# DECISIÓN TREES

CODIFICAR ANTES SI HACE FALTA

**Frequency Encoding (Codificación por Frecuencia)**

**Si tienes muchos valores únicos y no quieres crear muchas nuevas columnas:**

**Explicación: Esto asigna a cada categoría un valor basado en cuántas veces aparece.**

**artists\_cartelera['artist\_encoded'] = artists\_cartelera['artist'].map(artists\_cartelera['artist'].value\_counts())**

**2. Label Encoding (Codificación por Etiquetas)**

**Si tienes pocos valores únicos y quieres representar cada uno como un número (sin crear muchas columnas nuevas):**

**df['artist\_encoded'] = df['artist'].factorize()[0]**

**One-Hot Encoding (Codificación One-Hot)**

**Si quieres evitar cualquier relación ordinal entre categorías, pero no te importa crear muchas columnas**

**df = pd.get\_dummies(df, columns=['artist'], drop\_first=True)**

**Codificación Manual (Ejemplo de "Tempo")**

**Si quieres asignar valores específicos de forma manual (cuando tienes un conjunto pequeño y conocido de categorías)**

**df['tempo\_encoded'] = df['tempo'].map({"Slow Tempo": 0, "Medium Tempo": 1, "Fast Tempo": 2})**

**NUEVA BASE CON SOLO LOS CAMPOS QUE INTERESAN**

**artists\_cartelera = artists\_cartelera.drop(columns=['artist','mood','tempo','genre','artist\_type'])**

## 1.LIBRERIAS

### 1.1 CLASSIFIER

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

### 1.2 Regressión

from sklearn.tree import DecisionTreeRegressor

## 2. Definir variables objetivo

X = df.drop(['class'], axis=1)

y = df['class']

## 3. Dividir los datos en conjunto de train y test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.33, random\_state = 42)

## 4. Dimensión y codificación variables:

array\_1d.reshape(-1, 1)

ara modelos de machine learning, incluidas técnicas como árboles de decisión, las variables independientes (X\_train y X\_test) deben estar en formato bidimensional,

**# Para verificar las dimensiones de train**

**X\_train.shape, X\_test.shape**

## 

## 5. Defino modelos y parametros

# instantiate the DecisionTreeClassifier model with criterion gini index

DecisionTree = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=0)

## 6. TRAIN MODELO

# fit the model

DecisionTree.fit(X\_train, y\_train)

## 7. Predicciones

y\_pred = DecisionTree.predict(X\_test)

## 8.METRICAS classificación

**librerias: from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_auc\_score**

ACCURACY, RECALL, PRECISION, F1

# Cálculo de las métricas

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

# Imprimir los resultados

print("Precisión:", precision)

print("Recall:", recall)

print("F1-Score:", f1)

print("Accuracy:", accuracy)

Matriu confusion

matriu\_confusion = confusion\_matrix(y\_test, y\_pred)

print("Matriz de Confusión:\n", matriu\_confusion)

## 

## 9. Verificación Overfiting y Underfiting

y\_test\_pred =DecisionTree.predict(X\_train)

train\_accuracy = accuracy\_score(y\_train, y\_test\_pred)

print(train\_accuracy)

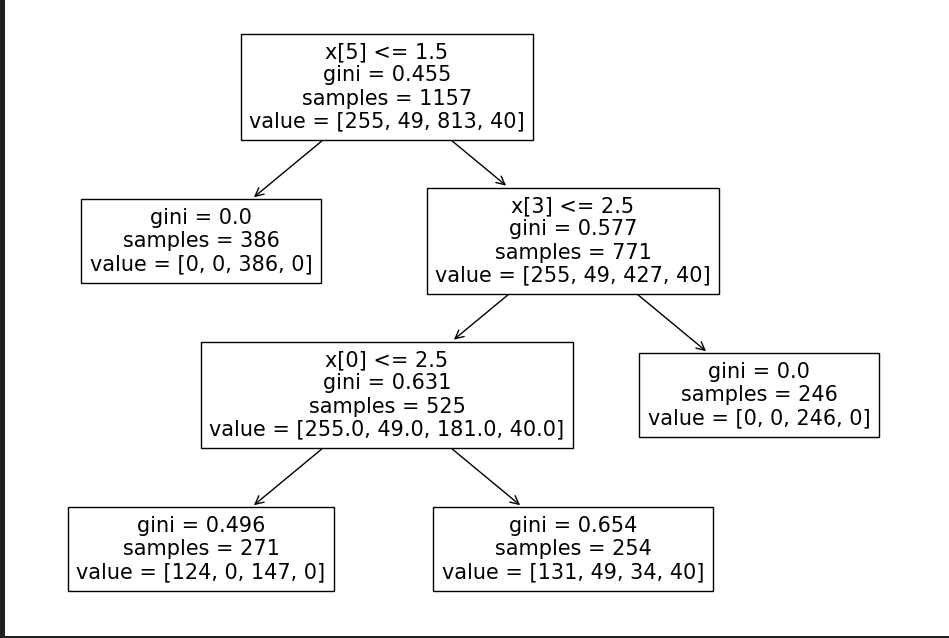
print(test\_accuracy)

## 10. Visualización:

plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot\_tree(DecisionTree.fit(X\_train, y\_train))



## 

## 11. Saber parametros optimos

param\_grid = {

'max\_depth': [3, 5, 10, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 5, 10],

'criterion': ['gini', 'entropy'],

'max\_leaf\_nodes': [None, 10, 20, 50],

'max\_features': [None, 'sqrt', 'log2'],

'class\_weight': [None, 'balanced']

}

grid\_search = GridSearchCV(

estimator=DecisionTreeClassifier(random\_state=42),

param\_grid=param\_grid,

scoring='accuracy',

cv=5, # Validació creuada

verbose=1,

n\_jobs=-1 # Utilitzar tots els cores disponibles

)

grid\_search.fit(X\_train, y\_train)

# Resultats del millor model

print("Millor combinació de paràmetres:", grid\_search.best\_params\_)

print("Millor puntuació d'accuràcia:", grid\_search.best\_score\_)

# Prediccions amb el millor model

best\_tree\_model = grid\_search.best\_estimator\_

y\_pred\_best = best\_tree\_model.predict(X\_test)

print("\nAccuracy del millor model:", accuracy\_score(y\_test, y\_pred\_best))

print("\nConfusion Matrix del millor model:\n", confusion\_matrix(y\_test, y\_pred\_best))

print("\nClassification Report del millor model:\n", classification\_report(y\_test, y\_pred\_best))