# **DECISION TREE**

### **Aim**

To implement a decision tree and calculate the Gini index and information gain

### **Problem description**

This project involves implementing a binary decision tree classifier with a focus on calculating the Gini index and information gain for optimal node splitting.Functions for Gini index and information gain calculations will be included, and the project will use these to find the optimal split for each node.

### **Algorithm**

#### **1) Import the required libraries**

Import the python libraries numpy,pandas,sklearn.metrics,sklearn.model\_selection,sklearn.tree,sklearn.preprocessing,sklearn.compose

#### **2) Load the spam dataset**

Load the spam dataset **“**classifierdata.csv**”**

#### **3) Preprocessing the dataset**

#### **4) Create the Decision tree model using scikit learn**

#### **5) Train the decision tree model using the training data and calculate the gini index and information gain**

#### **6) Testing the decision tree model using the test data**

### **Program code/ Pseudocode**

import numpy as np

import pandas as pd

from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import OneHotEncoder, LabelEncoder

from sklearn.compose import ColumnTransformer

df = pd.read\_csv(r"C:\Users\91830\Desktop\DUK\AIML\DT\classifierdata.csv",

sep=',', header=None)

# Printing the dataset shape

print("Dataset Length: ", len(df))

print("Dataset Shape: ", df.shape)

print("")

print("Dataset: \n", df.head())

print("")

X = df.iloc[:, 1:]

y = df.iloc[:, 0]

enc = LabelEncoder()

y = enc.fit\_transform(y)

categorical\_cols = [0, 1, 2, 3] # assuming all columns are categorical

encoder = ColumnTransformer(transformers=[("OneHot", OneHotEncoder(), categorical\_cols)], remainder='passthrough')

X\_encoded = encoder.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.3, random\_state=100)

clf\_gini = DecisionTreeClassifier(criterion="gini", random\_state=100,

max\_depth=3, min\_samples\_leaf=5)

clf\_gini.fit(X\_train, y\_train)

clf\_entropy = DecisionTreeClassifier(criterion="entropy", random\_state=100,

max\_depth=3, min\_samples\_leaf=5)

clf\_entropy.fit(X\_train, y\_train)

y\_pred\_entropy = clf\_entropy.predict(X\_test)

print("Predicted values using Information Gain:")

print(enc.inverse\_transform(y\_pred\_entropy)) # Convert back to original labels for readability

print("")

y\_pred\_gini = clf\_gini.predict(X\_test)

print("Predicted values using Gini Index:")

print(enc.inverse\_transform(y\_pred\_gini)) # Convert back to original labels for readability

print("")

print("Information Gain")

print("Confusion Matrix (IG): \n", confusion\_matrix(y\_test, y\_pred\_entropy))

print("Accuracy (IG): ", accuracy\_score(y\_test, y\_pred\_entropy) \* 100)

print("Report (IG): \n", classification\_report(y\_test, y\_pred\_entropy, zero\_division=1))

print("")

print("Gini Index ")

print("Confusion Matrix (Gini): \n", confusion\_matrix(y\_test, y\_pred\_gini))

print("")

print("Accuracy (Gini): ", accuracy\_score(y\_test, y\_pred\_gini) \* 100)

print("Report (Gini): \n", classification\_report(y\_test, y\_pred\_gini, zero\_division=1))

### **Result**

### The trained decision tree model correctly predicts the gini index and information gain.