ID3 algorithm

In [1]:
```
import pandas as pd
import numpy as np
```

In [6]:
```
dataset=pd.read_csv('program2.csv',names=['outlook','temperature','humidity','wind','class',])
attributes=('Outlook','Temperature','Humidity','Wind','PlayTennis')
```

In [7]:
```
def entropy(target_col):
    elements,counts=np.unique(target_col,return_counts=True)
    entropy=np.sum([(-counts[i]/np.sum(counts))*np.log2(counts[i]/np.sum(counts))
        for i in range(len(elements))])
    return entropy
```

In [9]:
```
def InfoGain(data,split_attribute_name,target_name="class"):
    total_entropy=entropy(data[target_name])
    vals,counts=np.unique(data[split_attribute_name],return_counts=True)
    Weighted_entropy=np.sum([(counts[i]/np.sum(counts))*entropy(data.where(data[split_attribute_name]==vals[i]).dropna()[target_name]) for i in
    range(len(vals))])
    Information_Gain=total_entropy-Weighted_entropy
    return Information_Gain
```

In [14]:
```
def ID3(data,originaldata,features,target_attribute_name="class",parent_node_class=None):
    if len(np.unique(data[target_attribute_name]))<=1:
        return np.unique(data[target_attribute_name])[0]
    elif len(data)==0:
        return
        np.unique(originaldata[target_attribute_name])[np.argmax(np.unique(originaldata[target_attibute_name],return_counts=True)[1])]
    elif len(features)==0:
        return parent_node_class
    else:
        parent_node_class=np.unique(data[target_attribute_name])[np.argmax(np.unique(data[target_attribute_name],return_counts=True)[1])]
        item_values=[InfoGain(data,feature,target_attribute_name) for feature
        in features]
        best_feature_index=np.argmax(item_values)
        best_feature=features[best_feature_index]
        tree={best_feature:{}}
        features=[i for i in features if i!=best_feature]
        for value in np.unique(data[best_feature]):
            value=value
            sub_data=data.where(data[best_feature]==value).dropna()
            subtree=ID3(sub_data,dataset,features,target_attribute_name,parent_node_class)
            tree[best_feature][value]=subtree
    return(tree)
```

In [15]:
```
def predict(query,tree,default=1):
    for key in list(query.keys()):
        if key in list(tree.keys()):
            try:
                result=tree[key][query[key]]
            except:
                return default
            result=tree[key][query[key]]
            if isinstance(result,dict):
                return predict(query,result)
            else:
                return result
```

In [16]:
```
def train_test_split(dataset):
    training_data=dataset.iloc[:14].reset_index(drop=True)
    return training_data
```

In [17]:
```
def test(data,tree):
    queries=data.iloc[:,:-1].to_dict(orient="records")
    predicted=pd.DataFrame(columns=["predicted"])
    for i in range(len(data)):
        predicted.loc[i,"predicted"]=predict(queries[i],tree,1.0)
    print('The predicted accuracy is:',(np.sum(predicted["predicted"]==data["class"])/len(data))*100,'%')

XX=train_test_split(dataset)
training_data=XX
tree=ID3(training_data,training_data,training_data.columns[:-1])
print('\nDisplay Tree\n',tree)
print('len=',len(training_data))
test(training_data,tree)
```

```
Display Tree
 {'outlook': {'Outlook': 'PlayTennis', 'Overcast': 'Yes', 'Rain': {'wind': {'Strong': 'No', 'Weak': 'Yes'}}, 'Sunny': {'humidity': {'High': 'No', 'Normal': 'Yes'}}}}
len= 14
The predicted accuracy is: 100.0 %
```