Act 12: Programando Árbol de Decisión en Python

Martín Alexis Martínez Andrade - 2049334

1. Introducción

Los árboles de decisión son algoritmos de aprendizaje supervisado que se utilizan tanto para tareas de clasificación como para regresión. Estos modelos representan gráficamente las decisiones mediante una estructura en forma de árbol, donde cada nodo interno corresponde a la evaluación de una condición y cada hoja representa una conclusión o respuesta final. En este ejercicio se usa scikit-learn para construir y visualizar un árbol de decisión que, mediante atributos de artistas y datos históricos del Billboard Hot 100, intenta predecir si un artista logrará alcanzar el puesto número 1 (top) en el ranking.

2. Metodología

La actividad se desarrolló siguiendo estos pasos:

1. **Carga y preprocesamiento de datos:**
    - Se cargó el dataset `artists_billboard_fix3.csv` utilizando Pandas.
    - Se revisaron los primeros registros y se imprimieron algunos valores estadísticos, así como el promedio y la desviación estándar de la edad de los artistas.
    - Se trató el problema de valores nulos en la columna del año de nacimiento, sustituyéndolos por valores aleatorios dentro del intervalo (usando el promedio y el desvío).
2. **Mapeo de variables:**
    - Se transformaron variables categóricas (como mood, tempo, género, tipo de artista) en valores numéricos.

- Se crearon nuevas columnas para .[ed]adEncodedz "durationEncoded".
- Se eliminó información innecesaria para formar el conjunto de datos final `artists_encoded`.

3. **Construcción y validación del árbol de decisión:**
   - Se utilizó la función `KFold` para probar distintos niveles de profundidad (max_depth del árbol) y determinar el valor óptimo. Se realizaron pruebas para profundidades desde 1 hasta el número de atributos + 1.
   - Se configuró el árbol de decisión y se usó el criterio de entropía para evaluar la calidad de las divisiones. Se configuró `class_weight` para compensar el desbalance en la variable de salida.
   - Se ajustó el modelo sobre el subset de entrenamiento y se evaluó la precisión obtenida.
4. **Visualización y predicción:**
   - Se visualizó el árbol con la función `plot_tree` de Matplotlib.
   - Se realizaron pruebas de predicción con dos ejemplos: uno para Camila Cabello (alcanzó el puesto número 1) y otro para Imagine Dragons (no alcanzó el top).

Fragmento de Código en Python

```
# Author: Mart n Alexis Mart nez Andrade - 2049334

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

```

```
# Datos de entrada
artists_billboard = pd.read_csv(r"artists_billboard_fix3.csv")
print(artists_billboard.head())

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

colores = ['orange', 'blue']  # si no estaban declarados previamente
tamanios = [60, 40]  # si no estaban declarados previamente

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

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)
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("Desvi  Std Edad: " + str(age_std))
print("Intervalo para asignar edad aleatoria: " + str(int(
    age_avg - age_std)) + " a " + str(int(age_avg + age_std)))


# 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
)
```

```
90
91  # Genre Mapping
92  artists_billboard['genreEncoded'] = artists_billboard['genre'
        ].map( {'Urban': 4,
93       'Pop': 3,
94       'Traditional': 2,
95       'Alternative & Punk': 1,
96       'Electronica': 1,
97       'Rock': 1,
98       'Soundtrack': 0,
99       'Jazz': 0,
100      'Other':0,'':0}
101      ).astype(int)
102
103 # artist_type Mapping
104 artists_billboard['artist_typeEncoded'] = artists_billboard['
        artist_type'].map(
105      {'Female': 2, 'Male': 3, 'Mixed': 1, '': 0}).astype(int)
106
107 # Mapping edad en la que llegaron al billboard
108 artists_billboard.loc[ artists_billboard['edad_en_billboard']
         <= 21, 'edadEncoded'] = 0
109 artists_billboard.loc[(artists_billboard['edad_en_billboard']
         > 21) & (artists_billboard['edad_en_billboard'] <= 26), '
        edadEncoded'] = 1
110 artists_billboard.loc[(artists_billboard['edad_en_billboard']
         > 26) & (artists_billboard['edad_en_billboard'] <= 30), '
        edadEncoded'] = 2
111 artists_billboard.loc[(artists_billboard['edad_en_billboard']
         > 30) & (artists_billboard['edad_en_billboard'] <= 40), '
        edadEncoded'] = 3
```

```
112 artists_billboard.loc[ artists_billboard['edad_en_billboard']
        > 40, 'edadEncoded'] = 4
113 # Mapping Song Duration
114 artists_billboard.loc[ artists_billboard['durationSeg'] <=
        150, 'durationEncoded'] = 0
115 artists_billboard.loc[(artists_billboard['durationSeg'] >
        150) & (artists_billboard[\
116 'durationSeg'] <= 180), 'durationEncoded'] = 1
117 artists_billboard.loc[(artists_billboard['durationSeg'] >
        180) & (artists_billboard[\
118 'durationSeg'] <= 210), 'durationEncoded'] = 2
119 artists_billboard.loc[(artists_billboard['durationSeg'] >
        210) & (artists_billboard[\
120 'durationSeg'] <= 240), 'durationEncoded'] = 3
121 artists_billboard.loc[(artists_billboard['durationSeg'] >
        240) & (artists_billboard[\
122 'durationSeg'] <= 270), 'durationEncoded'] = 4
123 artists_billboard.loc[(artists_billboard['durationSeg'] >
        270) & (artists_billboard[\
124 'durationSeg'] <= 300), 'durationEncoded'] = 5
125 artists_billboard.loc[ artists_billboard['durationSeg'] >
        300, 'durationEncoded'] = 6
126
127 drop_elements = ['id','title','artist','mood','tempo','genre'
        ,'artist_type','chart_date','anioNacimiento','durationSeg'
        ,'edad_en_billboard']
128 artists_encoded = artists_billboard.drop(drop_elements, axis
        = 1)
129
130 print(artists_encoded[['moodEncoded', 'top']].groupby(['
        moodEncoded'], as_index=False).agg(['mean', 'count', 'sum'
        ]))
131 print(artists_encoded[['artist_typeEncoded', 'top']].groupby
        (['artist_typeEncoded'], as_index=False).agg(['mean', '
        count', 'sum']))
132 print(artists_encoded[['genreEncoded', 'top']].groupby(['
        genreEncoded'], as_index=False).agg(['mean', 'count', 'sum
        ']))
133 print(artists_encoded[['tempoEncoded', 'top']].groupby(['
        tempoEncoded'], as_index=False).agg(['mean', 'count', 'sum
        ']))
134 print(artists_encoded[['durationEncoded', 'top']].groupby(['
        durationEncoded'], as_index=False).agg(['mean', 'count', '
        sum']))
135 print(artists_encoded[['edadEncoded', 'top']].groupby(['
        edadEncoded'], as_index=False).agg(['mean', 'count', 'sum'
        ]))
136
137 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)
# Mostramos los resultados obtenidos
df = pd.DataFrame({"Max Depth": depth_range, "Average Accuracy": accuracies})
df = df[["Max Depth", "Average Accuracy"]]
print(df.to_string(index=False))

# Crear arrays de entrenamiento y las etiquetas que indican si lleg  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)

# 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,
    )

# Visualizar sin Graphviz
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
tree.plot_tree(decision_tree,
    feature_names=list(artists_encoded.drop(['top'], axis=1).
        columns),
    class_names=['No', 'N1 Billboard'],
    filled=True,
    rounded=True,
    impurity=True)
plt.show()

acc_decision_tree = round(decision_tree.score(x_train,
    y_train) * 100, 2)
print(acc_decision_tree)

#predecir artista CAMILA CABELLO featuring YOUNG THUG
# con su canci n 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 canci n 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)) + "%")
```

