Ábrol de decisión

Valeria Rodríguez

28 de Marzo del 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 (acrónimo del inglés CART).

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 subdividirse hasta llegar a las hojas que son los nodos finales.

Este algoritmo analiza los datos y las salidas para decidir la mejor forma de hacer divisiones (*splits*) entre nodos. Este algoritmo siempre tiene en cuenta de qué manera lograr una predicción con mayor probabilidad de acierto.

2 Metodología

Para la realización de esta actividad se llevaron a cabo una serie de pasos encaminados a la recreación del problema de árbol de decisión presentado en el libro *Aprende Machine Learning* del autor Juan Ignacio Bagnato, páginas 52-71.

2.1 Creación de carpeta de trabajo

Se creó una carpeta llamada Arbol de decisión, en la cual se creó un archivo Python llamado arbol_decison para la codificación de la actividad. Posteriormente, se descargó el archivo csv necesario para la actividad y se añade a la carpeta de trabajo.

2.2 Desarrollo del código

Con ayuda del IDE Visual Studio Code se abrió el archivo .py anteriormente creado y se codificó según las especificaciones del libro. Primeramente, se hicieron los imports al archivo

Imports needed for the script

```
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
```

De ser necesario, se intalan las paqueterías que faltan con ayuda de *pip*.

Primeramente se hace la lectura del archivo csv para el análisis de los datos del mismo.

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

Se imprimen las columnas y regristros así como las primeras filas del archivo de datos.

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

Se agrupan los registros sobre si han alcanzado o no el *número uno* en el top Billboard. Estas indican 0 si no se ha alcanzado y 1 si sí se ha alcanzado el top.

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

Posteriormente, se analizan los datos por *tipo de artista*, *mood*, *tempo* y *género* de las canciones.

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

Se visualizan los top y no top de acuerdo a las fechas en los charts.

```
colores = ['orange', 'blue']
tamanios = [60, 40]
asignar = []
asignar2 = []
for index, row in artists_billboard.iterrows():
        asignar.append(colores[row['top'] % 2])
        asignar2.append(tamanios[row['top'] % 2])
f1 = artists_billboard['chart_date'].values
f2 = artists_billboard['durationSeg'].values
print(len(f1), len(f2), len(asignar), len(asignar2))
plt.scatter(f1, f2, c=asignar, s=asignar2)
plt.axis([20030101, 20160101, 0, 600])
plt.show()
```

Para preparar los datos y arreglar aquellos que dificultarían el análisis se harán algunos ajustes. Primeramente, se sustituirán los ceros en la columna del Año de nacimiento por el valor "None", el valor nulo de Python. Después se calculan las edades en una nueva columna restando el año de aparición al año de nacimiento.

```
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)
```

Finalmente, asignamos edades aleatorias a los registros faltantes con ayuda del promedio de la edad y su desvío estándar y lo visualizamos en color verde con una gráfica.

```
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()
```

Después se hace un mapeo de los datos según diversos intereses en los mismos y finalmente se hace un nuevo conjunto de datos con los atributos definitivos para crear el árbol. Se quitan las columnas que no son necesarias con un drop.

```
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)
```

Ahora, se revisan los top 1 en los diferentes atributos mapeados. Sobre la columna *sum* se mostrará la cantidad de los que sí llegaron al número 1.

```
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)
artists_encoded[['moodEncoded', 'top']].groupby(['moodEncoded'], as_index=False).agg(['mean',
'count', 'sum'])
artists_encoded[['artist_typeEncoded', 'top']].groupby(['artist_typeEncoded'],
as_index=False).agg(['mean', 'count', 'sum'])
artists_encoded[['genreEncoded', 'top']].groupby(['genreEncoded'], as_index=False).agg(['mean',
'count', 'sum'])
artists_encoded[['tempoEncoded', 'top']].groupby(['tempoEncoded'], as_index=False).agg(['mean',
'count', 'sum'])
artists_encoded[['durationEncoded', 'top']].groupby(['durationEncoded'], as_index=False).
agg(['mean', 'count', 'sum'])
artists_encoded[['edadEncoded', 'top']].groupby(['edadEncoded'], as_index=False).agg(['mean',
'count', 'sum'])
```

Posteriormente, se crea el árbol con ayuda de la librería sklearn y los parámetros necesarios, cuyos valores fueron obtenidos a base de prueba y error.

```
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)
```

Además, se hacen pruebas para verificar la profundidad ideal para el árbol.

