Experiment 5

AIM : Decision Tree using ID3

To build a Decision Tree using ID3 algorithm.

DESCRIPTION:

A **decision tree** is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision.

In decision tree learning, **ID3 (Iterative Dichotomiser 3)** is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset using **Entropy** and **Information Gain**.

Entropy $H(S)$ is a measure of the amount of uncertainty in the (data) set S (i.e. entropy characterizes the (data) set S S).

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Information gain $IG(A)$ is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A.

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$$

- S – The current (data) set for which entropy is being calculated
- X – Set of classes in S
- $p(x)$ – The proportion of the number of elements in class x to the number of elements in set S
- $H(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$

Algorithm :

```
ID3 algorithm (Split (node, data):
    1. A <- the best attribute for splitting the data ( having highest
       Information Gain )
    2. Decision attribute for this node <- A
    3. For each value of A, create new child node
    4. Split training data to child nodes
    5. For each child node / subset:
            if subset is pure: STOP
            else: Split (child_node, {subset} )
    END
```

CODE and OUTPUT :

In [1]:
```
# Making Decision Tree Using ID3 Algorithm
# Libraries Needed
import numpy as np
import math
import pandas as pd
```

In [2]:
```
# Node object for a Decision Tree
class node(object):
    def __init__(self, data=None):
        self.child = {}
        self.data = data
```

In [3]:
```
# To print a Tree
def printTree(root, Val, level):
    if root == None:
        return
    print(str('\t'*level)  +str(Val)+'->'+ root.data)
    for child in root.child:
        printTree(root.child[child],child,level+1)
    return
```

In [4]:
```
# Read data into pandas dataframe
data=pd.read_csv("tennis.csv",delimiter=',')
data
```

Out[4]:

	Outlook	Temperature	Humidity	Windy	PlayTennis
0	Sunny	Hot	High	False	No
1	Sunny	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Rainy	Mild	High	False	Yes
4	Rainy	Cool	Normal	False	Yes
5	Rainy	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Sunny	Mild	High	False	No
8	Sunny	Cool	Normal	False	Yes
9	Rainy	Mild	Normal	False	Yes
10	Sunny	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Rainy	Mild	High	True	No

In [5]:
```python
# Differentiate data into Target and Labels
X=data.values[:,:4]
Y=data.values[:, 4]
label = ['Outlook','Temperature','Humidity','Windy']
target = 'PlayTennis'
```

In [6]:
```python
# Count number instances of a class present in a data or a list
def countDistinct(att):
    c={}
    for val in att.unique():
        c[val]=att[att==val].count()
    return c
countDistinct(data.Outlook)
```

Out[6]:
```
{'Sunny': 5, 'Overcast': 4, 'Rainy': 5}
```

In [7]:
```python
# Calculation of Entropy
def Entropy(df):
    c = countDistinct(df)
    n = df.count()
    E = 0
    for val in c:
        E = E - ((c[val]/n)*math.log2(c[val]/n))
    return E

Entropy(data[target][data[label[0]] == 'Sunny'])
```

Out[7]:
```
0.9709505944546686
```

In [8]:
```python
# Calculation of Information Gain using Entropy
def InfoGain(data,att):
    infoG = Entropy(data[target])
    n = data[att].count()
    values = countDistinct(data[att])
    for val in values:
        infoG = infoG - ((values[val]/n)
                         *Entropy(data[target][data[att]==val]))
    return infoG

InfoGain(data,'Outlook')
```

Out[8]:
```
0.2467498197744391
```

In [9]:
```python
# Implementing ID3 and generating a tree using Information Gain
def id3(data,labels,target):
    n = data[target].count()
    vals = countDistinct(data[target])
    for val in vals:
        if vals[val] == n:
            return node(data=val)
    info = {}
    for att in labels:
        info[att] = InfoGain(data,att)
    A = sorted(info,key = lambda x:x[1],reverse=True)[0]
    Root = node(data=A)
    for val in data[A].unique():
        Root.child[val] = id3(data[data[A]==val],
                              [x for x in labels if x!=A],
                              target)
    return Root

Root = id3(data,label,target)
```

In [10]:
```python
# Final Decision Tree using ID3 algorithm
printTree (Root,'Root', 0)
```

```
Root->Outlook
        Sunny->Humidity
                High->No
                Normal->Yes
        Overcast->Yes
        Rainy->Humidity
                High->Windy
                        False->Yes
                        True->No
                Normal->Windy
                        False->Yes
                        True->No
```

In [11]:
```python
# Testing with sample data i.e. Outlook = Rainy,
# Humidity = High, Windy = True and Temperature = Mild
Root.child['Rainy'].child['High'].child[ True].data
```

Out[11]:
```
'No'
```

LEARNING OUTCOMES:

In this Experiment we learned how we can efficiently classify data in a decision tree using ID3 Algorithm