GALWIN BRITO 4MT17CS039

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('PlayTennis.csv')
attribute_names = list(df_tennis.columns)
attribute_names.remove('PlayTennis')
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([-probs * math.log
        (prob, 2) for prob in probs)]
```

GALWIN BRITO 4MT17CS039

PAGE NO : 6
DATE : 27/10/20
EXP.NO. : 3

```
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', 'propobservations']
    new_entropy = sum(df_agg_ent['Entropy']*
        df_agg_ent['propobservations'])
    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])
```

ARUN'S

Scanned with CamScanner

```
  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
```

GALWIN BRITO 4MT17CS039

PAGE NO : 8
DATE : 27/10/20
EXP.NO. : 3

```
tree = id3 (df-tennis, 'Play Tennis', attribute
            -names)
print ("\n The resultant Decision
        Tree is :\n")
pprint (tree)
```

GALWIN BRITO 4MT17CS039

Output:

```
['outlook', 'Temperature', 'Humidity', 'wind']

outlook IG : 0.2467498197744391
Temperature IG : 0.029222565658954647
Humidity IG : 0.15183550136234136
wind IG : 0.04812703040826927

Temperature IG : 0.01997309402197489
Humidity IG : 0.01997309402197489
wind IG : 0.9709505944546686

Temperature IG : 0.5709505944546686
Humidity IG : 0.9709505944546686
wind IG : 0.01997309402197489

The resultant Decision Tree is:
{'outlook': {'overcast': 'Yes', 'Rain':
        {'wind': {'strong': 'No', 'weak'
                : 'Yes'}},
'sunny': {'Humidity': {'High': 'No',
        'Normal': 'Yes'}}}}
```