LAB - 2 :- ID3 Algorithm : (17/3/25)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import copy
dataset = pd.read_csv('/content/weather.csv')
X = dataset.iloc[:, :].values
X
attribute = ['outlook', 'Temp', 'Humidity', 'wind']

class Node(object):
    def __init__(self):
        self.value = None
        self.decision = None
        self.child = None

def findEntropy(data, rows):
    yes = 0
    no = 0
    ans = -1
    idx = len(data[0]) - 1
    entropy = 0

    for i in rows:
        if data[i][idx] == 'yes':
            yes = yes + 1
        else:
            no = no + 1

    x = yes / (yes + no)
    y = no / (yes + no)
    if x != 0 and y != 0:
        entropy = -1 * (x * math.log2(x) + y * math.log2(y))

    if x == 1:
        ans = 1
    if y == 1:
        ans = 0

    return entropy, ans
```

```
def findMaxGain(data, rows, columns):
    maxGain = 0
    retidx = -1
    entropy, ans = findEntropy(data, rows)
    if entropy == 0:
        return maxGain, retidx, ans
    for j in columns:
        mydict = {}
        idx = j
        for i in rows:
            key = data[i][idx]
            if key not in mydict:
                mydict[key] = 1
            else:
                mydict[key] = mydict[key] + 1
        gain = entropy

        for key in mydict:
            yes = 0
            no = 0
            for k in rows:
                if data[k][j] == key:
                    if data[k][-1] == 'yes':
                        yes = yes + 1
                    else
                        no = no + 1
            x = yes/(yes+no)
            y = no/(yes+no)
            if x != 0 and y != 0:
                gain += (mydict[key] * (x * math.log2(x) +
                        y * math.log2(y)))/14
        if gain > maxGain:
            maxGain = gain
            retidx = j

    return maxGain, retidx, ans
```

```
def buildTree(data, rows, columns):
    maxGain, idx, ans = findMaxGain(X, rows, columns)
    root = Node()
    root.childs = []
    if maxGain == 0:
        if ans == 1:
            root.value = 'Yes'
        else:
            root.value = 'No'
        return root
    root.value = attribute[idx]
    mydict = {}
    for i in rows:
        key = data[i][idx]
        if key not in mydict:
            mydict[key] = 1
        else:
            mydict[key] += 1
    newcolumns = copy.deepcopy(columns)
    newcolumns.remove(idx)
    for key in mydict:
        newrows = []
        for i in rows:
            if data[i][idx] == key:
                newrows.append(i)
        temp = buildTree(data, newrows, newcolumns)
        temp.decision = key
        root.childs.append(temp)
    return root

def traverse(root, level = 0):
    indent = "  " * level
    print(f"{indent}+ Decision: {root.decision}, Value: {root.value}")
    for i, child in enumerate(root.childs):
        if i == len(root.childs) - 1:
            traverse(child, level+1)
        else:
            traverse(child, level+1)
```

```
def Calculate():
    rows = [i for i in range(0,14)]
    columns = [i for i in range(0,4)]
    root = buildTree(X, rows, columns)
    root.decision = 'Start'
    traverse(root)

Calculate()
```

Output:

```
+ Decision : Start, Value : Outlook
  + Decision : Sunny, Value : Humidity
    + Decision : High, Value : No
    + Decision : Normal, Value : Yes
  + Decision : Overcast, Value : Yes
  + Decision : Rain, Value : Wind
    + Decision : Weak, Value : Yes
    + Decision : Strong, Value : No
```