Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

BE-ETRX B **Sub- AIML Lab**
Name: Shubham Sawant **UID :2019110050**

Name of the Experiment: **Decision Tree (ID3) algorithm**

Objective:
Write Python program to demonstrate the working of the decision tree based ID3 algorithm by using appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Outcomes:
1. Find entropy of data and follow steps of the algorithm to construct a tree.
2. Representation of hypothesis using decision tree.
3. Apply Decision Tree algorithm to classify the given data.
4. Interpret the output of Decision Tree.

System Requirements: Windows with MATLAB

Data Set Link: https://www.kaggle.com/code/kralmachine/analyzing-the-heart-

Dataset Description:

Number of Instances: 14
Number of Attributes (including the class attribute): 8
Attribute Information:

- Age (age in years)
- Sex (1 = male; 0 = female)
- Height (Below 150 = Low, 150 – 180 = Normal, Above ,180 = High)
- Weight (Below 40 = Low, 40 – 80 = Normal, Above 80 = Over, Above 120 = Obese)
- FPS (fasting blood sugar > 120 mg/dl) (1 = true; 0= false)
- TRESTBPS (resting blood pressure (in mm Hg on admission to the hospital))
- CHOL (serum cholesterol in mg/dl)
- ACTIVE(1or0)

Algorithm:
The decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Entropy:

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

E(S) is the Entropy of the entire set, while the second term E(S, A) relates to an Entropy of an attribute A.

BE-ETRX B
Name: Shubham Sawant
Sub- AIML Lab
UID :2019110050

$$E(S) = \sum_{x \in X} -P(x) \log_2 P(x) \qquad E(S, A) = \sum_{x \in X} [P(x) * E(S)]$$

Information Gain:

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

$$IG(S, A) = E(S) - E(S, A)$$

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import copy
```

[2] ✓ 5.8s

```
# dataset = pd.read_csv(r'C:\Users\Dell\Desktop\Shubham\SEM7\AIML\EXP3\heart.csv')
dataset = pd.read_csv(r'C:\Users\Dell\Desktop\Shubham\SEM7\AIML\EXP3\cardio_train.csv')
X = dataset.iloc[:, 1:].values
print(X)
attribute = ['age', 'sex', 'height', 'weight','trtbps','bloodsugar','chol','active']
# attribute = ['age','sex','cp','trtbps','chol','fbs','restecg','thalachh','exng','oldpeak','slp','caa','thall','output']
```

[3] ✓ 0.1s

```
[['young' 'm' 'short' 'normal' 'normal' 'normal' 'active' 'Yes']
 ['young' 'f' 'tall' 'normal' 'normal' 'normal' 'active' 'Yes']
 ['old' 'f' 'average' 'normal' 'low' 'normal' 'not-active' 'Yes']
 ['adult' 'm' 'average' 'over' 'low' 'high' 'not-active' 'No']
 ['adult' 'f' 'short' 'over' 'low' 'high' 'not-active' 'No']
 ['adult' 'f' 'short' 'normal' 'normal' 'low' 'not-active' 'No']
 ['old' 'f' 'short' 'normal' 'high' 'high' 'active' 'No']
 ['young' 'm' 'tall' 'obese' 'high' 'normal' 'active' 'Yes']
 ['young' 'f' 'short' 'normal' 'low' 'normal' 'active' 'No']
 ['adult' 'f' 'average' 'obese' 'low' 'normal' 'not-active' 'No']
 ['young' 'f' 'average' 'obese' 'low' 'low' 'not-active' 'No']
 ['old' 'm' 'tall' 'normal' 'low' 'low' 'active' 'No']
 ['old' 'm' 'average' 'normal' 'low' 'low' 'not-active' 'No']
 ['adult' 'f' 'short' 'over' 'high' 'high' 'active' 'No']]
```

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

BE-ETRX B **Sub- AIML Lab**
Name: Shubham Sawant **UID :2019110050**

```
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:
        """if ans == 1:
            print("Yes")
        else:
            print("No")"""
        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

        # print(mydict)
        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
```

```python
                x = yes/(yes+no)
                y = no/(yes+no)
                # print(x, y)
                if x != 0 and y != 0:
                    gain += (mydict[key] * (x*math.log2(x) + y*math.log2(y)))/14
            # print(gain)
            if gain > maxGain:
                # print("hello")
                maxGain = gain
                retidx = j

    return maxGain, retidx, ans

def buildTree(data, rows, columns):

    maxGain, idx, ans = findMaxGain(X, rows, columns)
    root = Node()
    root.childs = []
    # print(maxGain
    #
    # )
    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)
        # print(newrows)
        temp = buildTree(data, newrows, newcolumns)
        temp.decision = key
        root.childs.append(temp)
    return root


def traverse(root):
    print(root.decision)
    print(root.value)

    n = len(root.childs)
    if n > 0:
        for i in range(0, n):
            traverse(root.childs[i])


def calculate():
    rows = [i for i in range(0, 14)]
    columns = [i for i in range(0, 6)]
    root = buildTree(X, rows, columns)
    root.decision = 'Start'
    traverse(root)

calculate()
```

Interpretation of output:

```
... Person is fit ?
    └── BloodSugar
        ├── high
        │   └── NO
        ├── low
        │   └── NO
        └── normal
            └── BloodPressure
                ├── high
                │   └── YES
                ├── low
                │   └── Age
                │       ├── adult
                │       │   └── NO
                │       ├── old
                │       │   ├── NO
                │       │   └── YES
                │       └── young
                └── normal
                    └── YES
```

Conclusion:

- We learned how to make a decision tree out of the given Dataset
- We learned to identify the root node, leaf node and connecting node
- We learned to calculate the entropy of the dataset and information gain of each attribute to decide the root node and subsequently the leaf nodes