Listing 1: Árbol de Decisión en Python

3. Resultados

- Tras optimizar la profundidad del árbol mediante cross validation, se obtuvo que una profundidad de 4 generaba la mejor precisión (alrededor del 64 % en entrenamiento).
- La visualización del árbol permitió observar que el nodo raíz realiza una división, por ejemplo, basada en el género, y que en niveles posteriores intervienen otras características como edad y duración.
- La precisión global del árbol, una vez ajustado, fue del 68.19 %.
- Para el caso de predicción de ejemplos:
  - Se predijo que Camila Cabello (con la canción "Havana") alcanzaría el top 1, con una probabilidad aproximada del 71.43 %.
  - Se predijo que Imagine Dragons (con "Believer") NO alcanzaría el top 1, con una probabilidad de acierto del 88.89 %.

4. Conclusión

En esta actividad se implementó un árbol de decisión utilizando Python y scikit-learn para predecir si un artista alcanzará el puesto número 1 en el Billboard Hot 100. Se procesaron y transformaron los datos originales, se realizó un mapeo de variables categóricas y se optimizó la profundidad del árbol usando cross validation. Aunque la precisión final del modelo no es muy alta, los experimentos con predicción de ejemplos nos permitieron evaluar el comportamiento del algoritmo frente a nuevos casos.

```
C:\AlexisAndradeDev\UANL\Inteligencia Artificial\Códigos\12>python main.py
   id                title  ... top anioNacimiento
0   0  Small Town Throwdown  ...   0         1975.0
1   1             Bang Bang  ...   0         1989.0
2   2                Timber  ...   1         1993.0
3   3       Sweater Weather  ...   0         1989.0
4   4             Automatic  ...   0            0.0

[5 rows x 11 columns]
Edad Promedio: 30.10282258064516
Desvió Std Edad: 8.40078832861513
Intervalo para asignar edad aleatoria: 21 a 38
  moodEncoded       top
                   mean count sum
0           0  0.000000     1   0
1           1  0.000000     8   0
2           2  0.274194    62  17
3           3  0.145631   103  15
4           4  0.136986   146  20
5           5  0.294872   156  46
6           6  0.270440   159  43
  artist_typeEncoded       top
                          mean count sum
0                  1  0.305263    95  29
1                  2  0.320261   153  49
2                  3  0.162791   387  63
  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
  tempoEncoded       top
                    mean count  sum
0            0  0.226415    53   12
1            1  0.246154    65   16
2            2  0.218569   517  113
  durationEncoded       top
                       mean count sum
0              0.0  0.295775    71  21
1              1.0  0.333333    30  10
```

```
  edadEncoded       top
                   mean count sum
0         0.0  0.246575    73  18
1         1.0  0.309677   155  48
2         2.0  0.256944   144  37
3         3.0  0.166667   216  36
4         4.0  0.042553    47   2
 Max Depth  Average Accuracy
         1          0.556101
         2          0.556126
         3          0.564038
         4          0.640923
         5          0.600000
         6          0.609524
         7          0.643998
68.19
C:\Users\alexi\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\validation.py:2732: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without fe
ature names
  warnings.warn(
Prediccion: [1]
C:\Users\alexi\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\validation.py:2732: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without fe
ature names
  warnings.warn(
Probabilidad de Acierto: 71.43%
C:\Users\alexi\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\validation.py:2732: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without fe
ature names
  warnings.warn(
Prediccion: [0]
C:\Users\alexi\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\validation.py:2732: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without fe
ature names
  warnings.warn(
Probabilidad de Acierto: 88.89%
```