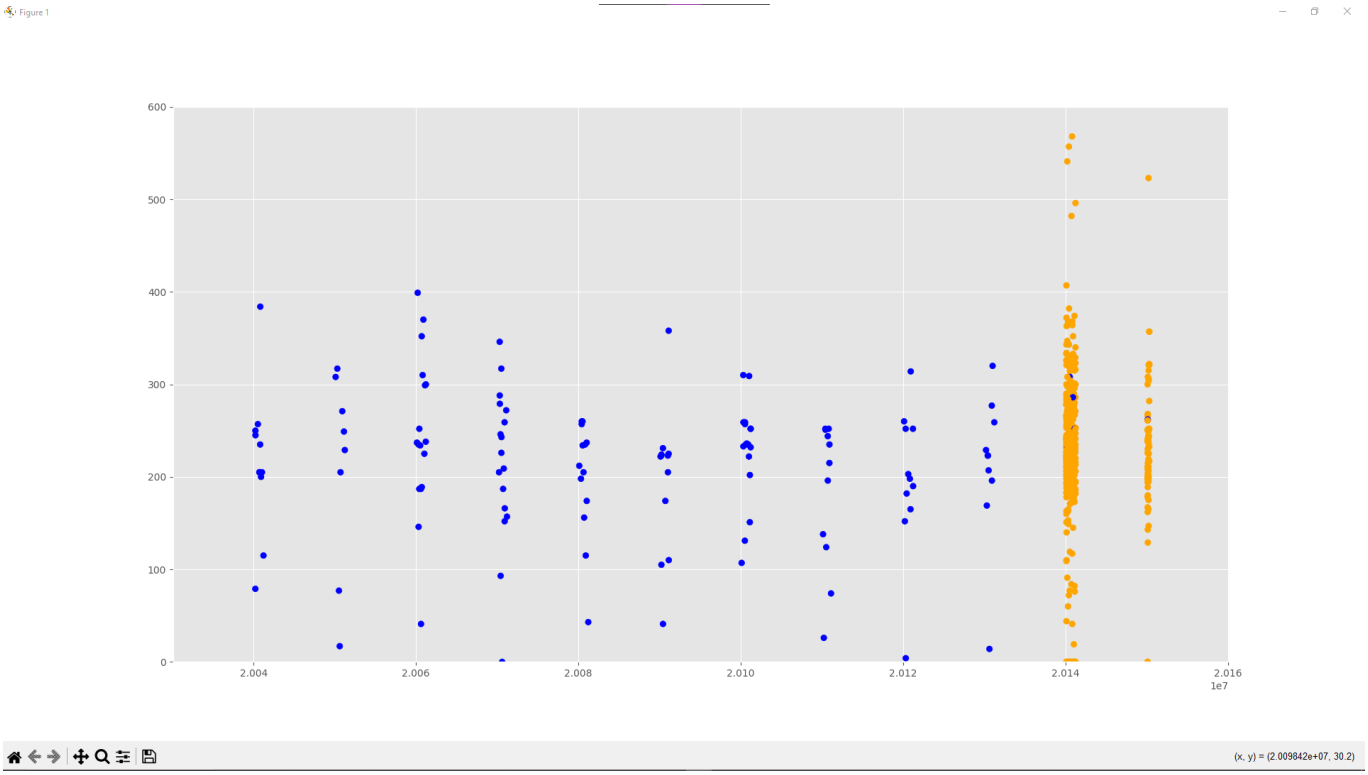

Ilustración 11: Relación entre Fecha de publicación y Duración (Top-Azul y no Top-Naranja)

3.2 Corrección de los Datos

Como se puede apreciar en la figura 9, existen muchos artistas sin año de nacimiento, por lo que será necesario calcularlo.

```
Cantidad de artistas sin año de nacimiento:
139
Edad Promedio: 30.10282258064516
Desvió Std Edad: 8.40078832861513
Intervalo para asignar edad aleatoria: 21 a 38
```

