import numpy as np

import pandas as pd

from collections import Counter

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot\_tree

def calculate\_entropy(data):

    labels = data.iloc[:, -1]

    label\_counts = Counter(labels)

    total\_samples = len(labels)

    entropy = -sum((count / total\_samples) \* np.log2(count / total\_samples) for count in label\_counts.values())

    return entropy

def calculate\_gini(data):

    labels = data.iloc[:, -1]

    label\_counts = Counter(labels)

    total\_samples = len(labels)

    gini = 1 - sum((count / total\_samples) \*\* 2 for count in label\_counts.values())

    return gini

def equal\_width\_binning(data, num\_bins=4):

    binned\_data = pd.DataFrame()

    for column in data.columns[:-1]:

        binned\_data[column] = pd.cut(data[column], bins=num\_bins, labels=False)

    binned\_data[data.columns[-1]] = data.iloc[:, -1]

    return binned\_data

def best\_feature(data):

    base\_entropy = calculate\_entropy(data)

    max\_info\_gain = -1

    best\_feature = -1

    for column in data.columns[:-1]:

        values = data[column].unique()

        split\_entropy = sum((len(data[data[column] == value]) / len(data)) \* calculate\_entropy(data[data[column] == value]) for value in values)

        info\_gain = base\_entropy - split\_entropy

        if info\_gain > max\_info\_gain:

            max\_info\_gain = info\_gain

            best\_feature = column

    return best\_feature

def build\_decision\_tree(data):

    X = data.iloc[:, :-1]

    y = data.iloc[:, -1]

    clf = DecisionTreeClassifier(criterion='entropy')

    clf.fit(X, y)

    return clf

def visualize\_tree(clf, feature\_names):

    plt.figure(figsize=(10, 8))

    plot\_tree(clf, feature\_names=feature\_names, filled=True)

    plt.show()

data = pd.DataFrame({

    'Feature1': [1, 2, 3, 4, 5, 6, 7, 8],

    'Feature2': [5, 3, 4, 2, 7, 8, 1, 6],

    'Label': ['A', 'A', 'B', 'B', 'A', 'A', 'B', 'B']

})

print("Entropy:", calculate\_entropy(data))

print("Gini Index:", calculate\_gini(data))

print("Best Feature for Root Node:", best\_feature(data))

binned\_data = equal\_width\_binning(data)

print("Binned Data:", binned\_data)

clf = build\_decision\_tree(data)

visualize\_tree(clf, data.columns[:-1])  
  
 **OUTPUT:**

A diagram of a algorithm

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.