LAB ASSIGNMENT-6

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn import metrics

# Load the dataset

data = pd.read\_csv('diabetes.csv')

# Entropy Calculation (A1)

def calculate\_entropy(data):

    labels = data['Outcome']

    label\_counts = labels.value\_counts()

    probabilities = label\_counts / len(labels)

    entropy = -np.sum(probabilities \* np.log2(probabilities))

    return entropy

# Gini Index Calculation (A2)

def calculate\_gini\_index(data):

    labels = data['Outcome']

    label\_counts = labels.value\_counts()

    probabilities = label\_counts / len(labels)

    gini\_index = 1 - np.sum(probabilities \*\* 2)

    return gini\_index

# Equal Width Binning (A4)

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

    binned\_data = data.copy()

    for col in data.columns[:-1]:

        if data[col].dtype in ['float64', 'int64']:

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

    return binned\_data

# Information Gain Calculation for Root Node (A3)

def calculate\_information\_gain(data, feature):

    total\_entropy = calculate\_entropy(data)

    values, counts = np.unique(data[feature], return\_counts=True)

    weighted\_entropy = np.sum([(counts[i] / np.sum(counts)) \* calculate\_entropy(data[data[feature] == values[i]]) for i in range(len(values))])

    information\_gain = total\_entropy - weighted\_entropy

    return information\_gain

# Identify Root Node (A3)

def find\_root\_node(data):

    features = data.columns[:-1]

    gains = {feature: calculate\_information\_gain(data, feature) for feature in features}

    root\_node = max(gains, key=gains.get)

    return root\_node, gains

# Train Decision Tree (A5)

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

X = binned\_data.drop('Outcome', axis=1)

y = binned\_data['Outcome']

dt\_classifier = DecisionTreeClassifier(random\_state=42)

dt\_classifier.fit(X, y)

# Visualize the Decision Tree (A6)

plt.figure(figsize=(15,10))

plot\_tree(dt\_classifier, feature\_names=X.columns, class\_names=['No Diabetes', 'Diabetes'], filled=True)

plt.title('Decision Tree Visualization')

plt.show()

# Calculate and print entropy and Gini index

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

print('Gini Index:', calculate\_gini\_index(data))

# Determine the root node and print the information gains

root\_node, information\_gains = find\_root\_node(data)

print('Root Node:', root\_node)

print('Information Gains:', information\_gains)

Output:

Entropy: 0.9331343166407831

Gini Index: 0.45437282986111116

Root Node: DiabetesPedigreeFunction

Information Gains: {'Pregnancies': np.float64(0.061825341680179724), 'Glucose': np.float64(0.304201127153376), 'BloodPressure': np.float64(0.0593095796416776), 'SkinThickness': np.float64(0.08166434463609251), 'Insulin': np.float64(0.2770945287640211), 'BMI': np.float64(0.343810627126824), 'DiabetesPedigreeFunction': np.float64(0.6509177483835624), 'Age': np.float64(0.14094080665096176)}

A diagram of a network

AI-generated content may be incorrect.