In [1]:
```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log
```

In [2]:
```
df = pd.read_csv('tennis.csv')
```

In [3]:
```
def find_entropy(df):
    target = df.keys()[-1]
    entropy = 0
    target_vars = df[target].unique()
    for v in target_vars:
        pi = df[target].value_counts()[v]/float(len(df[target]))
        entropy += -pi * np.log2(pi)
    return entropy
```

In [4]:
```
def find_attr_entropy(df,attr):
    target = df.keys()[-1]
    target_vars = df[target].unique()
    attr_vars = df[attr].unique()
    entropy2 = 0
    for v in attr_vars:
        entropy = 0
        for t in target_vars:
            num = len(df[attr][df[attr]==v][df[target]==t])
            den = len(df[attr][df[attr]==v])
            pi = num/(den+eps)
            entropy += -pi*np.log2(pi+eps)
        pi2 = den/len(df)
        entropy2 += -pi2 * entropy
    return abs(entropy2)
```

In [5]:
```
def get_winner(df):
    attr_ent = []
    ig = []
    for k in df.keys()[:-1]:
        ig.append(find_entropy(df)-find_attr_entropy(df,k))
    winner = np.argmax(ig)
    return df.keys()[:-1][winner]
```

In [6]:
```
def get_subtables(df,node,val):
    return df[df[node] == val].reset_index(drop=True)
```

In [7]:
```
def build_tree(df,tree=None):
    target = df.keys()[-1]
    node = get_winner(df)
    attr_vals = np.unique(df[node])
    if tree is None:
        tree = {}
        tree[node] = {}
    for v in attr_vals:
        subtable = get_subtables(df,node,v)
        tval,index = np.unique(subtable['play'],return_inverse=True)
        counts = np.bincount(index)
        if len(counts)==1:
            tree[node][v] = tval[0]
        else:
            tree[node][v] = build_tree(subtable)
    return tree
```

In [8]:
```
tree = build_tree(df)
import pprint
pprint.pprint(tree)
```

```
{'outlook': {'overcast': 'yes',
             'rainy': {'windy': {'Strong': 'no', 'Weak': 'yes'}},
             'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}}}
```

In [9]:
```
def predict(inst,tree):

    for nodes in tree.keys():
        value = inst[nodes]
        tree = tree[nodes][value]
        prediction = 0

        if type(tree) is dict:
            prediction = predict(inst, tree)
        else:
            prediction = tree
            break;

    return prediction
```

In [10]:
```
n = len(df)
count = 0
for i in range(n):
    pred = predict(df.iloc[i],tree)
    actual = df.iloc[i][-1]
    if pred == actual:
        count += 1
    print("predicted: {0}, actual: {1}".format(pred,actual))
print("accuracy: {0}%".format(count/n*100))
```

```
predicted: no, actual: no
predicted: no, actual: no
predicted: yes, actual: yes
predicted: yes, actual: yes
predicted: yes, actual: yes
predicted: no, actual: no
predicted: yes, actual: yes
predicted: no, actual: no
predicted: yes, actual: yes
predicted: yes, actual: yes
predicted: yes, actual: yes
predicted: yes, actual: yes
predicted: yes, actual: yes
predicted: no, actual: no
accuracy: 100.0%
```