De estas pruebas, se obtuvo que el valor ideal de capas es 7. Ahora, se realiza la visualización del árbol de decisión con los datos de entrada y los parámetros anteriormente configurados.

```
Source.from_file("tree1.dot").render("tree1", format="png")
```

Se verifica cuál fue la precisión del árbol.

```
acc_decision_tree = round(decision_tree.score(x_train, y_train) * 100, 2)
print("Precisión del árbol: ", acc_decision_tree)
```

Finalmente, ponemos a prueba el modelo con dos canciones: *Havana* de Camila Cabello que sí llegó al top 1, y *Believer* de Imagine Dragons que si bien alcanzó un buen puesto, no llegó al top.

```
#Havana
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))
proba=y_proba[0][y_pred]*100
print("Probabilidad de Acierto Havana: " + str(round(proba[0], 2))+"%")
#Believer
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))
proba=y_proba[0][y_pred]*100
print("Probabilidad de Acierto Havana: " + str(round(proba[0], 2))+"%")
```

Con esto se termina el código y se ejecuta para obtener los resultados.

3 Resultados

A continuación, se muestran los resultados obtenidos en las diversas fases de codificación mostradas en la sección anterior.

Primeramente, se ejecutaron una serie de comandos para obtener información del archivo de datos.

```
Shape:
(635, 11)
Head
   id                title                                             artist  ... durationSeg top anioNacimiento
0   0  Small Town Throwdown  BRANTLEY GILBERT featuring JUSTIN MOORE & THOM...  ...       191.0   0         1975.0
1   1            Bang Bang             JESSIE J, ARIANA GRANDE & NICKI MINAJ  ...       368.0   0         1989.0
2   2               Timber                           PITBULL featuring KE$HA  ...       223.0   1         1993.0
3   3      Sweater Weather                                THE NEIGHBOURHOOD  ...       206.0   0         1989.0
4   4            Automatic                                  MIRANDA LAMBERT  ...       232.0   0            0.0
```

Figure 1: Resultado obtenido del data.shape y data.head.

Después se obtuvo la cantidad de canciones que obtuvieron y no top 1.

```
top
0    494
1    141
dtype: int64
```

Figure 2: Cantidad de canciones que entraron al top 1 y no.

Posteriormente, se crearon diversas gráficas para visualizar la información.

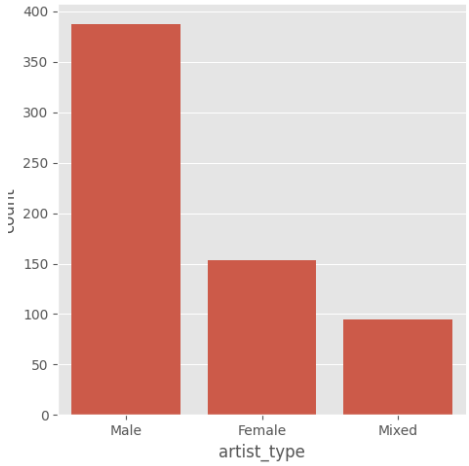

Figure 3: Artista por género.

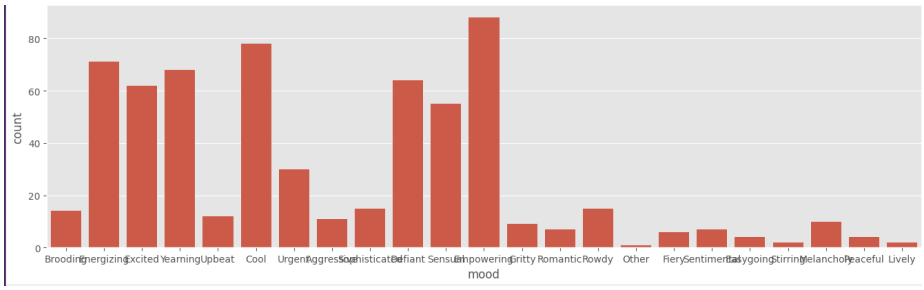

Figure 4: Canciones por mood.

