

ML_LAB
PROGRAM_4

```
import math

import csv

def load_csv(filename):

    lines=csv.reader(open(filename,"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("testdata.csv")
```

```
node1=build_tree(dataset,features)
```

```
print("The decision tree for the given dataset using ID3 algorithm is")
```

```
print_tree(node1,0)
```

```
testdata,features=load_csv("test.csv")
```

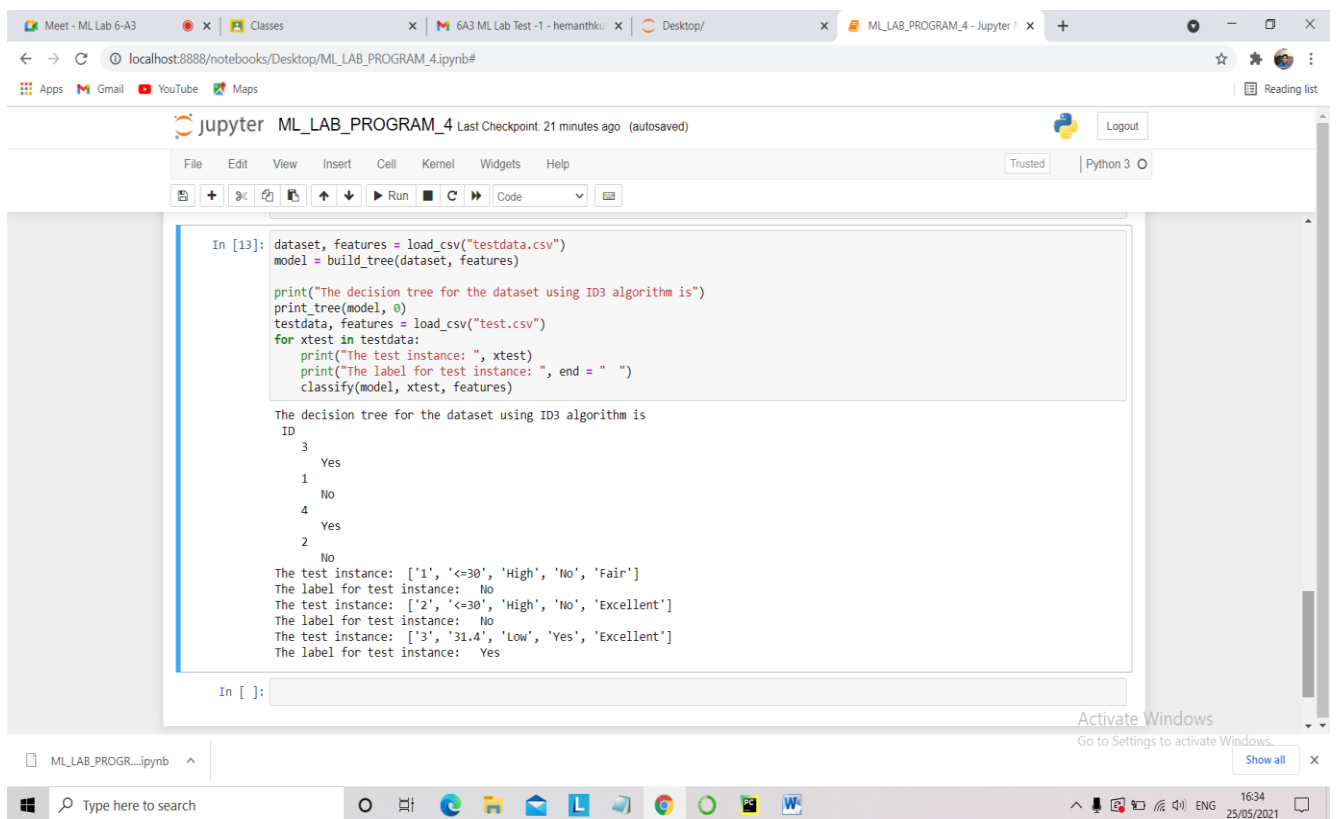
```
print()
```

for xtest in testdata:

```
    print("The test instance:",xtest)
```

```
    print("The label for test instance:",end="  ")
```

```
    classify(node1,xtest,features)
```



The screenshot shows a Jupyter Notebook titled "ML_LAB_PROGRAM_4" running on a local host. The notebook contains a code cell with the following Python code:

```
In [13]: dataset, features = load_csv("testdata.csv")
         model = build_tree(dataset, features)

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

The output of the code is displayed below the code cell:

```
The decision tree for the dataset using ID3 algorithm is
ID
3
  Yes
1
  No
4
  Yes
2
  No
The test instance: ['1', '<=30', 'High', 'No', 'Fair']
The label for test instance:  No
The test instance: ['2', '<=30', 'High', 'No', 'Excellent']
The label for test instance:  No
The test instance: ['3', '31.4', 'Low', 'Yes', 'Excellent']
The label for test instance:  Yes
```

The Jupyter Notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations, and a status bar at the bottom showing the current file name and search bar.