BINDIYA C.M., 4MT17CS028

PAGE NO : 4
DATE : 27/10/20
EXP. NO. : 3

3. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate dataset for building the decision tree and apply this knowledge to classify a new sample.

```
import pandas as pd
from pandas import DataFrame
df_tennis = pd.read_csv('lab3.csv')

attribute_names = list(df_tennis.columns)
attribute_names.remove('Play Tennis')
print(attribute_names)

def entropy_of_list(lst):
    from collections import Counter
    count = Counter(x for x in lst)
    num_instances = len(lst)*1
    probs = [x/num_instances for x in count.values()]
    return entropy(probs)

def entropy(probs):
    import math
    return sum([-prob*math.log(prob,2) for prob in
                probs])

total_entropy = entropy_of_list(df_tennis['Play Tennis'])
```

```
def information_gain(df, split_attribute_name, target_
                     attribute_name, trace=0):
    df_split = df.groupby(split_attribute_name)
    nobs = len(df.index) * 1
    df_agg_ent = df.split.agg({target_attribute_name:
                 [entropy_of_list, lambda x: len(x)/nobs]})
    df_agg_ent.columns = ['Entropy', 'propobservation']
    new_entropy = sum(df_agg_ent['Entropy'] * df_agg_ent
                  ['propobservation'])
    old_entropy = entropy_of_list(df[target_attribute_name])
    print(split_attribute_name, 'IG: ', old_entropy -
                                        new entropy)
    return old_entropy - new_entropy

def id3(df, target_attribute_name, attribute_names,
        default_class = None):
    from collections import Counter
    count = Counter(x for x in df[target_attribute_
            name])
    if len(count)==1:
        return next(iter(count))
    elif df.empty or (not attribute_names):
        return default_class
    else:
        default_class = max(count.keys())
        gain = [
            information_gain(df, attr, target_attribute_
            name) for attr in attribute_names
        ]
```

```
index_of_max = gain.index(max(gain))
best_attr = attribute_names[index_of_max]

tree = {best_attr:{}}

remaining_attribute_names = [i for i in attribute_
                             names if i != best_attr]

for attr_val, data_subset in df.groupby(best_attr):
    subtree = id3(data_subset, target_attribute_name,
                  remaining_attribute_names, default_class)
    tree[best_attr][attr_val] = subtree
return tree

from pprint import pprint
tree = id3(df_tennis, 'Play Tennis', attribute_names)
print("\n The Resultant Decision Tree is: \n")
pprint(tree)
```

Output:-

['Outlook', 'Temperature', 'Humidity', 'Wind']

Outlook IG: 0.246749819774391

Temperature IG: 0.029922256565 8954647

Humidity IG: 0.15183550136234136

Wind IG: 0.048127030408 26927

Temperature IG: 0.019973094021 97489

Humidity IG: 0.019973094021 97489

Wind IG: 0.9709505944546686

Temperature IG: 0.5709505944546686

Humidity IG: 0.9709505944546686

Wind IG: 0.019973094021 97489

The Resultant Decision Tree is:

{'Outlook': {'Overcast': 'Yes',
'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}},
'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}

Dataset used:

Outlook	Temperature	Humidity	Wind	Play Tennis
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