Ilustración 12: Datos calculados de la Edad

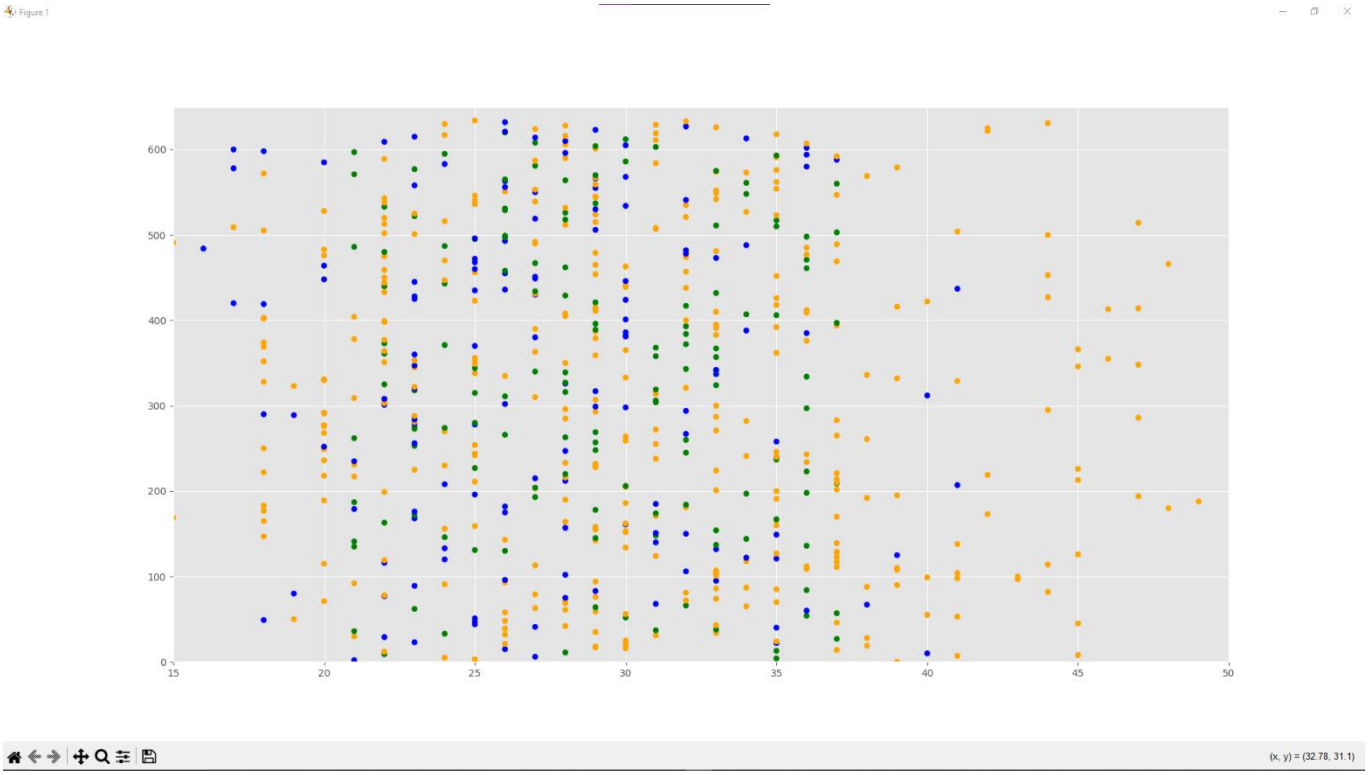

Ilustración 13: Distribución de la Edad (Verde=Nuevos Registros, Azul=Top, Naranja=No-Top)

3.3 Mapeo de los Datos categóricos

Para el mapeo de los datos, primero se tienen que conocer cada uno de los tipos que puede tener cada una de las columnas

```
Categorias de Mood:
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
### ### ###
```

Ilustración 14: Tipos de Mood

```
Categorias de Tempo:
Tempos de Canción: ['Medium Tempo' 'Slow Tempo' 'Fast Tempo']
### ### ###
```