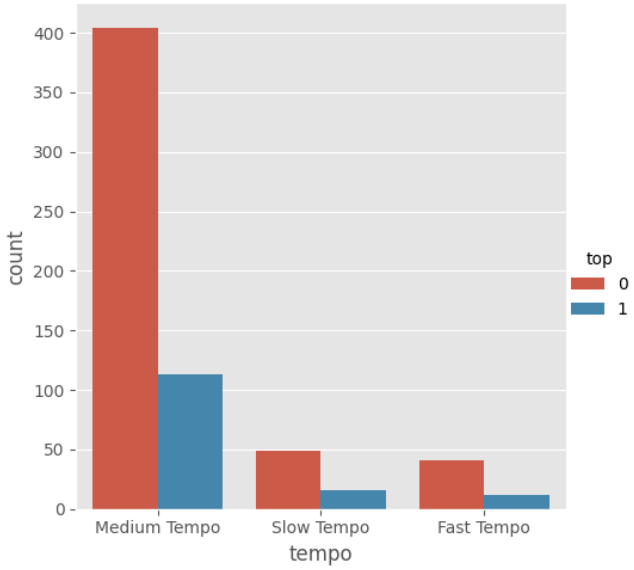

Figure 5: Canciones por tempo.

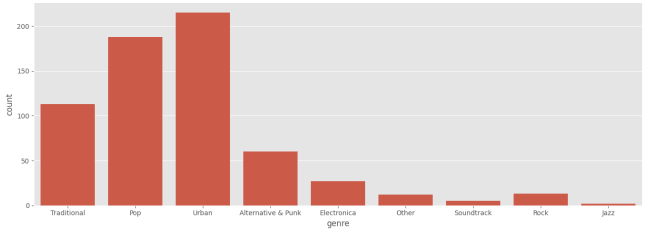

Figure 6: Canciones por género.

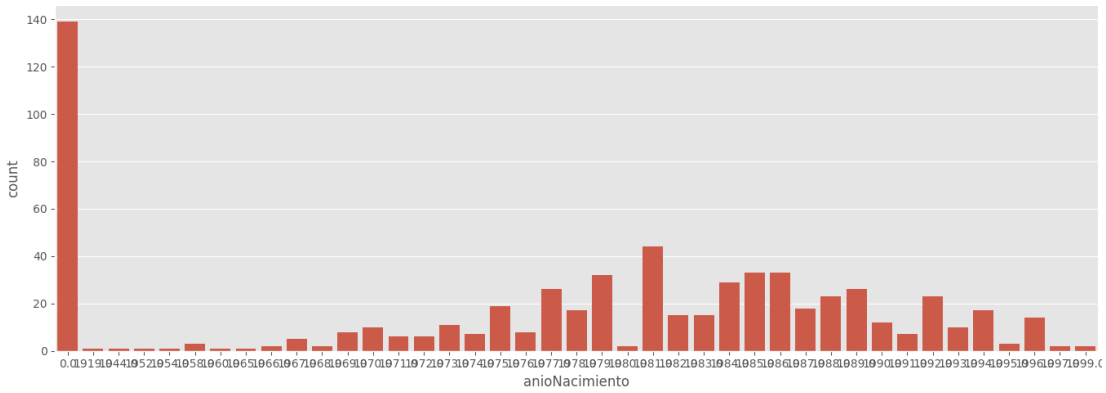

Figure 7: Canciones por año de nacimiento.

