In [6]:
```
import pandas as pd
import math
```

In [7]:
```
path = 'D:\\C\\docs\\DTU\\2. Second sem\\ML\\Lab\\4'
filename = 'DecisionTreeData'
filename+= '.csv'
df = pd.read_csv(path + '\\' + filename, index_col='Day')
df
```

Out[7]:

Day	Outlook	Temp	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

In [8]:
```
TARGET = df.columns[len(df.columns)-1]
TARGET
```

Out[8]:
```
'Play Tennis'
```

Find target values in a dataframe

In [9]:
```
def find_target(data) :
    # columns of dataframe
    cols = list(data.columns)

    # target values for the column
    tgt_val = cols[len(cols)-1]
    target = set(data[tgt_val].tolist())

    return tgt_val, target
```

Calculating entropy of S(Entire Dataframe)

In [10]:

```
def find_S(data) :
    tgt_val, target = find_target(data)

    S = 0
    count = {}

    for i in target :
        count[i]=0
    tgt = data[tgt_val].tolist()

    for i in tgt:
        count[i]+=1

    for val in count :
        tmp = count[val]/len(tgt)
        S-= tmp * math.log2(tmp)
    return S
```

Entropy for entire Dataset(For target values)

In [11]:
```
S = find_S(df)
S
```

Out[11]: 0.9402859586706311

Unique values in a column

In [12]:
```
def val_count(data, col, val) :
    return data.index[data[col]==val]
```

Unique values for every target

In [13]:
```
def tgt_count(data, col, val) :

    val_idx = val_count(data, col, val)
    temp = {}

    tgt_val, target = find_target(data)

    for i in target:
        temp[i]=0

    for idx in val_idx:
        temp[data.loc[idx][tgt_val]]+=1

    return temp
```

Function to find Entropy for a column value

In [14]:
```
def entropy(data, col, val) :
    temp = tgt_count(data, col, val)

    n = 0
    for i in temp:
        if temp[i]==0 :
            return 0
        n+=temp[i]

    ans = 0
```

```
    for i in temp:
        t = temp[i]/n
        ans-= t * math.log2(t)
    return ans
```

Function to find gain for a column

In [15]:
```
def gain(data, col) :
    ans = S

    tmp = set(data[col].tolist())

    tgt_val, target = find_target(data)

    for val in tmp:
        col_val = tgt_count(data, col, val)
        col_val_count = sum(col_val.values())

        ans-= col_val_count/len(data[tgt_val]) * entropy(data, col, val)
    return ans
```

In [16]:
```
def sample_run(df):
    cols = list(df.columns)
    features = cols[ : len(cols)-1]

    for feature in features :
        print(feature , ' : ', gain(df, feature))

sample_run(df)
```

```
Outlook  :  0.24674981977443933
Temp  :  0.02922256565895487
Humidity  :  0.15183550136234159
Wind  :  0.048127030408269544
```

Function for finding Max gain in a column

In [17]:
```
def get_max_gain_feature(data) :
    cols = list(data.columns)
    features = cols[ : len(cols)-1]

    max_gain = gain(data, features[0])
    max_gain_feature = features[0]

    for feature in features :
        gn = gain(data, feature)

        if gn > max_gain :
            max_gain = gn
            max_gain_feature = feature

    return max_gain_feature
```

Node class for a Column value

In [18]:
```
class Node :
    def __init__(self, col_name=None, edge=None, df=None, tgt_val=None) :
        self.col_name = col_name
        self.df = df
```

```
        self.edge = edge
        self.tgt_val = tgt_val
```

Recursive function for generating Decision Tree

In [19]:
```
adj_list = []
indx = 0

def generate_tree(data, index) :
    if len(set(data[TARGET])) <= 1 :
        return

    col_name = get_max_gain_feature(data)

    for col in set(data[col_name]) :
        df = data[data[col_name] == col]

        temp = Node(col_name, col, df)
        adj_list.append(temp)

        tgt_val = 'Many'
        data_tmp = list(df[TARGET])
        if len(set(data_tmp)) == 1 :
            tgt_val = data_tmp[0]

        temp.tgt_val = tgt_val

        generate_tree(df, index+1)
```

In [20]:
```
generate_tree(df, 0)
```

Printing Node in DFS manner along with feature value and Target value

In [21]:
```
for nn in adj_list :
    print('Feature Name : ', nn.col_name, ',     Feature Value : ', nn.edge, ",   Ta
```

```
Feature Name :  Outlook ,     Feature Value :  Sunny ,   Target Value :  Many
Feature Name :  Humidity ,     Feature Value :  Normal ,   Target Value :  Yes
Feature Name :  Humidity ,     Feature Value :  High ,   Target Value :  No
Feature Name :  Outlook ,     Feature Value :  Overcast ,   Target Value :  Yes
Feature Name :  Outlook ,     Feature Value :  Rain ,   Target Value :  Many
Feature Name :  Wind ,     Feature Value :  Weak ,   Target Value :  Yes
Feature Name :  Wind ,     Feature Value :  Strong ,   Target Value :  No
```