Ilustración 15: Tipos de Tempo

```
Categorias de Tipo de Artista:
Tipos de Artista: ['Male' 'Female' 'Mixed']
### ### ###
```

Ilustración 16:Tipos de Artistas

```
Categorias de Genero:
genre
Urban                 215
Pop                   188
Traditional           113
Alternative & Punk     60
Electronica            27
Rock                   13
Other                  12
Soundtrack              5
Jazz                    2
dtype: int64
```

Ilustración 17: Tipos de géneros

```
Datos de Entrada Categoricos:
   top  moodEncoded  tempoEncoded  genreEncoded  artist_typeEncoded  \
0    0            4             2             2                   3
1    0            6             2             3                   2
2    1            5             2             4                   1
3    0            4             2             1                   3
4    0            4             2             2                   2

   edadEncoded  durationEncoded
0          3.0              2.0
1          1.0              6.0
2          0.0              3.0
3          1.0              2.0
4          3.0              3.0
```

Ilustración 18: Primeros Registros de los Datos Clasificados

```
Caracteristicas de los Datos de Entrada:
                top  moodEncoded  tempoEncoded  genreEncoded  \
count  635.000000   635.000000    635.000000    635.000000
mean     0.222047     4.344882      1.730709      2.755906
std      0.415950     1.350003      0.603553      1.165463
min      0.000000     0.000000      0.000000      0.000000
25%      0.000000     3.000000      2.000000      2.000000
50%      0.000000     4.000000      2.000000      3.000000
75%      0.000000     5.500000      2.000000      4.000000
max      1.000000     6.000000      2.000000      4.000000

       artist_typeEncoded  edadEncoded  durationEncoded
count          635.000000   635.000000       635.000000
mean             2.459843     2.022047         3.179528
std              0.740583     1.143967         1.775017
min              1.000000     0.000000         0.000000
25%              2.000000     1.000000         2.000000
50%              3.000000     2.000000         3.000000
75%              3.000000     3.000000         4.000000
max              3.000000     4.000000         6.000000
```

Ilustración 19: Datos estadísticos de los datos clasificados

moodEncoded	top		
	mean	count	sum
0	0.000000	1	0
1	0.000000	8	0
2	0.274194	62	17
3	0.145631	103	15
4	0.136986	146	20
5	0.294872	156	46
6	0.270440	159	43

Ilustración 20: Datos estadísticos de Mood Clasificado

artist_typeEncoded	top		
	mean	count	sum
0	1	0.305263	95	29
1	2	0.320261	153	49
2	3	0.162791	387	63

Ilustración 21: Datos estadísticos de Tipo de Artista Clasificado

	genreEncoded	top		
		mean	count	sum
0	0	0.105263	19	2
1	1	0.070000	100	7
2	2	0.008850	113	1
3	3	0.319149	188	60
4	4	0.330233	215	71

Ilustración 22: Datos estadísticos de Genero Clasificado

	tempoEncoded	top		
		mean	count	sum
0	0	0.226415	53	12
1	1	0.246154	65	16
2	2	0.218569	517	113

Ilustración 23: Datos estadísticos de Tempo Clasificado

	durationEncoded	top		
		mean	count	sum
0	0.0	0.295775	71	21
1	1.0	0.333333	30	10
2	2.0	0.212963	108	23
3	3.0	0.202381	168	34
4	4.0	0.232143	112	26
5	5.0	0.145455	55	8
6	6.0	0.208791	91	19

Ilustración 24: Datos estadísticos de duración Clasificado

	edadEncoded	top		
		mean	count	sum
0	0.0	0.250000	68	17
1	1.0	0.312102	157	49
2	2.0	0.246667	150	37
3	3.0	0.169014	213	36
4	4.0	0.042553	47	2

Ilustración 25: Datos estadísticos de edad Clasificado

3.4 Resultados del Árbol

Con los datos listos, es posible crear un árbol de decisión con ellos y poder entrenarlo

```
Resultados del Arbol:
 Max Depth  Average Accuracy
         1          0.556101
         2          0.556126
         3          0.564038
         4          0.645685
         5          0.612698
         6          0.622073
         7          0.633085
Precision del Arbol
68.19
```