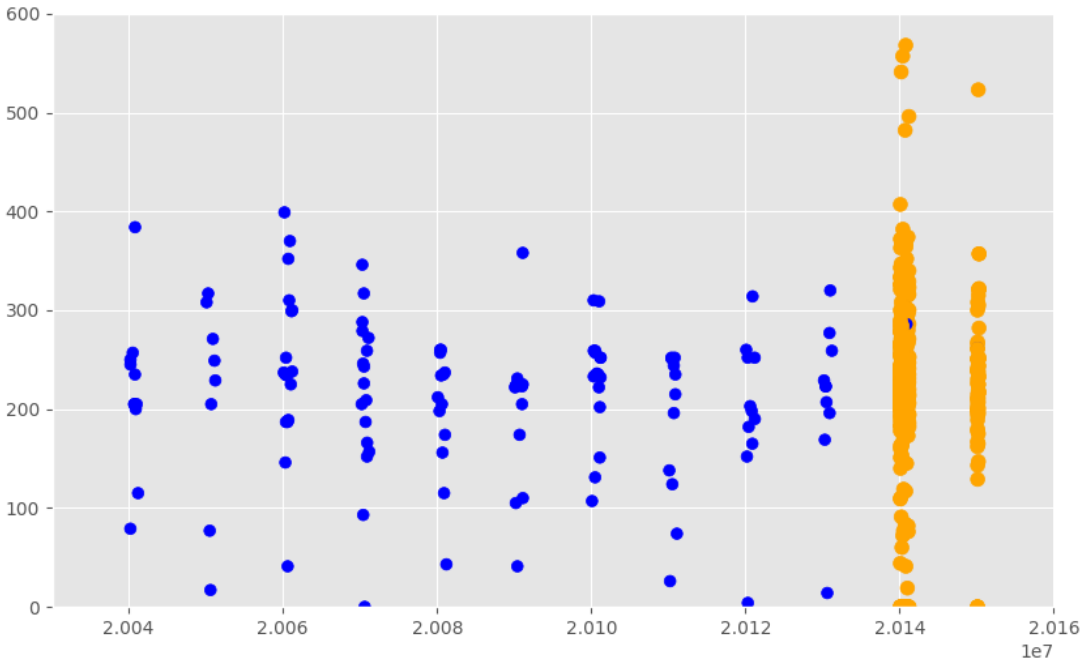

Figure 8: Canciones que entraron al top por año.

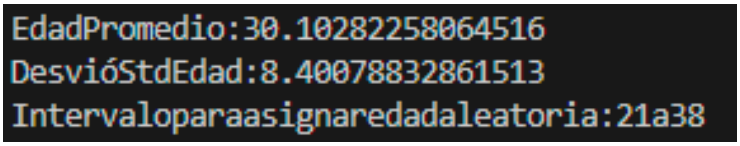

Figure 9: Promedio de edad.

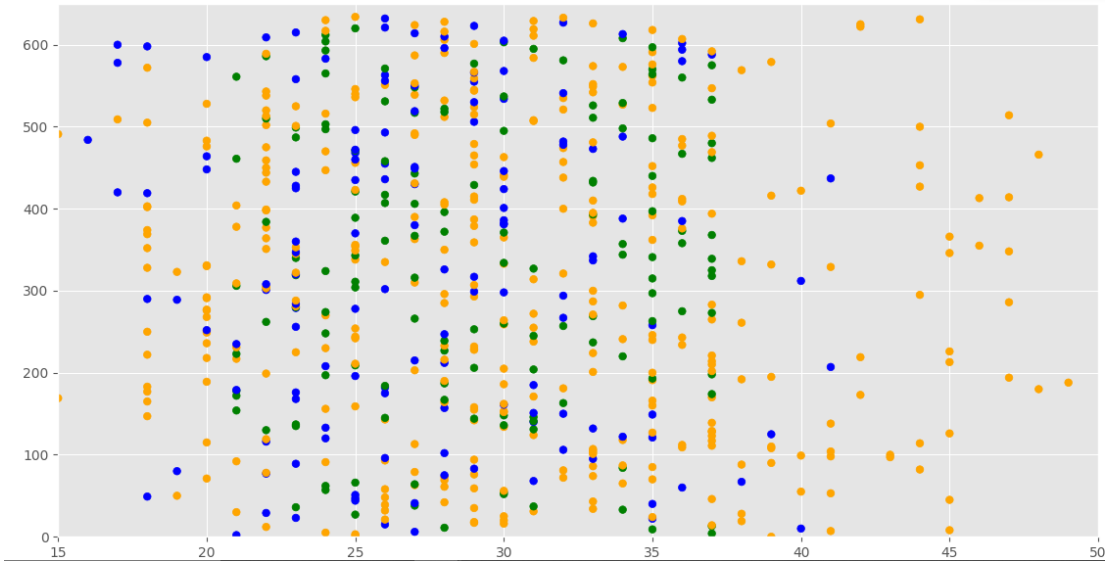

Figure 10: Canciones por año de nacimiento arreglado.

	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

Figure 11: Mood mapeado.

