In [1]:

```
import pandas as pd
import math

df = pd.read_csv('PlayTennis.csv')
print("\n Input Data Set is:\n", df)

t = df.keys()[-1]
print('Target Attribute is: ', t)
attribute_names = list(df.keys())
attribute_names.remove(t)
print('Predicting Attributes: ', attribute_names)


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

def entropy_of_list(ls,value):
    from collections import Counter
    cnt = Counter(x for x in ls)
    total_instances = len(ls)
    probs = [x / total_instances for x in cnt.values()]
    return entropy(probs)

def information_gain(df, split_attribute, target_attribute,battr):
    df_split = df.groupby(split_attribute)
    glist=[]
    for gname,group in df_split:
        glist.append(gname)

    glist.reverse()
    nobs = len(df.index) * 1.0
    df_agg1=df_split.agg({target_attribute:lambda x:entropy_of_list(x, glist.pop())})
    df_agg2=df_split.agg({target_attribute :lambda x:len(x)/nobs})

    df_agg1.columns=['Entropy']
    df_agg2.columns=['Proportion']


    new_entropy = sum( df_agg1['Entropy'] * df_agg2['Proportion'])
    if battr !='S':
```

```
        old_entropy = entropy_of_list(df[target_attribute],'S-'+df.iloc[0][df.columns.get_loc(battr)])
    else:
        old_entropy = entropy_of_list(df[target_attribute],battr)
    return old_entropy - new_entropy

def id3(df, target_attribute, attribute_names, default_class=None,default_attr='S'):

    from collections import Counter
    cnt = Counter(x for x in df[target_attribute])


    if len(cnt) == 1:
        return next(iter(cnt))


    elif df.empty or (not attribute_names):
        return default_class


    else:

        default_class = max(cnt.keys())
        gainz=[]
        for attr in attribute_names:
            ig= information_gain(df, attr, target_attribute,default_attr)
            gainz.append(ig)

        index_of_max = gainz.index(max(gainz))
        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, remaining_attribute_names,default_class,best_attr)
            tree[best_attr][attr_val] = subtree
        return tree

    from pprint import pprint
tree = id3(df,t,attribute_names)
print("\nThe Resultant Decision Tree is:")
print(tree)
```

```
def classify(instance, tree,default=None):
    attribute = next(iter(tree))
    if instance[attribute] in tree[attribute].keys():
        result = tree[attribute][instance[attribute]]
        if isinstance(result, dict):
            return classify(instance, result)
        else:
            return result
    else:
        return default

df_new=pd.read_csv('PlayTennisTest.csv')
df_new['predicted'] = df_new.apply(classify, axis=1, args=(tree,'?'))
print(df_new)
```

```
 Input Data Set is:
      Outlook Temperature Humidity    Wind PlayTennis
0      Sunny         Hot     High    Weak         No
1      Sunny         Hot     High  Strong         No
2   Overcast         Hot     High    Weak        Yes
3       Rain        Mild     High    Weak        Yes
4       Rain        Cool   Normal    Weak        Yes
5       Rain        Cool   Normal  Strong         No
6   Overcast        Cool   Normal  Strong        Yes
7      Sunny        Mild     High    Weak         No
8      Sunny        Cool   Normal    Weak        Yes
9       Rain        Mild   Normal    Weak        Yes
10     Sunny        Mild   Normal  Strong        Yes
11  Overcast        Mild     High  Strong        Yes
12  Overcast         Hot   Normal    Weak        Yes
13      Rain        Mild     High  Strong         No
Target Attribute is:  PlayTennis
Predicting Attributes:  ['Outlook', 'Temperature', 'Humidity', 'Wind']

The Resultant Decision Tree is:
{'Outlook': {'Overcast': 'Yes', 'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}}, 'Sunny': {'Humidity': {'High': 'N
o', 'Normal': 'Yes'}}}}
  Outlook Temperature Humidity  Wind PlayTennis predicted
0   Sunny         Hot     High  Weak          ?        No
1    Rain        Mild     High  Weak          ?       Yes
```

In [ ]:

In [ ]:

In [ ]: