ID3 algorithm

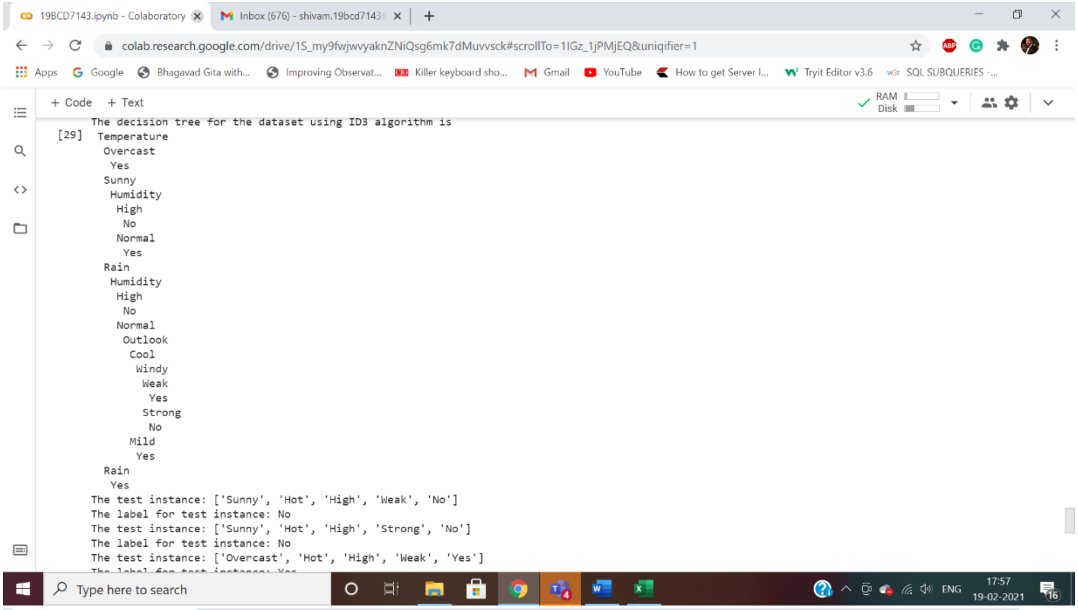
Code

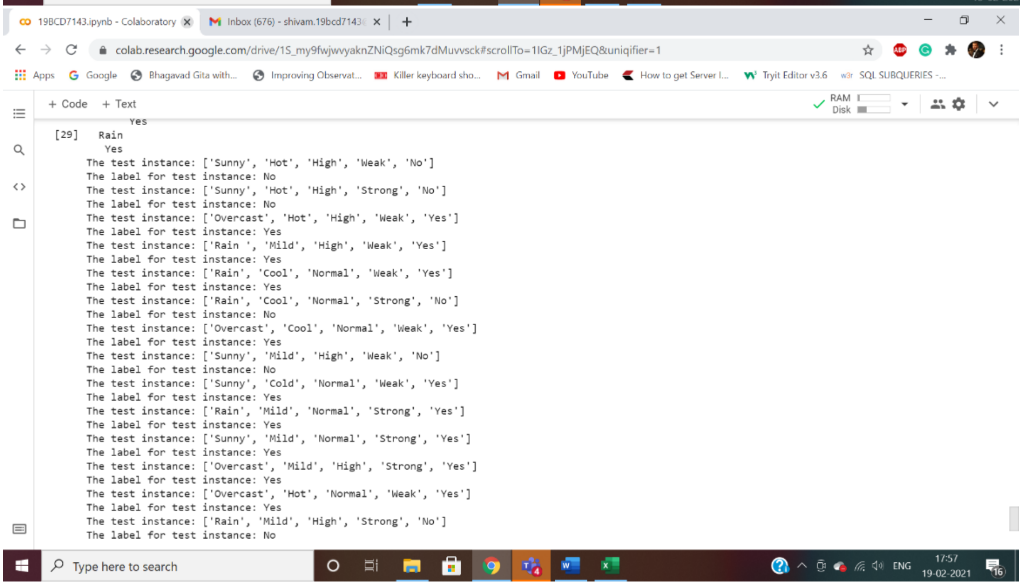
```
import math
import csv
def load_csv(PlayTennis):
   lines=csv.reader(open(PlayTennis,"r"));
   dataset = list(lines)
   headers = dataset.pop(0)
   return dataset,headers
class Node:
  def __init__(self,attribute):
    self.attribute=attribute
    self.children=[]
    self.answer=""
def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))
    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
    for x in range(len(attr)):
     for y in range(r):
      if data[y][col]==attr[x]:
         counts[x]+=1
    for x in range(len(attr)):
      dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
      pos=0
      for y in range(r):
        if data[y][col]==attr[x]:
          if delete:
            del data[y][col]
          dic[attr[x]][pos]=data[y]
          pos+=1
    return attr,dic
def entropy(S):
    attr=list(set(S))
    if len(attr)==1:
        return 0
    counts=[0,0]
    for i in range(2):
      counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)
    sums=0
    for cnt in counts:
      sums+=-1*cnt*math.log(cnt,2)
    return sums
```