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

Figure 12: Artista mapeado.

	genreEncoded	top		
		mean	count	sum
0	0	0.088608	79	7
1	1	0.050000	40	2
2	2	0.008850	113	1
3	3	0.319149	188	60
4	4	0.330233	215	71

Figure 13: Género mapeado.

	tempoEncoded	top		
		mean	count	sum
0	0	0.222047	635	141

Figure 14: Tempo mapeado.

	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

Figure 15: Duración mapeado.

	edadEncoded	top		
		mean	count	sum
0	0.0	0.261538	65	17
1	1.0	0.296296	162	48
2	2.0	0.238994	159	38
3	3.0	0.178218	202	36
4	4.0	0.042553	47	2

Figure 16: Edad mapeado.

MaxDepth	AverageAccuracy
1	0.556101
2	0.556126
3	0.564038
4	0.642584
5	0.623834
6	0.636235
7	0.645709

Figure 17: Max Depth.

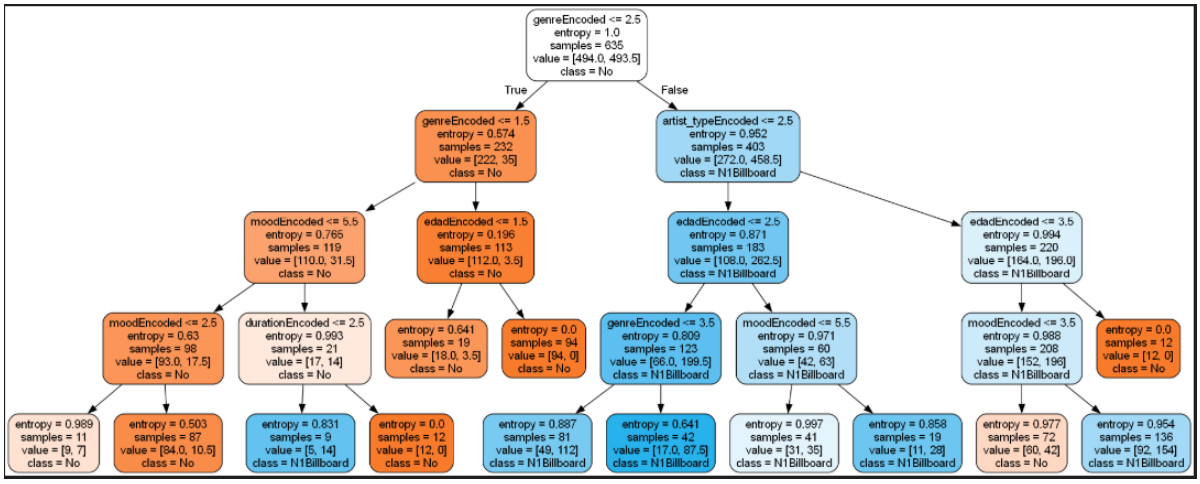

Figure 18: Árbol visualizado.

```
Precisión del árbol:  64.88
```

Figure 19: Precisión del árbol.

```
Prediccion: [1]
Probabilidad de Acierto Havana: 83.73%
```

Figure 20: Predicción para Havana.

```
Prediccion: [0]
Probabilidad de Acierto Believer: 88.89%
```

Figure 21: Predicción para Believer.

Como se observa, se obtuvo que la canción Havana de Camila Cabello ingresaría la top 1 con una probabilidad de acierto de 83.73%. Así bien, la canción Believer de Imagine Dragons no entraría al top 1 con una precisión del 88.89%. Ambos resultados coinciden con la realidad de los datos y tienen una probabilidad de acierto relativamente buena.

4 Conclusiones

Considero que la realización de esta actividad fue primordial para comprender el tema de árbol de decisión en Machine Learning. Durante la codificación fue necesario hacer algunos ajustes en el código especificado por el libro ya que este marcaba algunos errores. Se tuvieron que instalar paqueterías, cambiar comandos y agregar algunas líneas para que el código se ejecutara correctamente. Aún así, estos cambios no representaron una gran dificultad y ayudaron a tener una mejor comprensión del código.

5 Referencias

Bagnato, J. (2019). Aprende Machine Learning. Leanpub.
Materiales de clase (2025). UANL.