Experiment 6

AIM : Decision Tree using CART

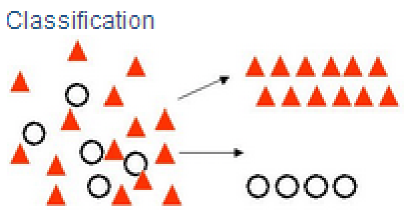
To build a Decision Tree using CART algorithm.

DESCRIPTION:

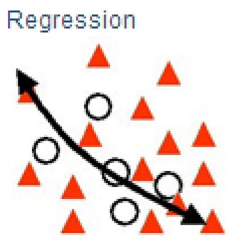
A **decision tree** is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision.

CART - Classification And Regression Trees methodology was introduced in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone as an umbrella term to refer to the following types of decision trees:

Classification Trees - where the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.



Regression Trees - where the target variable is continuous and tree is used to predict it's value.



The **CART algorithm** is structured as a sequence of questions, the answers to which determine what the next question, if any should be. The result of these questions is a tree like structure where the ends are terminal nodes at which point there are no more questions.

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

$$I_G(p) = 1 - \sum_{i=1}^{J} {p_i}^2$$

Algorithm :

```
ID3 algorithm (Split (node, data):
    1. A <- the best attribute for splitting the data ( having Lowest Gini Index
 )
    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} )
```

CODE and OUTPUT :

In [1]:

```python
# Making Decision Tree Using CART Algorithm
# Libraries Needed
import numpy as np
import math
import pandas as pd
```

In [2]:

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

In [3]:

```python
# 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]:

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

In [6]:

```
# 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]:

```
# Calculation of GINI index
def Gini(df):
    c = countDistinct(df)
    n = df.count()
    E = 1
    for val in c:
        E = E - ((c[val]/n)*(c[val]/n))
    return E

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

Out[7]:

```
0.48
```

In [8]:

```
# Calculation of Average GINI index using GINI index of individual value in att
def AverageGini(data,att):
    infoG = 0
    n = data[att].count()
    values = countDistinct(data[att])
    for val in values:
        infoG = infoG + ((values[val]/n)*Gini(data[target][data[att]==val]))
    return infoG

AverageGini(data,'Outlook')
```

Out[8]:

```
0.34285714285714286
```

In [9]:

```
# Implementing cart and generating a tree using Average GINI index
def cart(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] = AverageGini(data,att)
    A = sorted(info,key = lambda x:x[1])[0]
    Root = node(data=A)
    for val in data[A].unique():
        Root.child[val] = cart(data[data[A]==val],[x for x in labels if x!=A],target)
    return Root

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

In [10]:

```
# Final Decision Tree using CART algorithm
printTree (Root,'Root', 0)
```

```
Root->Temperature
        Hot->Windy
                False->Outlook
                        Sunny->No
                        Overcast->Yes
                True->No
        Mild->Windy
                False->Outlook
                        Rainy->Yes
                        Sunny->No
                True->Outlook
                        Sunny->Yes
                        Overcast->Yes
                        Rainy->No
        Cool->Windy
                False->Yes
                True->Outlook
                        Rainy->No
                        Overcast->Yes
```

In [11]:

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

Out[11]:

```
'No'
```

LEARNING OUTCOMES:

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