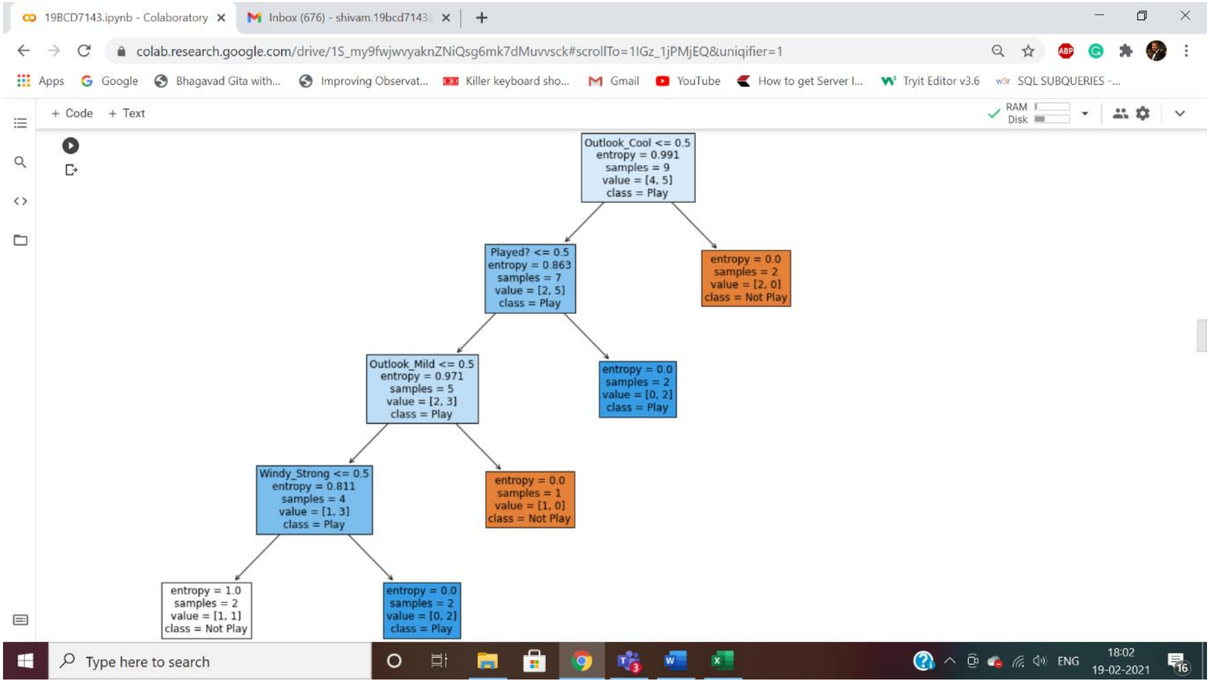
```
def compute_gain(data,col):
  attr,dic = subtables(data,col,delete=False)
  total_size=len(data)
  entropies=[0]*len(attr)
  ratio=[0]*len(attr)
  total_entropy=entropy([row[-1] for row in data])
  for x in range(len(attr)):
      ratio[x]=len(dic[attr[x]])/(total_size*1.0)
      entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
      total_entropy-=ratio[x]*entropies[x]
  return total_entropy
def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol)))==1:
       node=Node("")
       node.answer=lastcol[0]
       return node
    n=len(data[0])-1
    gains=[0]*n
    for col in range(n):
       gains[col]=compute_gain(data,col)
    split=gains.index(max(gains))
    node=Node(features[split])
    fea = features[:split]+features[split+1:]
    attr,dic=subtables(data,split,delete=True)
    for x in range(len(attr)):
      child=build_tree(dic[attr[x]],fea)
      node.children.append((attr[x],child))
    return node
def print_tree(node,level):
    if node.answer!="":
      print("  "*level,node.answer)
      return
    print("  "*level,node.attribute)
    for value,n in node.children:
      print("  "*(level+1),value)
      print_tree(n,level+2)
def classify(node,x_test,features):
    if node.answer!="":
      print(node.answer)
      return
    pos=features.index(node.attribute)
    for value, n in node.children:
      if x_test[pos]==value:
        classify(n,x_test,features)
'''Main program'''
dataset,features=load_csv("PlayTennis.csv")
node1=build_tree(dataset,features)
```

```python
print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
testdata,features=load_csv("PlayTennis.csv")
for xtest in testdata:
  print("The test instance:",xtest)
  print("The label for test instance:",end=" ")
  classify(node1,xtest,features)
```

OUTPUT







Observations

ID3 determines the information gain for each candidate attribute (i.e., Outlook, Temperature, Humidity, and Wind), then selects the one with highest information gain. S denotes the collection of training examples. Outlook is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., Sunny, Overcast, and Rain).

Gain(S,Outlook)=0.246

Gain(S,Humidity)=0.151

Gain(S,Wind)=0.048

Gain(S,Temperature)=0.029

Conclusion

So, decision tree algorithms transform the raw data into rule based mechanism. They can use nominal attributes whereas most of common machine learning algorithms cannot. However, it is required to transform numeric attributes to nominal in ID3. Besides, its evolved version C4.5 exists which can handle nominal data. Even though decision tree algorithms are powerful, they have long training time. On the other hand, they tend to fall over-fitting. Besides, they have evolved versions named random forests which tend not to fall over-fitting issue and have shorter training times.