Ilustración 26: Resultados obtenidos del modelo

3.5 Exportacion del Modelo

El modelo exportado es el siguiente:

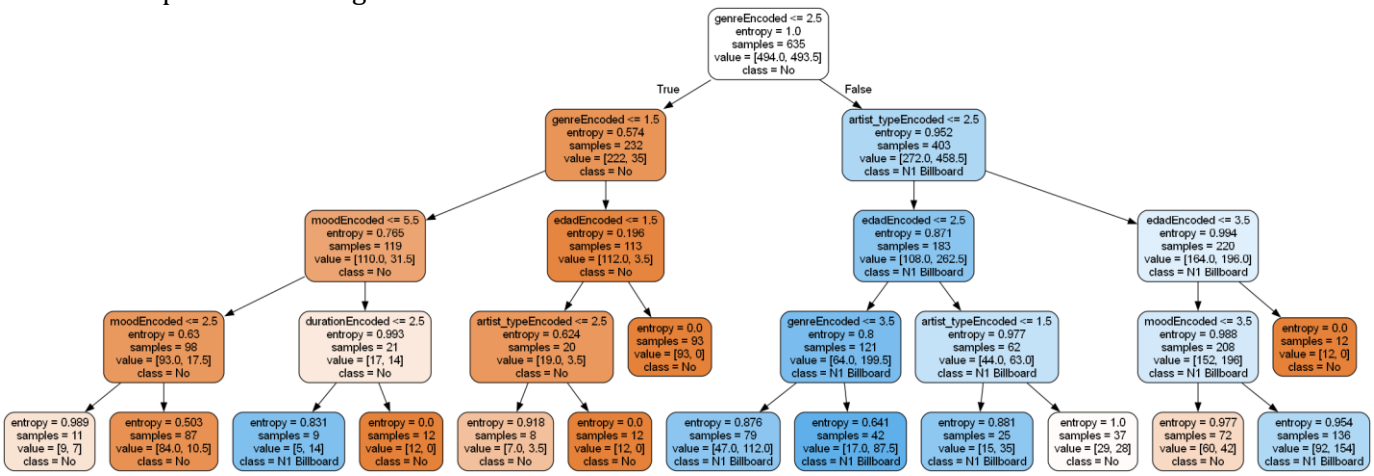

Ilustración 27: Árbol de decisión creado

3.6 Predicciones

```
E:\Documentos\UANL\6\Inteligencia
d without feature names
  warnings.warn(
Prediccion: [1]
E:\Documentos\UANL\6\Inteligencia
d without feature names
  warnings.warn(
Probabilidad de Acierto: 83.73%
```

Ilustración 28: Predicción para Havana (Top 1)

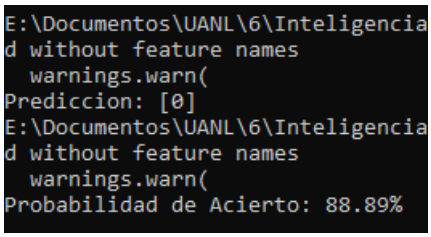

Ilustración 29: Predicción para Beliver (No Top)

4 Conclusión

un árbol de decisión es una herramienta poderosa para estructurar y visualizar el proceso de toma de decisiones. Su capacidad para descomponer problemas complejos en pasos más pequeños y manejables lo hace ideal para una variedad de aplicaciones, desde el análisis empresarial hasta la inteligencia artificial.

Además, su representación clara y lógica facilita la evaluación de diferentes escenarios y la identificación de la mejor alternativa según criterios definidos. Sin embargo, en problemas muy grandes o con muchas variables, los árboles de decisión pueden volverse complejos y difíciles de manejar, por lo que, en estos casos, pueden complementarse con otras técnicas como el aprendizaje automático o la poda de árboles para optimizar su eficiencia.