Expt. No. : 5
Date : 13 / 07 / 21

Apply the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and unpruned tree.

Aim:-

To apply the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and unpruned tree.

Program:-

```
import numpy as np
import math
from data_loader import read_data
class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""
    def __str__(self):
        return self.attribute
def subtables(data, col, delete):
    dict = {}
    items = np.unique(data[:, col])
    count = np.zeros((items.shape[0], 1), dtype = np.int32)
    for x in range(items.shape[0]):
        for y in range(data.shape[0]):
```

```
            if data[y, col] == items[x]:
                count[x] += 1
        for x in range(items.shape[0]):
            dict[items[x]] = np.empty(int(count[x]), data.shape[1]),
                                                    dtype="|S32")
            pos = 0
            for y in range(data.shape[0]):
                if data[y, col] == items[x]:
                    dict[items[x]][pos] = data[y]
                    pos += 1
            if delete:
                dict[items[x]] = np.delete(dict[items[x]], col, 1)
        return items, dict
def entropy(S):
    items = np.unique(S)
    if items.size == 1:
        return 0
    counts = np.zeros(items.shape[0], 1))
    sums = 0
    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size * 1.0)
    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums
def gain_ratio(data, col):
    items, dict = subtables(data, col, delete, False)
```

```
    total_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    intrinsic = np.zeros((items.shape[0], 1))
    for x in range(items.shape[0]):
        ratio = dict[items[x]].shape[0] / (total_size * 1.0)
        entropies[x] = ratio * entropy(dict[items[x]][:,-1])
        intrinsic[x] = ratio * math.log(ratio, 2)
    total_entropy = entropy(data[:,-1])
    iv = -1 * sum(intrinsic)
    for x in range(entropies.shape[0]):
        total_entropy -= entropies[x]
    return total_entropy / iv
def create_node(data, metadata):
    if (np.unique(data[:,-1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:,-1])[0]
        return node
    gains = np.zeros((data.shape[1] - 1, 1))
    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)
    split = np.argmax(gains)
    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)
    items, dict = subtables(data, split, delete = True)
    for x in range(items.shape[0]):
        child = create_node(dict[items[x]], metadata)
```

```
            node.children.append((items[x], child))
        return node
def empty(size):
    s=""
    for x in range(size):
        s+=""
    return s
def print_tree(node, level):
    if node.answer!="":
        print(empty(level), node.answer)
        return
    print(empty(level), node.attribute)
    for value, n in node.children:
        print(empty(level+1), value)
        print_tree(n, level+2)
metadata, traindata = read_data("tennis.csv")
data=np.array(traindata)
node=create_node(data, metadata)
print_tree(node, 0)
```

Data_loader.py

```
import csv
def read_data(filename):
    with open(filename, 'r') as csvfile:
        datareader = csv.reader(csvfile, delimiter=',')
        headers=next(datareader)
```

```
    metadata = []
    traindata = []
    for name in headers:
        metadata.append(name)
    for row in datareader:
        traindata.append(row)
return(metadata, traindata)
```

Tennis.csv

outlook, temperature, humidity, wind,
answer sunny, hot, high, weak, no
sunny, hot, high, strong, no
overcast, hot, high, weak, yes
rain, mild, high, weak, yes
rain, cool, normal, weak, yes
rain, cool, normal, strong, no
overcast, cool, normal, strong, yes
sunny, mild, high, weak, no
sunny, cool, normal, weak, yes
rain, mild, normal, weak, yes
sunny, mild, normal, strong, yes
overcast, mild, high, strong, yes
overcast, hot, normal, weak, yes
rain, mild, high, strong, no

Result:-

Thus the program to apply the technique of pruning for a noisy data monk 2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and unpruned tree has been executed successfully.

Output :-

```
outlook, [illegible]
    overcast
        b'yes'
    rain
        wind
            b'strong'
                b'no'
            b'weak'
                b'yes'
    sunny
        humidity
            b'high'
                b'no'
            b'normal'
                b'yes'
```